ESQUEMA PARA CONTRATAR PROYECTOS DE DESARROLLO DE SISTEMAS DE INFORMACIÓN

CONTRATO INTERADMINISTRATIVO N° 000376 DE 2015

Bogotá, Diciembre de 2015

FORMATO PRELIMINAR AL DOCUMENTO

Título:	**ESQUEMA PARA CONTRATAR PROYECTOS DE DESARROLLO DE SISTEMAS DE INFORMACIÓN**		
Fecha dd/mm/aaaa:	04/12/2015		
Resumen:	El presente documento presenta los lineamientos para mitigar los riesgos asociados con la contratación y gestión de proyectos de desarrollo de sistemas de información. Para esto se desarrollaran las fichas tipo que resuman los elementos técnicos más relevantes a tener en cuenta durante todo el ciclo de vida del desarrollo de sistemas de información.		
Palabras Claves:	Juicio de expertos, interesados, riesgo, gestión del alcance, requerimiento, diseño y arquitectura, calidad, pruebas, codificación, puesta en producción, uso y apropiación, mantenimiento		
Formato:	DOC		
Código:	No Aplica	Versión	1.6
Autor (es):	Catalina Vergara **Asesora Senior** Lucio Alfredo Riveros **Asesor Senior** Iván Darío Niño **Consultor Junior de TI**		
Revisó:	Ing. Javier Torres **Director Conceptual** **Corporación Colombia Digital** Alexandra Parra **Asesora Conceptual** **Corporación Colombia Digital** Diego Campos **Asesor Conceptual** **Corporación Colombia Digital**		
Aprobó:	Jorge Bejarano **Dirección Estándares y Arquitectura** **Ministerio de Tecnologías de la Información y las Comunicaciones** Johanna Pimiento **Dirección Gobierno en Línea** **Ministerio de Tecnologías de la Información y las Comunicaciones**		
Información Adicional:	No Aplica		
Ubicación:	No Aplica		

HISTORIA

VERSIÓN	FECHA	CAMBIOS INTRODUCIDOS
1.0	04/09/2015	Versión inicial
1.1	29/09/2015	Inclusión de los capítulos de codificación, puesta en producción, elemento transversal de gestión de la calidad y pruebas
1.2	23/10/2015	Inclusión de los capítulos de diseño y arquitectura, uso y apropiación y mantenimiento
1.3	03/11/2015	Ajustes a los capítulos de codificación y pruebas con la inclusión de un apartado sobre pruebas de seguridad
1.4	26/11/2015	Corrección de estilo y ajustes al documento
1.5	04/12/2015	Ajustes al documento por comentarios de MinTIC
1.6	23/12/2015	Corrección de estilo.

TABLA DE CONTENIDO

Tabla de contenido

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

SECCIONES PRELIMINARES

DERECHOS DE AUTOR

CRÉDITOS

Este documento se elabora dentro del marco de la ejecución del Contrato Interadministrativo N° 000376 de 2015 el cual tiene por objeto Contratar Servicios de acompañamiento especializado al Ministerio TIC en la implementación de las iniciativas: Fortalecimiento de la Gestión de TI en el estado y la Estrategia de Gobierno en línea . Este documento hace parte integral del desarrollo de la ejecución del Contrato Interadministrativo No. 000376 de 2015, definiendo las actividades y actores que en este intervienen.

AUDIENCIA DEL DOCUMENTO

Este documento es un instrumento aplicable por los actores del Contrato Interadministrativo No.000376 de 2015 el cual tiene por objeto “Contratar los servicios de acompañamiento especializado al Ministerio TIC en la implementación de las iniciativas: Fortalecimiento de la Gestión de TI en el estado y la Estrategia de Gobierno en línea.”

1 GLOSARIO

Acuerdo de niveles de servicio (ANS): Documento en el cual se establecen en términos medibles y cuantificables, todas las condiciones para la prestación de los servicios, las responsabilidades de las partes, el ofrecimiento estándar del servicio, las variables que miden la gestión, los indicadores de servicio, los estándares de rendimiento y los tipos de reportes.

Administrador de requerimientos: Es la persona encargada de definir con detalle los requerimientos (casos de uso, historia de usuario, requerimientos funcionales, etcétera). Este rol lo desempeña un profesional de ingeniería con conocimiento del dominio del problema.

Análisis de puntos de función: Método dirigido a medir el tamaño de la funcionalidad de un sistema de información. La medida es independiente de la tecnología y puede usarse como la base para calcular la productividad, la estimación de los recursos necesarios y el control del proyecto.

Análisis de riesgos: Proceso para evaluar y estimar su impacto y probabilidad de ocurrencia.

Arquitectura de TI: Describe la estructura y las relaciones de todos los elementos de TI de una organización. Se descompone en arquitectura de información, arquitectura de sistemas de información y arquitectura de servicios tecnológicos. Incluye además las arquitecturas de referencia y los elementos estructurales de la estrategia de TI (visión de arquitectura, principios de arquitectura, lineamientos y objetivos estratégicos).

Aseguramiento de la calidad: Ámbito de la gestión centrada en proporcionar confianza en el cumplimiento de los requisitos de calidad.

Automatización de pruebas: Uso del *software* para realizar o soportar actividades de prueba.

Calidad: Grado de satisfacción de los requisitos especificados y las necesidades y expectativas del usuario/cliente.

Caso de prueba: Conjunto de condiciones o variables bajo las cuáles un analista de pruebas determinará si una aplicación, un *software*, o una característica de éstos es parcial o completamente satisfactoria

Caso de uso: Es una descripción de los pasos o las actividades que deben realizarse para llevar a cabo un proceso. Las personas o entidades que participan se denominan actores.

En el contexto de ingeniería del *software*, un caso de uso es una secuencia de interacciones que se desarrollan entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

CMDB (Base de datos de la gestión de la configuración): Es un repositorio de información donde se relacionan todos los componentes de un sistema de información, ya sean hardware, *software,* documentación, etcétera.

CMMI: Es un modelo para medir la calidad del *software* y conocer la madurez de los procesos de su producción. Este modelo clasifica las empresas por niveles de madurez.

Comité de cambios (CAB-Change Advisory Board): Grupo de personas encargadas de evaluar, priorizar y programar los cambios.

Comité de cambios de emergencia (ECAB-Emergency Change Advisory Board): Grupo de personas que son convocadas cuando el CAB requiere tomar decisiones de emergencia y fuera de la programación.

Criterio de aceptación: Son un conjunto preciso y bien definido de condiciones que un producto que se va a adquirir o construir debe satisfacer en el momento de su entrega, para que sea aceptado por una entidad.

Defecto: Imperfección en un componente o sistema que puede causar un fallo.

Disponibilidad: Grado de disposición de un elemento para estar operativo y accesible para usarlo.

Eficiencia: Capacidad del *software* de proporcionar el rendimiento apropiado, en relación con la cantidad de recursos usados.

Ejecución de pruebas: Proceso para ensayar un componente o sistema bajo prueba, produciendo resultados reales.

Entorno de pruebas: Ambiente que incluye *hardware*, instrumentación, simuladores, herramientas de *software* y otros elementos de soporte necesarios para llevar a cabo una prueba.

Escalabilidad: Capacidad del *software* de ser mejorado para contener incrementos de carga.

Escenario de prueba: Describe paso a paso la funcionalidad de un caso de uso del producto o negocio, sin profundo detalle, con el objetivo de hacer entender cómo funciona este caso de uso.

Especificación de casos de prueba: Documento que especifica un conjunto de casos de prueba, incluye (objetivos, entradas, acciones de prueba, resultados esperados y precondiciones de ejecución).

Especificación de diseño de pruebas: Documento que especifica las condiciones de prueba para un elemento, el enfoque de pruebas detallado y los casos de alto nivel asociados.

Especificación de pruebas: Documento que especifica el diseño, los casos y procedimientos de prueba.

Estabilidad: Capacidad del *software* de evitar efectos inesperados por modificaciones.

Estimación del esfuerzo: Documenta el propósito de las actividades de medición y análisis, y especifica el tipo de acciones que se pueden realizar de acuerdo con los análisis de datos.

Estructura de desglose de trabajo (EDT): Es una agrupación de trabajo orientada a la entrega de los elementos del proyecto, que organiza y define el alcance del mismo.

Fiabilidad: Capacidad del *software* de funcionar de realizar las funciones requeridas satisfacer, bajo unas condiciones indicadas durante un periodo o un número operaciones concreto.

Funcionalidad: Capacidad del *software* de funcionar para satisfacer unas necesidades indicadas bajo unas condiciones específicas.

Gestión de la calidad: Es el conjunto de actividades que determina la calidad, los objetivos y las responsabilidades del equipo de trabajo; se implanta por medios tales como la planificación, el control, el aseguramiento (garantía) y la mejora en el marco del sistema de calidad.

Gestión de la configuración: Directrices técnicas y administrativas, que regulan la identificación y documentación de las características funcionales y físicas de un componente de configuración. Con este tipo de gestión se controlan los cambios de las características, se informa sobre el proceso de cambio y el estado de implementación, y se verifica la conformidad con los requisitos especificados.

Gestión de la configuración de *software* (GCS): Es el conjunto de actividades desarrolladas para gestionar los cambios a lo largo del ciclo de vida del desarrollo de *software*. La GCS es una actividad que garantiza la calidad que se aplica en todas las fases del proceso de ingeniería.

Gestión del cambio: Acciones con las que se apoya directamente el desarrollo y mantenimiento del *software*, mediante la conservación de la integridad del producto antes y después de su puesta en producción.

Gestión de pruebas: Proceso en el que se planifican, estiman, monitorizan y controlan las actividades de pruebas, normalmente llevadas a cabo por el jefe de pruebas.

Gestión de riesgos: Aplicación sistemática de procedimientos y prácticas para identificar, analizar, priorizar y controlar riesgos.

Índice de defectos: Puede expresarse en términos de número de defectos eliminados por miles de líneas de código o por puntos de función. El objetivo de este índice es reducir el número total de defectos de una versión a la siguiente, independientemente del tamaño.

Interoperabilidad: Capacidad del *software* de interactuar de manera coordinada con uno o más componentes o sistemas determinados.

Juicio de expertos: Opiniones dadas por profesionales expertos en proyectos de desarrollo de sistemas de información.

Líder funcional: Área o personas de la organización que participan activamente en la identificación de necesidades y en la definición de los requerimientos del sistema.

Línea base: Se refiere a una especificación o *software* que ha sido formalmente revisado o acordado, de tal forma que sirva como base para futuros desarrollos, y que sólo puede cambiarse por medio de un proceso formal de control de cambios.

Mantenibilidad: Facilidad con la que un *software* puede ser modificado para corregir defectos, satisfacer nuevos requisitos, modificar para hacer el mantenimiento futuro más sencillo, o adaptar a cambios de entorno.

Mantenimiento: Ajuste de un *software* en operación, para corregir defectos, mejorar el rendimiento u otros atributos, y adaptarlo a un entorno modificado.

Metodología Ágil: Procedimiento utilizado para obtener un desarrollo ágil de *software*. Es un marco conceptual que promueve iteraciones en el desarrollo a lo largo del ciclo de vida del proyecto.

Metodología / Procedimiento: En términos del desarrollo de *software*, es el conjunto de actividades relacionadas y sustentadas en un marco conceptual establecido y validado para lograr un objetivo: generar un producto o prestar de un servicio.

Middleware: Conecta componentes de *software* o aplicaciones para que puedan intercambiar datos entre ellos.

Modelo de desarrollo iterativo: Desarrollo del ciclo de vida de un proyecto, dividido en un número de iteraciones (usualmente cuantioso). Una iteración es una serie completa, que resulta en una nueva versión ejecutable de un producto, como uno de los elementos de un proyecto, que crece a partir de iteraciones.

Plan de pruebas: Documento que describe el alcance, enfoque, recursos y programación de las actividades de prueba. Identifica los elementos, las características por probar, las tareas, quien debe ejecutar, etcétera.

Plan Detallado de Trabajo (PDT): Es la lista detallada de actividades enfocada en cumplir los objetivos de cada etapa de una iniciativa, bajo un tiempo y con recursos estimados.

Portabilidad: Facilidad con la que un producto *software* puede ser trasladado de un entorno a otro.

Proceso de pruebas: Evaluación fundamental que comprende planificación, especificación, ejecución, registro, comprobación y actividades de cierre de las pruebas realizadas al *software.*

Pruebas alpha: Pruebas operativas reales o simuladas llevadas a cabo por usuarios/clientes potenciales o por un equipo de la organización, pero independiente del desarrollo del proyecto.

Prueba basada en requisitos: Enfoque de pruebas cuyos casos se diseñan con base en los objetivos y condiciones obtenidas de los requisitos de *software*.

Prueba beta: Revisión operativa realizada por usuarios/clientes potenciales y externos a la organización, para determinar si un componente o sistema satisface sus necesidades y encaja en los procesos de negocio.

Prueba de aceptación: Prueba formal de un sistema para medir si sus resultados satisfacen las necesidades de los usuarios, de los requisitos y los procesos de negocio, de tal manera que cumpla los criterios de aceptación.

Prueba de caja blanca: Análisis de la estructura interna de un componente o sistema.

Prueba de caja negra: Test, tanto funcional como no funcional, sin referencia de la estructura interna del componente o sistema.

Prueba de carga: Tipo de prueba centrada en medir el comportamiento de un componente o sistema para determinar la carga incremental que puede manejar, se revisa por ejemplo el número de usuarios paralelos y de transacciones.

Prueba de caso de uso: Técnica de diseño de caja negra (*appliance*) con la que los casos de prueba se formulan para ejecutar escenarios de usuario.

Prueba de estrés: Evaluación al límite de los requisitos especificados de un sistema o componente.

Prueba de funcionalidad: Verificación para determinar la funcionalidad de un *software*.

Prueba de integración: Exámenes realizados para detectar defectos en las interfaces e interacciones entre componentes o sistemas integrados.

Prueba de rendimiento: Proceso para determinar el rendimiento de un *software*.

Prueba de regresión: Revisión de una aplicación o sistema ya probados, que recibe modificaciones posteriores en el *software* o en su entorno. Este tipo de prueba se hace para asegurar que no se han hecho o identificado defectos en las áreas no intervenidas, como resultado de los cambios introducidos.

RUP (*Rational Unified Process*): Metodología de desarrollo de *software* utilizada para analizar, implementar y documentar sistemas orientados a objetos. Es adaptable al contexto y necesidad de cada organización. Divide el proceso en cuatro fases clave: inicio, elaboración, construcción y transición.

Requerimiento funcional: Define una función del sistema de *software* o de sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas.

Requerimiento no funcional: En la ingeniería de sistemas y de *software*, se refiere a un requisito que determina criterios para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que estos corresponden a los requerimientos funcionales. Incluye todos los requisitos que no describen información por guardar, ni funciones por realizar.

Riesgo residual: Es aquél que permanece después de que la dirección desarrolle sus respuestas a los riesgos. El riesgo residual refleja el riesgo remanente una vez se han implantado de manera eficaz las acciones planificadas por la dirección para mitigar el riesgo inherente.

Robustez: Grado con el que un componente o sistema puede funcionar correctamente en presencia de entradas inválidas o con condiciones de estrés del entorno.

Seguridad: Atributos del producto *software* que se relaciona con su capacidad para prevenir accesos no autorizados, bien sean accidentales o deliberados, a programas o datos.

Sistema de información / Solución informática / *Software* aplicativo: Conjunto de equipos, programas y procesos, con su documentación, que brindan una solución al procesamiento de información de un área o una Organización.

Scrum: Metodología ágil que aplica de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Utiliza la ejecución de iteraciones (*sprint*: bloques cortos y fijos), que proporcionan un resultado completo y, un incremento en la producción final que es entregado con mínimo esfuerzo al cliente en el tiempo planeado. Un principio clave de *Scrum* es reconocer que durante la ejecución de un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan.

Stakeholder: Define a los interesados en un proyecto de desarrollo de *software* y a todas aquellas personas o áreas que son afectados directa o indirectamente.

Tasa de defectos: Medida de fallos de una categoría dada en unidades de medida dada, por ejemplo fallos por unidad de tiempo o número de transacciones.

Usabilidad: capacidad del *software* de ser atractivo, entendido, aprendido y usado por el usuario.

2 INTRODUCCIÓN

2.1 ANTECEDENTES

El Ministerio de Tecnologías de Información y las Comunicaciones (MinTIC) busca optimizar el modelo de la adquisición de bienes y servicios de TI con el fin de lograr una mayor eficiencia y transparencia en el uso de los recursos del Estado. Dicha iniciativa se ha materializado en herramientas integrales que le brindan al Estado medios y criterios para hacer inversiones inteligentes con las que acceda a tecnología que realmente se ajuste a las necesidades de las entidades y que corresponda con las mejores opciones que ofrece el mercado en términos de calidad y costo.

La transformación que se está logrando a partir de esta iniciativa consiste en cambiar el enfoque tradicional de compra de bienes por un modelo de adquisición y gestión de servicios que se concentra en la maximización de la generación de valor a través de un esquema flexible, oportuno y con altos estándares de calidad. Esta transformación se apalanca en la construcción colectiva y en el desarrollo de competencias profesionales para obtener y gestionar de mejor modo los servicios.

2.2 DESAFÍOS

Como resultado del diagnóstico y análisis que se exponen en este documento, se identificó que el proceso de desarrollo de *software* es una labor que requiere un tratamiento con

profundidad, por tal razón la presente guía se focaliza en estructurar lineamientos, herramientas y recomendaciones al respecto.

El Ministerio TIC quiere transformar el proceso de desarrollo de *software* en las entidades del Estado desde cuatro aristas que permiten generar mayor valor para las instituciones y los ciudadanos: la apropiada planeación de los sistemas de información, la adecuada gestión de la información y los servicios, el óptimo aprovechamiento de los recursos disponibles y la eficiencia en la gestión del desarrollo de los sistemas de información.

Adoptar un modelo que optimice el desarrollo de *software* y garantice mejores niveles de calidad, le permite a las entidades apoyar el cumplimiento de sus objetivos misionales, alcanzar altos niveles de excelencia en términos de eficiencia administrativa, contar con información de calidad de forma oportuna y proveer herramientas que apoyen el proceso de toma de decisiones. La siguiente gráfica muestra el esquema de generación de valor en proyectos de desarrollo de *software*.

Figura 1. Esquema de generación de valor para proyectos de desarrollo de software. Fuente: Corporación Colombia Digital.

La planeación de los sistemas de información contempla la alineación estratégica de los proyectos de desarrollo de *software* con los procesos y planes de TI de la entidad de tal forma que exista coherencia entre las metas de alto nivel y los esfuerzos que hace cada una de las áreas y de los funcionarios. De igual forma, el proceso de planeación debe considerar los lineamientos de usabilidad, accesibilidad e interoperabilidad para que el *software* cumpla con las expectativas de la organización en términos de la generación de valor.

La apropiada gestión de la información y de los servicios, implica no perder de vista el aporte en valor que se busca con el proyecto de desarrollo de *software*; eso significa que se deben identificar las necesidades de los usuarios y demás involucrados con el fin de llenar sus expectativas. En este sentido, la entidad además debe considerar los atributos mínimos de calidad del *software* en términos de veracidad, confiabilidad, disponibilidad, seguridad, etcétera. Sumado a esto, la adecuada gestión de los recursos financieros, humanos y tecnológicos de la entidad es también un factor clave en el éxito de los proyectos de desarrollo pues soportan la materialización de cada una de las iniciativas y actividades planeadas.

El último eslabón del esquema de generación de valor es la gestión del desarrollo de los sistemas de información que hace referencia a cada una de las etapas del ciclo de vida de desarrollo de *software*. Es importante resaltar que la estrategia de mejoramiento que propone este documento se concentra en este último eslabón del proceso de generación de valor sin perder de vista la fuerte dependencia e interrelación que existe con los demás componentes del modelo.

Modelo de mercado

Este modelo representa todas las fuerzas vivas que participan en el proceso de desarrollo de *software* y se entienden principalmente a partir de la oferta y demanda, que interactúan bajo un esquema de adquisición en el que ambas partes buscan generar valor. La figura que se muestra a continuación ilustra el modelo de mercado descrito:

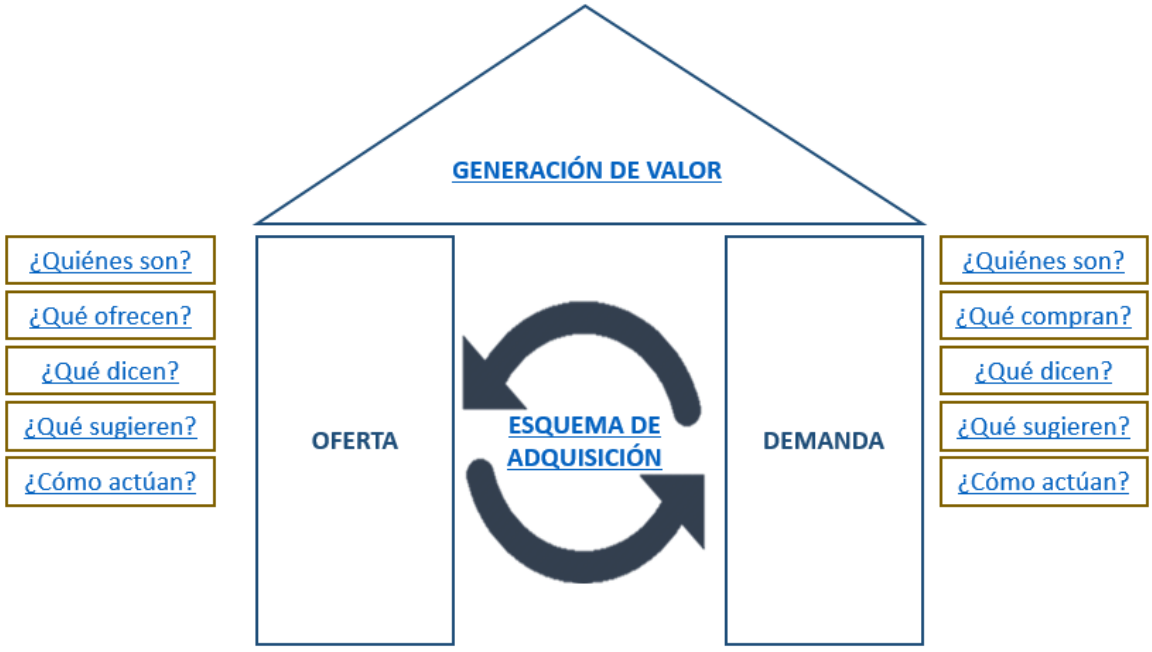


Figura 2. modelo de mercado. Fuente: Corporación Colombia Digital.

La oferta

Dentro de la investigación realizada para conocer los detalles de la oferta, se estableció que la componen empresas con los siguientes perfiles y portafolios de servicios:

- Empresas dedicadas al desarrollo de *software* a la medida.
- Fábricas de *software*.
- Empresas especializadas en el levantamiento y gestión de requerimientos.
- Empresas especializadas en mantenimiento de *software*.
- Empresas especializadas en la medición de esfuerzo y productividad.
- Empresas especializadas en ejecutar pruebas y garantizar la calidad del *software.*

- Empresas dedicadas a proveer recursos humanos especializados en proyectos de desarrollo de *software*.
- Desarrollo interno (*In-house)* para la organización.

Asimismo, a través de reuniones y entrevistas con diferentes representantes de la industria (Ver figura No. 3) fue posible identificar las siguientes oportunidades de mejora desde la perspectiva de la oferta:

- Con frecuencia se subestima el alcance y la complejidad de los proyectos de desarrollo de *software,* por lo que se deben hacer ajustes importantes durante su realización, que conllevan costos adicionales y demoras en el cronograma inicialmente planteado. En los casos más críticos, en los que no es posible hacer ajustes del presupuesto ni de los tiempos, los resultados son proyectos inconclusos que no generan valor.
- Los proveedores encuentran con alguna frecuencia que las entidades no cuentan con funcionarios capacitados para la gestión de proyectos de *software* o que quienes están disponibles y tienen el conocimiento y experiencia, están sobrecargados laboralmente.
- Los proveedores identifican que algunas entidades tienen altos índices de rotación de personal y esto altera el curso normal de los proyectos. Dentro del ciclo de vida la etapa de aprobaciones de los procesos de gestión de cambios y aceptación de pruebas son las más afectadas por dicha rotación.
- Los extensos tiempos para la contratación son considerados una difícil barrera contra el éxito de los proyectos de desarrollo de *software,* ya que con frecuencia el alcance o requerimientos establecidos inicialmente, están desactualizados al iniciar las fases de diseño y desarrollo.
- Algunos proveedores han participado en procesos licitatorios en los que observan que el presupuesto disponible es insuficiente para ejecutar el proyecto. Sin embargo, de manera preocupante, en la industria hay proveedores que participan en dichos procesos, a pesar de las condiciones, con el riesgo de desencadenar problemas para la entidad.

- Con mucha frecuencia, los proveedores observan que no es suficiente la participación del usuario final durante todo el ciclo de vida del proyecto, lo que conlleva a generar inconformidades importantes frente a las expectativas de la entidad.
- Los esquemas de gestión de cambios en los proyectos de desarrollo de *software* del Estado no cuentan con la flexibilidad que la naturaleza de este tipo de proyectos exige, debido a la rigidez que significan los procesos de contratación y a la informalidad con que se manejan los cambios en la organización.
- Las entidades usuarias no cuentan con ambientes de prueba, lo que dificulta y pone en riesgo la calidad del *software* que se utilizará en el ambiente de producción.
- Los cronogramas de trabajo muchas veces no incluyen una etapa para facilitarles a los empleados el uso y apropiación de los nuevos desarrollos, por lo tanto se presentan dificultades en las organizaciones.
- Los contratos no contemplan la curva de aprendizaje que debe surtir el proveedor para entender la naturaleza del negocio de la organización o los sistemas existentes.
- En algunos sistemas de información de las entidades, la documentación es insuficiente o inexistente, esto dificulta el trabajo de los proveedores.
- Algunas entidades buscan solucionar los problemas de falta de documentación con requerimientos de documentación muy detallados que no generan valor para la entidad y significan costos adicionales por el tiempo que requieren.

Proveedores	Encuesta
ADA	Realizada
Asesoftware	Realizada
Carvajal	Realizada
Choucair	Realizada
Gonet	Realizada
Heinsohn	Realizada
PSL	Realizada
Fatto	Realizada
Leda	Realizada

Figura 3. Proveedores participantes en la fase de diagnóstico. Fuente: Corporación Colombia Digital.

La demanda

En el lado opuesto a la oferta, es decir la demanda, está integrada por las áreas de TI, los CIO, los equipos de desarrollo interno (*In-house)* y las áreas funcionales de las entidades. El tipo de bienes y servicios que requieren las entidades incluyen: asesoría en los procesos para dimensionar los esfuerzos, desarrollo de soluciones a la medida, desarrollo de sistemas funcionales y misionales, servicios de fábricas de *software*, servicios para levantar requerimientos, desarrollo de pruebas, servicios de aseguramiento de la calidad y asesoría general en la gestión de proyectos de desarrollo de *software*.

Al igual que con la oferta, el equipo de estructuración hizo reuniones con varias entidades (ver figura No. 4), con el fin de conocer, desde la óptica de la demanda, cuáles eran las oportunidades de mejora, problemas y buenas prácticas que se identificaban en el proceso de contratación de proyectos de desarrollo de *software*.

Entidad	Encuesta
Colciencias	Realizada
DIAN	Realizada
Ecopetrol	Realizada
ICBF	Realizada
Mintic	Realizada
Presidencia de la República	Realizada
Procuraduría	Realizada
Superintendencia de Servicios Públicos	Realizada
Banco de la República	Realizada
Ministerio de Hacienda y Crédito Público	Realizada

Figura 4. Entidades participantes en la fase de diagnóstico. Fuente: Corporación Colombia Digital.

Los hallazgos de esta fase de diagnóstico fueron los siguientes:

- La calidad de la documentación se debe mejorar para que incluya un nivel de detalle que le permita a la entidad y a los proveedores trabajar y seguir el desarrollo original.
- Se necesita mejorar la comunicación entre las áreas de tecnología y las demás dependencias de una entidad para evitar duplicidad de esfuerzos
- Se requieren profesionales especializados en proyectos de desarrollo de *software* para supervisar y gestionar los contratos de este tipo de conocimiento.
- Es necesario mejorar el estudio e implementación de los requerimientos de integración de los sistemas en desarrollo con las plataformas existentes.
- Se requieren mecanismos que permitan asegurar que los proveedores asignen al proyecto los recursos humanos con los perfiles descritos en la oferta, para evitar demoras y problemas de calidad generados por falta de experiencia o de conocimiento.

- Se precisa tener esquemas flexibles de desarrollo de *software,* que se adapten a las características de este tipo de proyectos y que consideren las condiciones regulatorias que debe cumplir cualquier entidad al ejecutar un proceso de contratación.
- Se deben establecer metodologías que permitan medir la productividad de los proveedores durante el proyecto.
- La dificultad de identificar y evaluar desde el inicio del proyecto todas las variables que deben tenerse en cuenta para el desarrollo, obstaculiza el dimensionamiento y contratación.
- Se observan altos niveles de rotación en el personal asignando por los proveedores, lo que resulta en que el proyecto pierda inercia y requiera esfuerzos adicionales de la entidad.
- Se identifica resistencia al cambio en los usuarios finales de las entidades frente a las iniciativas y desarrollos liderados por el área de TI.
- Los procesos de transferencia de conocimiento contratados carecen de una metodología formal que garantice su eficacia y apropiado desarrollo.
- Se requiere integrar a la administración de los proyectos mecanismos que permitan identificar y gestionar los riesgos.

Después de reconocer los inconvenientes más frecuentes en el proceso de desarrollo de *software*, se construyó un listado del valor que aporta este tipo de proyectos tanto para la demanda como para la oferta. El siguiente listado cubre las ideas destacadas por ambas partes:

- Aseguramiento en las entregas del *software* desarrollado desde etapas tempranas del proyecto, a partir de entregas parciales, por ejemplo de las funcionalidades desarrolladas. Es decir, posibilitar que la entidad saque provecho de las funcionalidades desarrolladas en cada una de las iteraciones.

- Garantiza que el *software* desarrollado cumple con los requerimientos estipulados en términos de calidad, usabilidad, funcionalidad, interoperabilidad y seguridad. Es decir, que la entidad cuente con un software que cumpla sus expectativas y que apalanque el cumplimiento de su misión.
- Garantiza que el proyecto se desarrolla dentro de los márgenes de tiempo y recursos previstos. Es decir, se maximiza el valor por dinero y por lo tanto la entidad hace uso eficiente de recursos limitados.

Con base en la anterior descripción del escenario, se concluye que la problemática en los proyectos de desarrollo de *software* para las entidades puede ser modelada como un sistema de retroalimentación negativa (ver figura No. 5), que inicia con las malas prácticas en la gestión. Como consecuencia, el producto final no cumple con las expectativas de la organización, lo que ocasiona deficiencias en la generación de valor y en el cumplimiento de objetivos de mayor nivel para la organización. La alta visibilidad que tiene el fracaso de los proyectos de desarrollo de *software* provoca desconfianza entre los líderes de las entidades, que se traduce en recortes significativos de los recursos disponibles para este tipo de proyectos. Esta decisión empeora aún más el escenario debido a que las limitaciones en recursos significan un deterioro aún más pronunciado de las prácticas de gestión de proyectos.

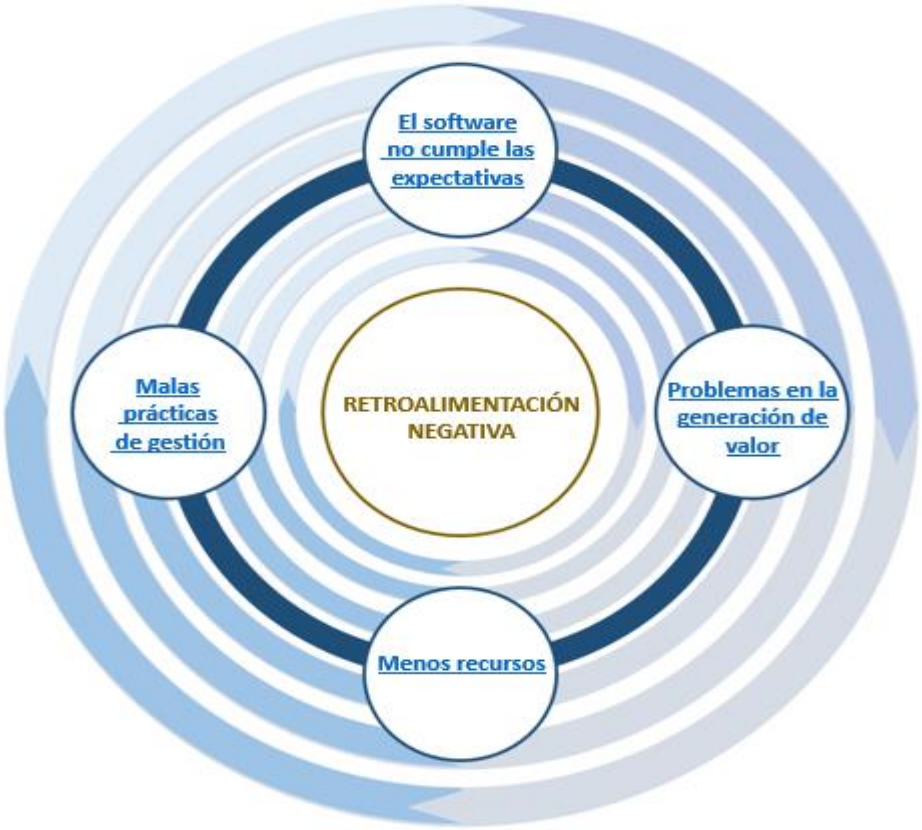


Figura 5. Descripción de la situación actual de desarrollo de software en las entidades. Fuente: Corporación Colombia Digital.

Durante el diagnóstico de la problemática se evidenció que se deben revisar algunos puntos prioritarios: los procesos de planeación y estudios previos para la contratación de desarrollo de sistemas de información, y el levantamiento y la gestión de requerimientos.

Asimismo, el modelo de contratación actual genera retos importantes en dos frentes:

- El uso de las metodologías más eficientes de desarrollo de *software*, que determinan el éxito o fracaso de un proyecto.

- El extenso tiempo del proceso de contratación que puede generar desactualización de la información y pérdida de relevancia del proyecto respecto a los objetivos estratégicos de la entidad.

Finalmente, otro de los principales puntos que debe ser incluido dentro de los aspectos prioritarios para revisar, corresponde a la necesidad de contar con personal experto que pueda detectar sinergias para seleccionar los proyectos de *software*, teniendo como criterio una fuerte base de conocimiento y buenas prácticas para la gestión durante todo el ciclo de vida del proyecto.

2.3 SITUACION ACTUAL

Para 2013 el portafolio de inversión de TI fue de 1,39 billones de pesos aproximadamente, lo que corresponde al 3,3% del presupuesto nacional de inversión. Dichos recursos fueron distribuidos: 49,7% en servicios de TI, 33% en infraestructura y 17,3% en *software*, según datos del Ministerio de Tecnología de la Información y las Comunicaciones (Mintic) y del Departamento Nacional de Planeación (DNP). Dada la restricción del presupuesto para invertir en *software* en el Estado, es de vital importancia usarlo óptimamente, para esto se debe identificar las necesidades prioritarias y mejorar las condiciones de los proyectos para que cumplan con las expectativas planteadas.

La tasa de éxito de los proyectos de *software* ha sido una preocupación constante desde que el Grupo Standish publicó en 1994 el reporte Chaos, en el que indicaba que el 31% de los proyectos de TI eran cancelados o abandonados. Un estudio más reciente (2006) muestra que esta tasa disminuyó al 19%; sin embargo, continúa siendo un porcentaje alto de fracaso (El Emam & Koru, 2008). De acuerdo con el último estudio, las principales causas que determinan el fracaso de los proyectos, en orden de relevancia, son:

- Falta de involucramiento del equipo líder.

- Demasiados requisitos o cambios en el alcance.
- Carencia de habilidades de gestión
- Inconvenientes con el presupuesto
- Deficiencia de habilidades técnicas
- Cambios en las prioridades del negocio
- Dificultades de tiempo
- Fracaso en el uso de una nueva tecnología
- Insuficiencia de personal
- Problemas de calidad
- Falta involucramiento de los usuarios

En conclusión, es importante tener en cuenta que aunque hay espacio de mejora a través del uso de diferentes herramientas y técnicas, todavía se requiere su diseño y robustecimiento para tener tasas de éxito que coincidan con las expectativas de cualquier organización (El Emam & Koru, 2008).

De acuerdo con el análisis de la información recopilada en las entrevistas con las entidades y los proveedores, se ha evidenciado una creciente necesidad del Estado de contar con herramientas que faciliten la contratación de proyectos de desarrollo de sistemas de información. Específicamente, se busca que el proceso de contratación entregue un producto o servicio que cumpla con las expectativas de los interesados, en términos de calidad, bajo un modelo que gestione de forma eficiente los recursos disponibles.

Con el fin de establecer cuáles son las herramientas apropiadas para cada tipo de proyecto de desarrollo de *software*, uno de los aspectos importantes es seleccionar la metodología de desarrollo adecuada según las características expuestas más adelante, en el capítulo de variables y herramientas de evaluación del proyecto. En la actualidad, el mercado ofrece principalmente dos categorías de metodologías: metodologías tradicionales y metodologías

ágiles. Por una parte, las primeras se caracterizan por desarrollar cada una de las etapas que componen el ciclo de vida del proyecto en forma secuencial. En contraste, las metodologías ágiles se desarrollan en forma incremental e iterativa, por lo que el ciclo de vida completo se ejecuta múltiples veces; es decir, todas las etapas se efectúan en un tiempo más corto para entregar una funcionalidad específica y el proceso se repite hasta entregar todas las funcionalidades pactadas.

Otra de las características importantes del enfoque de cada metodología consiste las aproximaciones para gestionar la triple restricción: alcance, recursos y tiempo. Las metodologías tradicionales se caracterizan por estar muy orientadas al plan, lo que se evidencia en que el primer paso es establecer el alcance del proyecto para, posteriormente, determinar cuáles son los recursos y tiempos requeridos para cumplir con el alcance propuesto. Por su parte, las metodologías ágiles abordan el tema desde una orientación hacia generar valor; es decir, establecen cuáles son los recursos y tiempo disponibles para luego pensar qué se puede alcanzar, teniendo en cuenta las restricciones que suponen esas dos variables. La representación gráfica de estos dos enfoques se detalla en la siguiente figura:

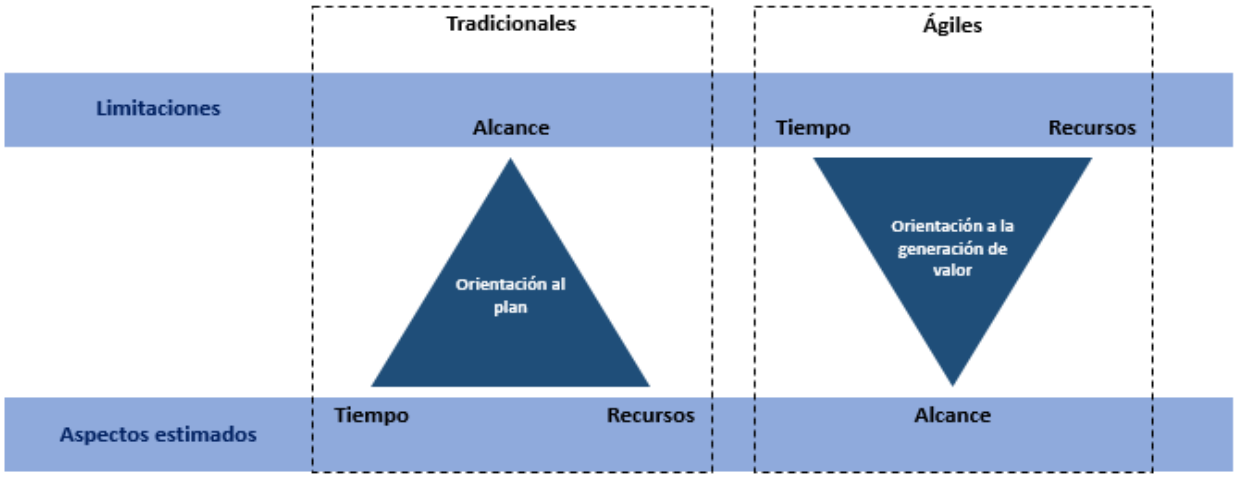


Figura 6. Enfoque de las metodologías de desarrollo de software. Fuente: Corporación Colombia Digital.

La tasa de éxito de los proyectos que usan metodologías ágiles es mayor, como se evidencia en los resultados de la siguiente figura. Sin embargo, es importante tener en cuenta que para que las metodologías ágiles funcionen apropiadamente, el equipo involucrado y la naturaleza del proyecto deben tener características muy particulares.

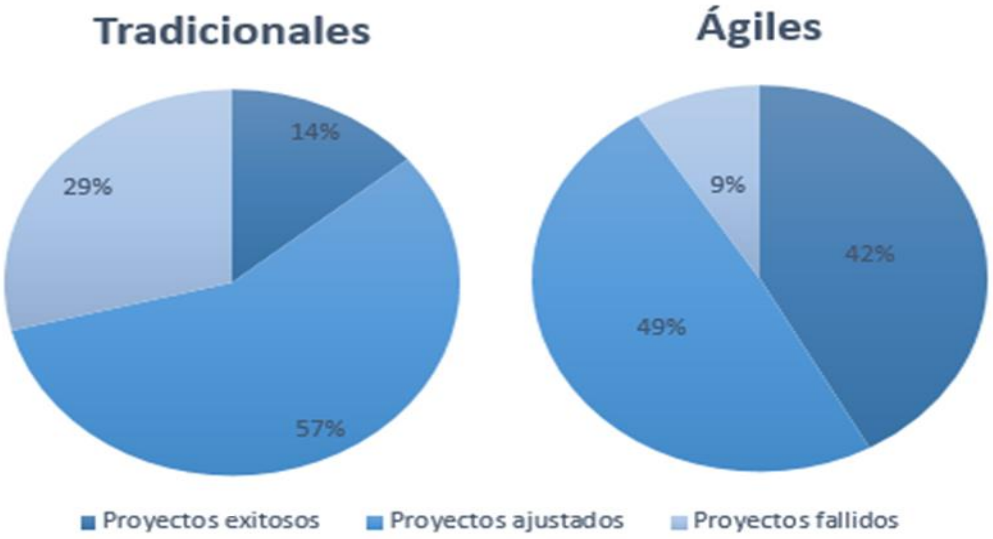


Figura 7. Comparación metodologías tradicionales vs ágiles. (The CHAOS Manifesto – The Standish Group).

Teniendo en cuenta lo anterior, para decidir cuál metodología ofrece mayores probabilidades de éxito, es necesario que cada entidad evalúe las condiciones particulares de cada proyecto, de su entorno cultural y del equipo del proyecto. Dicha evaluación se hace teniendo en cuenta las siguientes variables y herramientas:

2.3.1 Variables y herramientas de evaluación del proyecto

Durante el proceso de investigación se identificaron diez variables que permiten establecer qué tipo de metodología es más apropiada. La siguiente gráfica muestra dichas variables. De acuerdo con su resultado lo apropiado es ubicarse en la franja gris que se muestra en la gráfica.

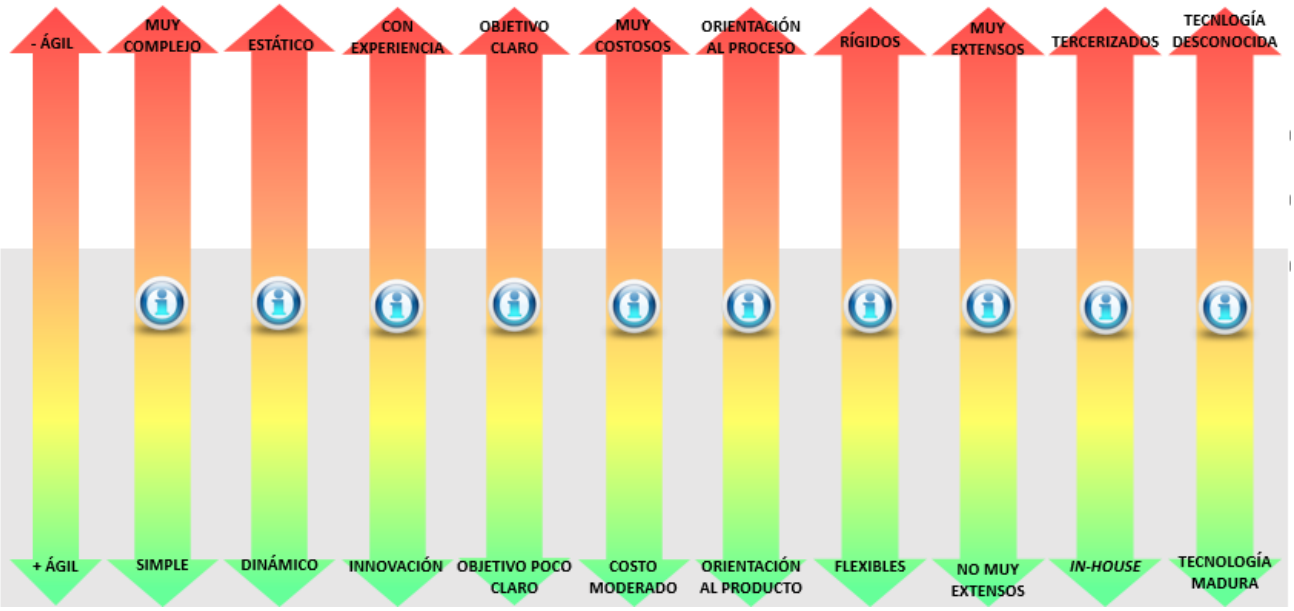


Figura 8. Variables del proyecto que determinan idoneidad de la metodología. Fuente: Corporación Colombia Digital.

A continuación también se explican las herramientas para evaluar cada una de las variables.

2.3.1.1 Evaluación de la complejidad del proyecto.

Una de las formas más rápidas para evaluar la complejidad de un proyecto consiste en dimensionar el esfuerzo que requiere, al entender la solución que propone para resolver un problema o cubrir una necesidad. Esa evaluación puede ser calificada en tres niveles: bajo, medio y alto, y debe ser hecha por un experto en el proceso o área de negocio a la que corresponde el proyecto. De igual manera, el experto debe evaluar el nivel de incertidumbre, es decir, debe determinar qué tan fácil es predecir desde un comienzo todos los aspectos y riesgos relevantes, de tal modo que no queden por fuera elementos críticos que después impliquen cambios importantes en el alcance, recursos y tiempo planeados. La calificación de la capacidad para predecir se hace de forma cualitativa, desde un nivel predecible hasta un nivel caótico, la siguiente gráfica muestra la matriz para evaluar dicha complejidad:

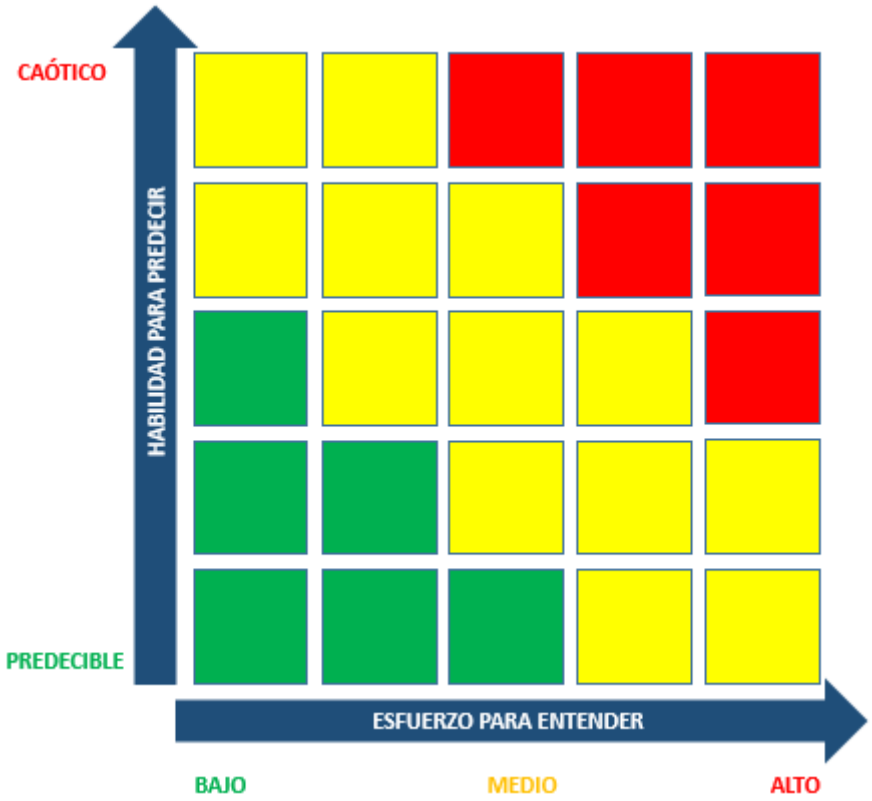


Figura 9. Evaluación rápida de la complejidad de un proyecto. Fuente: Corporación Colombia Digital.

Es importante resaltar que la evaluación rápida deja a un lado una serie de parámetros que hacen más dispendiosa la evaluación de la complejidad de un proyecto y que el costo que tiene esta simplificación puede ser que no se dimensione apropiadamente la complejidad. Para los casos en que se disponga de tiempo y recursos para hacer una evaluación más rigurosa, se sugiere que un experto evalúe cada una de las siguientes dimensiones:

Duración.

Dimensión	Complejidad baja	Complejidad moderada	Complejidad alta
Tiempo	< de 3 meses	Entre 3 y 6 meses	> De 6 meses

Costo.

Dimensión	Complejidad baja	Complejidad moderada	Complejidad alta
Costo (Entidades grandes)	< 200 millones de COP	Entre 200 y 800 millones de COP	> De 800 millones de COP
Costo (Entidades pequeñas)	< 100 millones de COP	Entre 100 y 300 millones de COP	> De 300 millones de COP

Tamaño del equipo.

Dimensión	Complejidad baja	Complejidad moderada	Complejidad alta
Tamaño del equipo	Entre 3 y 4 personas	Entre 5 y 10 personas	Más de 10 personas

Composición del equipo.

Dimensión	Complejidad baja	Complejidad moderada	Complejidad alta
Composición del equipo	Todo el equipo ha trabajado en grupo en oportunidades anteriores.	▪El equipo está compuesto por personas internas y externas a la organización. ▪Parte del equipo ha trabajado en grupo en oportunidades anteriores.	▪El equipo es altamente heterogéneo: equipos tercerizados + equipos virtuales + equipos con diferencias culturales, entre otros. ▪El equipo no ha trabajado en grupo antes.

Desempeño del equipo.

Dimensión	Complejidad baja	Complejidad moderada	Complejidad alta
Desempeño del equipo	▪El equipo cuenta con fuertes habilidades de liderazgo de proyectos en el tema de interés. ▪El proyecto cuenta con procedimientos y	▪El equipo cuenta con suficientes competencias en el liderazgo de proyectos del tema de interés.	▪El equipo no cuenta con experiencia suficiente en el liderazgo de proyectos del tema de interés.

	metodologías previamente definidas y probadas en el área de conocimiento del proyecto.	▪Se conoce a través de indicadores el desempeño general del equipo. ▪Existen procedimientos de aseguramiento de la calidad previamente definidos.	▪Existen metodologías de trabajo diversas y conflictivas. ▪No se conoce el desempeño del equipo.

Urgencia e importancia.

Dimensión	Complejidad baja	Complejidad moderada	Complejidad alta
Urgencia e importancia	▪El proyecto no es urgente ni tiene alta importancia para la organización. ▪El proyecto se ejecuta después de estudiar cuidadosamente su viabilidad y realizar todo el proceso de planeación. ▪El proyecto no tiene como objetivo solucionar un problema que esté generando consecuencias negativas para la organización.	▪El proyecto tiene: alta urgencia y baja importancia o baja urgencia y alta importancia. ▪El proyecto puede planearse de forma aceptable. ▪En las situaciones en que el equipo de proyecto debe improvisar el impacto de una mala decisión es bajo.	▪El proyecto tiene alta importancia y urgencia para el negocio. ▪El proyecto debe ejecutarse inmediatamente y hay fuertes restricciones de tiempo para completar adecuadamente el proceso de planeación. ▪El proyecto tiene como objetivo solucionar un problema que esta generado costos adicionales, perjuicios legales, desastres ambientales, etc.

Flexibilidad de la triple restricción: alcance, recursos y tiempo.

Dimensión	Complejidad baja	Complejidad moderada	Complejidad alta
Flexibilidad de la triple restricción: Alcance, costo y tiempo.	▪El proyecto tiene pocos hitos. ▪Existe alta flexibilidad para modificar el presupuesto. ▪El alcance del proyecto es moderado según el juicio de expertos.	▪El presupuesto puede variar aproximadamente un 10% pero los tiempos no son negociables. ▪Según el juicio de expertos el alcance planteado y los hitos se puede cumplir sin mayores inconvenientes.	▪El alcance del proyecto es ambicioso. ▪El cronograma de trabajo es ambicioso. ▪La fecha límite es difícil de cumplir y no se puede modificar. ▪No existe flexibilidad en cuanto a presupuesto, alcance, tiempo y calidad.

Claridad en el problema o la oportunidad.

Dimensión	Complejidad baja	Complejidad moderada	Complejidad alta
Claridad del problema o la oportunidad	▪Los objetivos de la organización son claros. ▪La oportunidad o problema están completamente definidos, han sido apropiadamente comunicados a todas las partes interesadas y se articula con la estrategia de la organización.	▪Los objetivos de la organización están formalmente definidos pero hay oportunidades de mejora en su comunicación. ▪El problema u oportunidad está siendo definido o está en proceso de aclaración.	▪Los objetivos de negocio son confusos o no se ha definido formalmente la relación del proyecto con la estrategia de la organización. ▪El problema que intenta resolver el proyecto no está completamente claro. ▪La oportunidad que el proyecto busca aprovechar no ha sido claramente identificada.

Importancia estratégica.

Dimensión	Complejidad baja	Complejidad moderada	Complejidad alta
Importancia estratégica	▪ Hay apoyo visible de la alta gerencia. ▪ No tiene implicaciones políticas. ▪ Hay una comunicación directa y fluida.	▪ Cuenta con adecuado apoyo de la alta gerencia. ▪ Tiene algún impacto en la misión de la organización. ▪ Tiene mínimas implicaciones políticas. ▪ Hay 2 o 3 grupos de interesados. ▪ La comunicación supone algunos retos y esfuerzos de coordinación.	▪ Tiene soporte insuficiente o inexistente de la alta gerencia. ▪ Afecta directamente la misión de la organización. ▪ Tiene significativas implicaciones políticas. ▪ Es un proyecto visible para los niveles más altos de la organización. ▪ Hay múltiples grupos de interesados. ▪ Hay intereses en conflicto alrededor del proyecto.

Nivel de impacto dentro de la organización.

Dimensión	Complejidad baja	Complejidad moderada	Complejidad alta
Nivel de impacto dentro de la organización	▪Impacta solo un área, unidad de negocio, proceso o sistema.	▪Impacta dos o tres áreas, unidades de negocio, procesos o sistemas.	▪ Significa un cambio a gran escala, que impacta a toda la organización. ▪ Incluye proyectos que transforman o desplazan a una organización.

			▪ Impacta varias áreas, unidades de negocio, procesos o sistemas.

Riesgos, limitaciones y dependencias.

Dimensión	Complejidad baja	Complejidad moderada	Complejidad alta
Riesgos, limitaciones y dependencias.	▪ Riesgos de bajo impacto y probabilidad ▪ Grado mínimo de dependencia de factores externos ▪ No existen desafíos significativos en término de integración ▪ No hay requerimientos de carácter regulatorio que sean desconocidos ▪ No representa exposición punitiva	▪ Riesgos de mediano impacto y probabilidad ▪ Algunos de los objetivos dependen de factores externos ▪ Existen algunos desafíos de integración ▪ Hay algunos nuevos aspectos regulatorios ▪ Se presenta una exposición punitiva aceptable	▪ Riesgos de alto impacto y probabilidad ▪ El éxito del proyecto depende en gran medida de factores externos ▪ Se requiere alta integración ▪ Es un sector nuevo en el que se desconoce la regulación ▪ Hay un alto nivel de exposición punitiva

Una vez se tenga la evaluación del proyecto, se retoma la matriz de la figura 8 del numeral 2.3.1 Variables y herramientas de evaluación del proyecto, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del proyecto de la siguiente manera:

- Si el resultado de complejidad del proyecto es bajo, en la matriz se ubica en el rango de simple, el cual se identifica con el color verde

- Si el resultado de complejidad del proyecto es medio, en la matriz se ubica en la mitad del rango entre simple y muy complejo, el cual se identifica con los colores naranja y amarillo
- Si el resultado de complejidad del proyecto es alto, en la matriz se ubica en el rango de muy complejo, el cual se identifica con el color rojo

2.3.1.2 Evaluación del nivel de estabilidad de un proyecto.

Todo proyecto tiene unos niveles de estabilidad durante su desarrollo; la evaluación de esta variable, desde etapas tempranas, permite predecir de forma cualitativa el volumen de cambios esperados y, por lo tanto, la metodología que mejor se ajusta a un entorno con un mayor o menor grado de incertidumbre. Para evaluar el nivel de estabilidad de un proyecto se sugiere revisar las siguientes dimensiones y determinar si el proyecto es estático, semi estático o dinámico:

Resultado esperado.

Dimensión	Proyectos estáticos	Proyectos semi estáticos	Proyectos dinámicos
Resultado esperado	Se espera el mismo resultado durante todo el proyecto.	Se esperan reformas menores al resultado esperado.	Se esperan transformaciones profundas al resultado esperado.

Propósito.

Dimensión	Proyectos estáticos	Proyectos semi-estáticos	Proyectos dinámicos
Propósito	Se espera generar mejoras de desempeño del sistema en desarrollo.	Se espera responder a necesidades adicionales de los interesados o suplir deficiencias encontradas durante las fases de desarrollo y pruebas.	Se espera resolver problemas de desarrollo desde una perspectiva del sistema como un todo.

			Se espera atender cambios en las necesidades del negocio.

Participación.

Dimensión	Proyectos estáticos	Proyectos semi-estáticos	Proyectos dinámicos
Participación	Se espera que la mayoría de los cambios puedan ser gestionados por el grupo respetando los procedimientos establecidos.	Se espera que la mayoría de los cambios puedan ser gestionados involucrando a los interesados relevantes para resolver el problema.	Se espera que la mayoría de los cambios puedan ser gestionados involucrando las máximas autoridades dentro del proyecto y la organización.

Proceso.

Dimensión	Proyectos estáticos	Proyectos semi-estáticos	Proyectos dinámicos
Proceso	▪ Se espera que se mantengan las mismas actividades y procedimientos. ▪ Se espera que la estructura continúe siendo igual. ▪ Se espera que las relaciones, roles y responsabilidades de cada individuo dentro del proyecto continúen siendo las mismas.	▪ Se espera que los procedimientos y actividades deban ser revisados para acoger los cambios propuestos. ▪ Se espera que la estructura se ajuste para acoger cambios propuestos. ▪ Se espera que las relaciones, roles y responsabilidades de cada	▪ Se espera que los procedimientos y actividades deban ser rediseñados para ajustarse a la nueva orientación del proyecto. ▪ Se espera que la estructura deba ser definida nuevamente. ▪ Se espera que las relaciones, roles y responsabilidades de cada individuo se definan desde

		individuo se ajusten para articular cambios.	cero para cumplir con nuevos objetivos.

Una vez se tenga la evaluación del nivel de estabilidad del proyecto, se retoma la matriz de la figura 8 del numeral 2.3.1 Variables y herramientas de evaluación del proyecto, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del proyecto de la siguiente manera:

- Si el resultado del proyecto es dinámico, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado del proyecto es semi-estático, en la matriz se ubica en la mitad del rango entre dinámico y estático, el cual se identifica con los colores naranja y amarillo
- Si el resultado del proyecto es estático, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.1.3 Evaluación de proyectos para determinar si predomina la innovación o la experiencia.

De acuerdo con la naturaleza misma del proyecto existen objetivos que deben alcanzarse por medio de soluciones innovadoras o basadas en la experiencia y el conocimiento previo. Los siguientes son listados de las características más relevantes que tienen un proyecto con un enfoque innovador y basado en la experiencia. Utilice la lista para determinar la orientación de su proyecto.

Proyecto en el que predomina la innovación:

- ☑ El proyecto inicia conociendo una necesidad u oportunidad, pero se requiere un proceso de investigación y desarrollo para conocer el entregable.
- ☑ Las metas y objetivos se definen durante el desarrollo.
- ☑ Hay un alto nivel de incertidumbre.
- ☑ El desarrollo se hace a través de la investigación y la generación de ideas creativas.
- ☑ El valor en el proyecto se obtiene reconociendo que las condiciones cambian con el tiempo y por lo tanto se requieren nuevas soluciones. Los errores son una oportunidad de aprendizaje.
- ☑ El equipo se concentra en maximizar los beneficios a partir de ciclos repetitivos de aprendizaje, creación de prototipos, evaluación y mejoramiento continuo.

Proyecto en el que predomina la experiencia.

- ☑ El proyecto inicia conociendo el producto o servicio que debe ser entregado como resultado.
- ☑ Las metas y objetivos son claras desde el inicio.
- ☑ Hay un bajo nivel de incertidumbre.
- ☑ El proceso de desarrollo se hace bajo esquemas de imitación o ingeniería inversa, que añaden poco valor y requieren poco aprendizaje.
- ☑ El valor en el proyecto se apalanca en la experiencia que se adquiere durante el aprendizaje y se busca no repetir errores del pasado.
- ☑ El equipo se concentra en definir los hitos y tiempos para articular la entrega oportuna del resultado.

Una vez se tenga la evaluación de proyectos para determinar si predomina la innovación o la experiencia, se retoma la matriz de la figura 8 del numeral 2.3.1 Variables y herramientas de evaluación del proyecto, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del proyecto de la siguiente manera:

- Si el resultado determina que predomina la innovación, en la matriz se ubica en el rango con este mismo nombre, el cual se identifica con el color verde

- Si el resultado determina que es una combinación entre innovación y experiencia , en la matriz se ubica en la mitad del rango, el cual se identifica con los colores naranja y amarillo
- Si el resultado determina que predomina la experiencia, en la matriz se ubica en el rango con este mismo nombre, el cual se identifica con el color rojo

2.3.1.4 Evaluación de la claridad de los objetivos.

Para establecer la claridad de los objetivos del proyecto, el primer paso es evaluar si dichos objetivos están alineados con las metas de más alto nivel de la entidad y si esa alineación es comprendida por todos los interesados. La siguiente figura ilustra los diversos tipos de objetivos que pueden existir en una misma entidad y cómo los de menor nivel permiten articular los objetivos de escalas más altas.

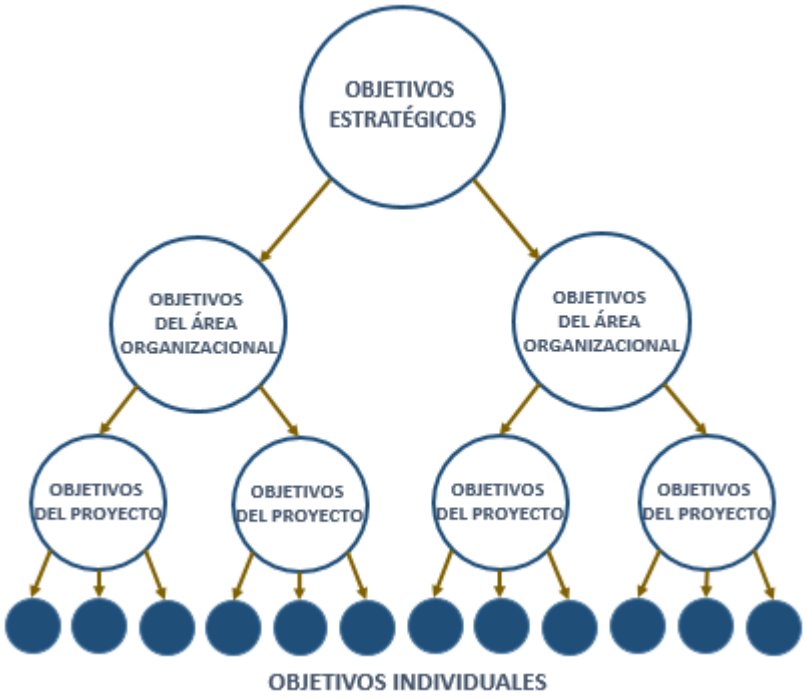


Figura 10. Alineación de los diferentes niveles de objetivos dentro de una organización. Fuente: Goal Setting– Harvard Business Review.

Una vez se ha establecido la relación entre los objetivos de proyecto y los objetivos estratégicos de la entidad, se procede a verificar si los interesados tienen claros cuáles son los objetivos críticos, habilitantes y deseados, que se entienden como:

- Objetivos críticos:

Son las metas cruciales que el proyecto debe cumplir para ser calificado como exitoso. Se identifican con facilidad en los proyectos con objetivos claros.

- Objetivos habilitantes:

Son las metas o condiciones deseadas que facilitan la normal evolución del desarrollo planeado. Se identifican con facilidad en los proyectos con objetivos claros.

- Objetivos deseados:

Son las metas o condiciones que permiten que el proyecto se desarrolle de forma más fácil y rápida. Se identifican con facilidad en los proyectos con objetivos claros.

Seguido a esta identificación, se debe verificar que los objetivos planteados cumplan con cinco características que favorecen su claridad y comprensión entre los interesados:

- El objetivo es específico:

Contiene detalles suficientes para que pueda ser comprendido por cualquiera de los interesados.

- El objetivo es medible:

Puede ser medido de forma cuantitativa o cualitativa para verificar su cumplimiento.

- El objetivo es lograble:

Puede ser alcanzado según el criterio de expertos en desarrollo de *software* y en el área de negocio a la que pertenece el objetivo.

- El objetivo es realista:

Tiene en cuenta las limitaciones de tiempo y recursos disponibles.

- El objetivo es limitado en el tiempo:

Especifica un intervalo de tiempo en el que debe ser cumplido.

Para finalizar el proceso de evaluación de la claridad de los objetivos, presentamos un listado con las características más relevantes que tiene un proyecto, en el que los objetivos son claros desde el inicio. Utilice la lista para complementar la evaluación que se ha hecho hasta el momento.

- Características de un proyecto que tiene claros los objetivos desde el inicio:

☑ El alcance es definido y no se modifica desde el inicio.
☑ Se mantiene el alcance ajustando el tiempo y los recursos.
☑ Se planean todas las características del desarrollo de *software*.
☑ Se cuenta con descripciones muy detalladas de los requerimientos.

- Características de un proyecto con objetivos inciertos al inicio:

☑ Se definen el tiempo y los recursos desde un inicio. El alcance se ajusta durante el desarrollo.
☑ Se mantienen fijos el tiempo y los recursos para que durante el proyecto se definan los objetivos por cumplir.
☑ Se planea y diseña una característica inicial del desarrollo de *software* y durante el proyecto se descubren las siguientes características.

Una vez se tenga la evaluación de la claridad de los objetivos, se retoma la matriz de la figura 8 del numeral 2.3.1 Variables y herramientas de evaluación del proyecto, para ubicar

el resultado que permite establecer el tipo de metodología que favorece el desarrollo del proyecto de la siguiente manera:

- Si el resultado indica que los objetivos son poco claros, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que no todos los objetivos son claros, en la matriz se ubica en la mitad del rango entre objetivos claros y poco claros, el cual se identifica con los colores naranja y amarillo
- Si el resultado indica que los objetivos son claros, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.1.5 Evaluación del nivel de costo de los proyectos.

El nivel de costo de los proyectos se determina de acuerdo a la siguiente tabla.

Dimensión	Bajo costo	Costo moderado	Alto costo
Costo (Presupuestos grandes)	< 200 millones de COP	Entre 200 y 800 millones de COP	> 800 millones de COP
Costo (Presupuestos pequeños)	< 100 millones de COP	Entre 100 y 300 millones de COP	> 300 millones de COP

Tabla 1. Nivel de costo de los proyectos. Fuente: Corporación Colombia Digital.

Una vez se tenga el nivel de costo del proyecto, se retoma la matriz de la figura 8 del numeral 2.3.1 Variables y herramientas de evaluación del proyecto, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del proyecto de la siguiente manera:

- Si el resultado indica que es un presupuesto pequeño, en la matriz se ubica en el rango de costo moderado, el cual se identifica con el color verde
- Si el resultado indica que es un presupuesto grande, en la matriz se ubica en el rango de muy costoso, el cual se identifica con el color rojo

2.3.1.6 Evaluación de proyectos en los que predomina la orientación al producto o al proceso.

Los proyectos pueden tener dos tipos de orientaciones que determinan la metodología de trabajo requerida. En el primer tipo la prioridad es el producto y por lo tanto las variables relevantes para tomar las decisiones son: el valor, el negocio, el cliente y el mercado. Este tipo de proyectos tiene un enfoque en las características y funcionalidades del desarrollo de *software*, todos los esfuerzos van dirigidos a cumplir principalmente con las expectativas y el estado y avance se mide respecto al cumplimiento de los requerimientos. La siguiente figura muestra las variables que enmarcan un proyecto orientado al producto:

Figura 11. Restricciones en el modelo de gestión del proyecto orientado al producto. Fuente: Differences between product and project management – Hector del Castillo.

En contraste con el tipo anterior, en los proyectos orientados al proceso se observa que las decisiones están enmarcadas por las variables: calidad, alcance, tiempo y recursos. Es decir, el enfoque en este caso va dirigido a gestionar las actividades que deben ser ejecutadas para crear el desarrollo de *software* y el avance y estado se miden respecto al cumplimiento de dichas actividades según el plan establecido. La siguiente figura muestra las variables que enmarcan un proyecto orientado al proceso.

Figura 12. Restricciones en el modelo de gestión del proyecto orientado al proceso. Fuente: Differences between product and project management – Hector del Castillo.

Una vez se tenga la evaluación del proyecto para determinar si predomina la orientación al producto o al proceso, se retoma la matriz de la figura 8 del numeral 2.3.1 Variables y herramientas de evaluación del proyecto, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del proyecto de la siguiente manera:

- Si el resultado determina que predomina la orientación al producto, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado determina que predomina la orientación al proceso, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.1.7 Evaluación de la flexibilidad de los proyectos.

La flexibilidad mide qué tan fácil se puede adaptar un proyecto a cambios de las condiciones durante todo el ciclo de vida. Dependiendo de la naturaleza del proyecto pueden existir diferentes niveles de adaptabilidad según los siguientes aspectos:

- Cambios en el alcance:
  Este tipo de proyectos se caracteriza por contar con el apoyo y seguimiento cercano del equipo de liderazgo de la entidad. Desde el principio los interesados son conscientes que se pueden producir eventuales cambios y por eso tienen mecanismos

preestablecidos para revisar y ajustar el alcance inicialmente planteado. Los procesos de gestión de cambio son rápidos y sencillos.

- Flexibilidad en los recursos:

  Se da cuando el proyecto es crítico para la organización y existe poca flexibilidad en el tiempo disponible o en la acotación del alcance inicialmente pactado, en ambos casos la organización cuenta con recursos adicionales y con la voluntad de invertirlos para sacar adelante el proyecto. Para esto, se preestablecen mecanismos de gestión de cambios rápidos y simples, para revisar y ajustar los recursos asignados.

- Flexibilidad en el tiempo:

  Estos proyectos tienen un nivel bajo de criticidad para la organización, por lo que pueden ser postergados sin afectar los planes de la entidad. Son proyectos que normalmente no son flexibles en términos del alcance y cuentan con mecanismos preestablecidos de gestión de cambios rápidos y simples para revisar y ajustar los tiempos inicialmente pactados.

- Flexibilidad en el nivel de calidad:

  Por lo general este no es un aspecto que sea flexible en los proyectos ya que los requisitos de calidad se establecen bajo condiciones mínimas que no pueden modificarse. Sin embargo, en contadas excepciones se encuentran proyectos en los que el cliente está dispuesto a sacrificar algún aspecto poco crítico de la calidad del producto para tener una entrega oportuna o dentro del rango de recursos disponibles.

- Flexibilidad en las regulaciones:

  La leyes, normas y regulaciones que debe tener en cuenta un proyecto determinan un marco que limita los tipos de soluciones que pueden tenerse en cuenta para resolver un problema o suplir una necesidad. La flexibilidad del proyecto es delimitada por las normas que le aplican.

- Flexibilidad en la tecnología:

  Los proyectos que son flexibles frente a la tecnología tienen libertad para evaluar y seleccionar diferentes tipos de soluciones gracias a que cuentan con requerimientos

técnicos elásticos y a que el mercado ofrece un amplio portafolio de opciones en el que se puede evaluar la relación costo-beneficio de cada alternativa.

- Flexibilidad en los procesos:
  Los procesos que afectan el proyecto son un factor crítico de flexibilidad porque son una fuente importante de requerimientos y restricciones. Un proceso que permite cambios amplía la gama de opciones para proponer una solución.
- Flexibilidad en las personas:
  La capacidad de adaptación al cambio de los grupos interesados incide en las decisiones que toman sobre el proyecto. Normalmente, todos los procesos de gestión de cambios requieren la aprobación de un grupo de personas que tienen asignada esta responsabilidad. El nivel personal de flexibilidad de este grupo de personas incide directamente en el nivel de flexibilidad del proyecto.

Es importante tener en cuenta que la flexibilidad del proyecto es diferente a la estabilidad del proyecto. En la evaluación de estabilidad se midió el volumen de cambios esperados, mientras que con la evaluación de flexibilidad lo que se busca es establecer qué tan fácil se puede adaptar un proyecto a los cambios en las diferentes etapas del ciclo de vida. La siguiente gráfica ilustra en verde un proyecto que se mantiene flexible a lo largo del ciclo de vida, es decir, que se adapta al cambio fácilmente incluso en las etapas finales; en contraste, la línea roja muestra un proyecto que es flexible en las etapas iniciales pero que pierde flexibilidad con el avance del proyecto.

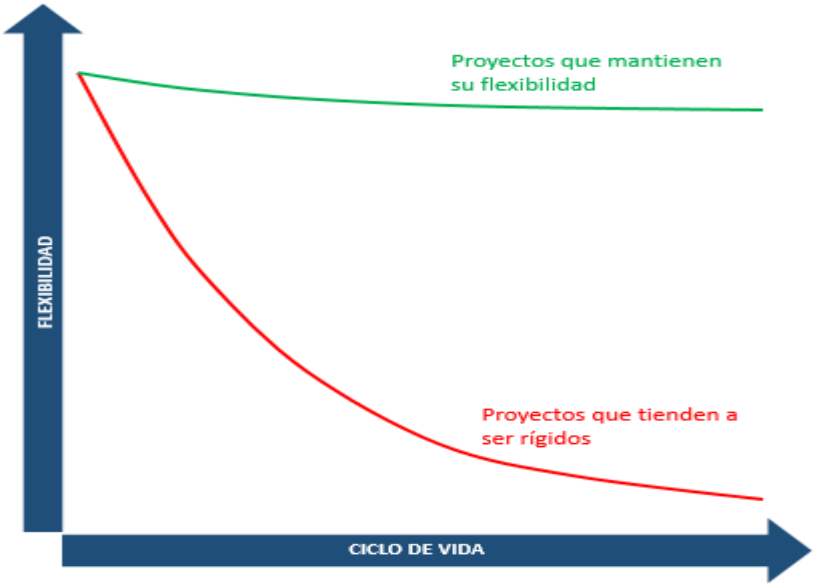


Figura 13. Flexibilidad de los proyectos a lo largo del ciclo de vida. Fuente: Agile benefits– VersionOne.

Una vez se tenga la evaluación de la flexibilidad de los proyectos, se retoma la matriz de la figura 8 del numeral 2.3.1 Variables y herramientas de evaluación del proyecto, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del proyecto de la siguiente manera:

- Si el resultado indica que el proyecto es flexible, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que el proyecto se encuentra en flexible y rígido, en la matriz se ubica en la mitad del rango, el cual se identifica con los colores naranja y amarillo
- Si el resultado indica que el proyecto es rígido, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.1.8 Consideraciones para determinar la extensión en tiempo de un proyecto.

La evaluación de la extensión de un proyecto busca determinar los márgenes de tiempo que se manejan. Para este examen se establecieron seis dimensiones que permiten determinar

si un proyecto es extenso o poco extenso. La siguiente tabla presenta las dimensiones y los rangos de evaluación sugeridos para cada caso:

Dimensiones	Extenso	Poco extenso
Horizonte de planeación	Largo y mediano plazo: > 1 año	Corto plazo: < 1 año
Distancia temporal entre el usuario final y el desarrollador	Más de una hora.	Menos de una hora.
Tiempos entre la especificación y la implementación.	Mayor a dos semanas.	Menor a dos semanas.
Tiempo para detectar problemas	Mayor a ocho semanas.	Menor a ocho semanas.
Riegos asociados al cronograma del proyecto	Riesgos gestionados de alta probabilidad e impacto.	Riesgos gestionados de baja probabilidad e impacto.
Rapidez para responder al cambio	Inferior a dos semanas	Superior a dos semanas.

Tabla 2. Dimensiones que permiten establecer si un proyecto es extenso o poco extenso. Fuente: Agile vs Waterfall Project Management– Venveo.

Una vez se tenga la evaluación para determinar la extensión en tiempo, se retoma la matriz de la figura 8 del numeral 2.3.1 Variables y herramientas de evaluación del proyecto, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del proyecto de la siguiente manera:

- Si el resultado indica que no es muy extenso, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que es muy extenso, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.1.9 Proyectos In-House vs proyectos tercerizados.

La decisión de desarrollar los proyectos dentro de la entidad (*in*-hose) o a través de los servicios de un proveedor (tercerizar) tiene implicaciones importantes en la elección de la

metodología que más se ajusta a la dinámica de trabajo. Las variables que determinan si el desarrollo será *in-house* o tercerizado son principalmente: el nivel de criticidad y el nivel de competencia de la entidad en el tema. La matriz que se muestra a continuación ilustra los casos en que resulta conveniente cada tipo de desarrollo:

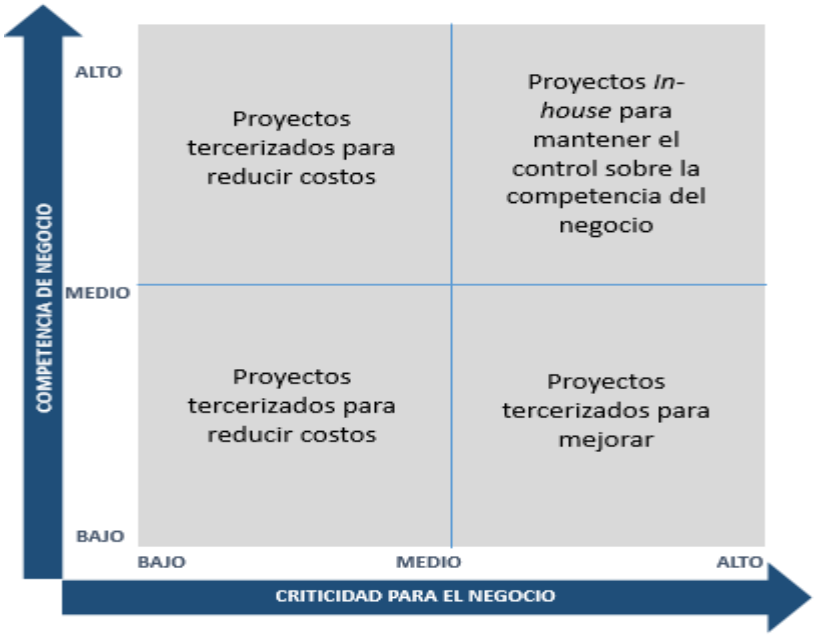


Figura 14. Variables que determinan que un proyecto sea tercerizado. (Proposed Outsourcing Matrix – Hunger y Wheelen).

Existen otras seis variables que también influyen sobre el proceso de tercerización de un proyecto:

Calidad.

Variable	In-house	Tercerizado
Calidad	Se tiene control sobre las iniciativas de calidad que se desean implementar en el proyecto.	El proveedor está especializado en el tipo de proyecto y por lo tanto cuenta con iniciativas de calidad mucho más robustas.

	Se pueden ejecutar con facilidad procesos de trazabilidad sobre los problemas detectados.	

Costo.

Variable	In-house	Tercerizado
Costo	Es difícil lograr economías de escala en proyectos In-house.	Se generan economías de escala ya que el proveedor gestiona y ejecuta varios proyectos similares. El proveedor tiene incentivos para disminuir los costos y mejorar sus márgenes de ganancia.

Velocidad.

Variable	In-house	Tercerizado
Velocidad	Se pueden programar los tiempos de las actividades del proyecto de acuerdo a las necesidades del negocio.	Los tiempos quedan estipulados en el contrato y su incumplimiento implica penalidades para el proveedor.

Flexibilidad.

Variable	In-house	Tercerizado
Flexibilidad	Este tipo de proyecto se ajusta fácilmente a realidad y necesidades específicas del negocio. Pueden presentarse limitaciones de capacidad debido a la disponibilidad limitada de los recursos.	Se tiene mayor flexibilidad y capacidad para responder a eventos inesperados. La demanda de recursos adicionales significa retos para balancear los requerimientos de todos los clientes.

Experiencia.

Variable	In-house	Tercerizado

Experiencia	La experiencia en el proyecto puede ser reducida y no tener el grado de especialización que el proyecto demanda.	Ya que el foco del negocio está en una clase específica de proyectos se tiene más experiencia y recursos altamente especializados.

Comunicación.

Variable	In-house	Tercerizado
Comunicación	La comunicación fluye fácilmente y no significa costos adicionales para la organización.	La comunicación es más compleja, se deben formalizar los medios y momentos de comunicación y se generan costos adicionales.

Una vez se tenga la evaluación si el proyecto lo desarrolla In-House o a través de un tercero, se retoma la matriz de la figura 8 del numeral 2.3.1 Variables y herramientas de evaluación del proyecto, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del proyecto de la siguiente manera:

- Si el resultado indica que es In-House, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que es tercerizado, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.1.10 Evaluación de madurez de la tecnología.

La madurez de la tecnología especifica el tipo de riesgos a los que se enfrentará un proyecto y por lo tanto debe considerarse en la dinámica de trabajo. Esta madurez puede ser evaluada a partir de los parámetros: evolución, sostenibilidad y obsolescencia tecnológica. La siguiente tabla describe los criterios para determinar el nivel de cada uno de los tres parámetros:

Parámetro	Muy bajo	Bajo	Medio	Alto	Muy alto

Evolución de la tecnología	Tecnología probada y ampliamente usada en la industria.	Tecnología probada para la aplicación actual y lista para su adopción masiva.	Tecnología probada en proyectos pilotos y lista para ser adoptada en proyectos de producción.	Tecnología lista para ser usada como piloto.	Tecnología experimental.
Sostenibilidad de la tecnología	Dentro del ciclo de vida del producto la tecnología está ubicada en la fase de estabilización.	Dentro del ciclo de vida del producto la tecnología está ubicada al inicio o al final de la fase de estabilización.	Dentro del ciclo de vida del producto la tecnología está ubicada al final de la fase de desarrollo de producto o al comienzo de la fase de declinación del producto.	Dentro del ciclo de vida del producto la tecnología está ubicada al inicio de la fase de desarrollo de producto o al final de la fase de declinación del producto.	Dentro del ciclo de vida del producto la tecnología se ubica en la fase de desarrollo o de obsolescencia.
Obsolescencia	Se cuenta con tecnología que corresponde al estado del arte de la industria. La obsolescencia no es un problema en el momento.	Se cuenta con tecnología que corresponde al estado del arte de la industria. La obsolescencia no es un problema en el momento.	Se cuenta con tecnología vigente pero existen desarrollos en curso que remplazarán la tecnología en el futuro.	Se cuenta con tecnología vigente pero existen desarrollos que remplazarán la tecnología en el corto plazo.	La tecnología ya no es vigente y no debe ser usada en nuevos proyectos. El soporte es escaso.

Tabla 3. Criterios de evaluación de madurez de la tecnología. Fuente: An Approach to Technology Risk Management – Ricardo Valerdi – Ron Khol.

En particular, para determinar el nivel de riesgo que significa la sostenibilidad tecnológica es útil tener en cuenta el ciclo de vida de un producto tecnológico, para facilitarlo lo ilustramos en la siguiente figura. El eje X muestra una flecha que cambia de color de acuerdo con el nivel de riesgo de incertidumbre tecnológica para cada etapa: rojo representa riesgo

de alto impacto, amarillo es riesgo moderado y verde riesgo bajo. El eje Y muestra la sostenibilidad de la tecnología en aspectos como desarrollo de la tecnología, desarrollo del producto, estabilización del producto, declinación del producto y obsolescencia.

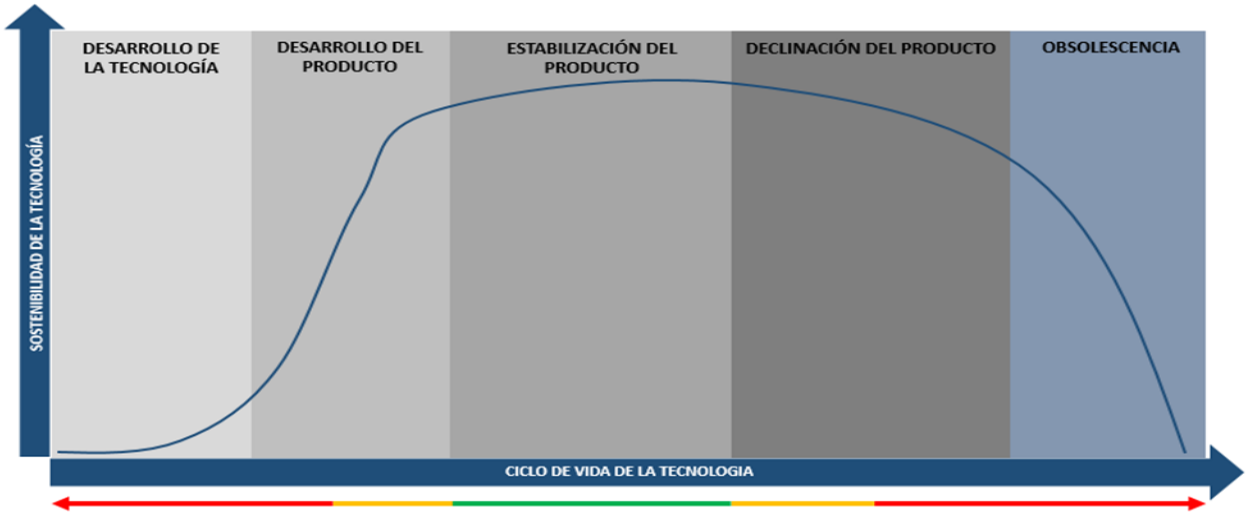


Figura 15. Nivel de sostenibilidad de la tecnología según la el ciclo del vida del producto tecnológico. Fuente: An Approach to Technology Risk Management – Ricardo Valerdi – Ron Khol.

Una vez se tenga la evaluación de madurez de la tecnología, se retoma la matriz de la figura 8 del numeral 2.3.1 Variables y herramientas de evaluación del proyecto, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del proyecto de la siguiente manera:

- Si el resultado indica que es una tecnología madura, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que es una tecnología desconocida, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.2 Variables y herramientas de evaluación del equipo y la entidad.

Para el comportamiento del equipo de proyecto y la estructura organizacional de la entidad en el diagnóstico se identificaron diez variables que permiten establecer qué tipo de metodología es más apropiada, donde lo apropiado es ubicarse en la franja gris de se muestra en la gráfica.

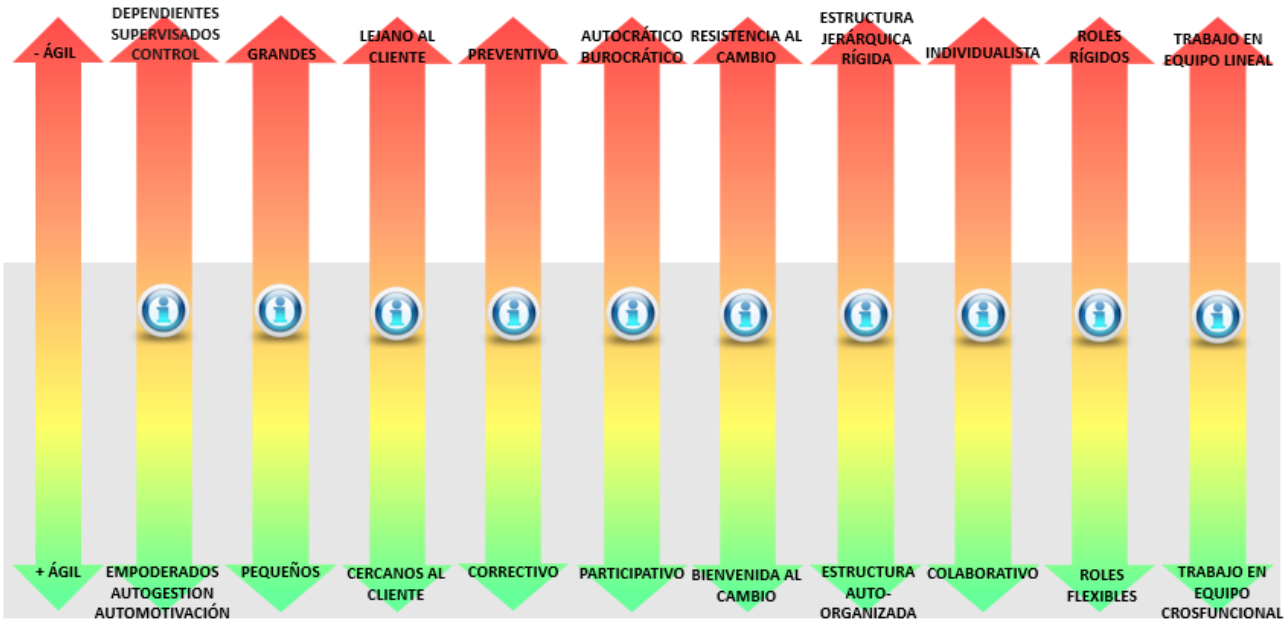


Figura 16. Variables del comportamiento del equipo y la estructura organizacional que determinan idoneidad de la metodología. Fuente: Corporación Colombia Digital.

Las herramientas para evaluar cada una de las variables se explican a continuación:

2.3.2.1 Evaluación del empoderamiento del equipo.

El empoderamiento es un proceso cultural de autodeterminación con el que las personas y los equipos participan activamente en las decisiones y actividades que afectan su entorno. En cuanto a los proyectos de desarrollo de *software*, el empoderamiento determina la dinámica de trabajo y la selección de la metodología de desarrollo. Para evaluarlo se deben tener en cuenta las habilidades blandas de cada uno de los integrantes y su capacidad de decidir y responsabilizarse con un objetivo. Se sugiere utilizar como guía de evaluación la siguiente matriz:

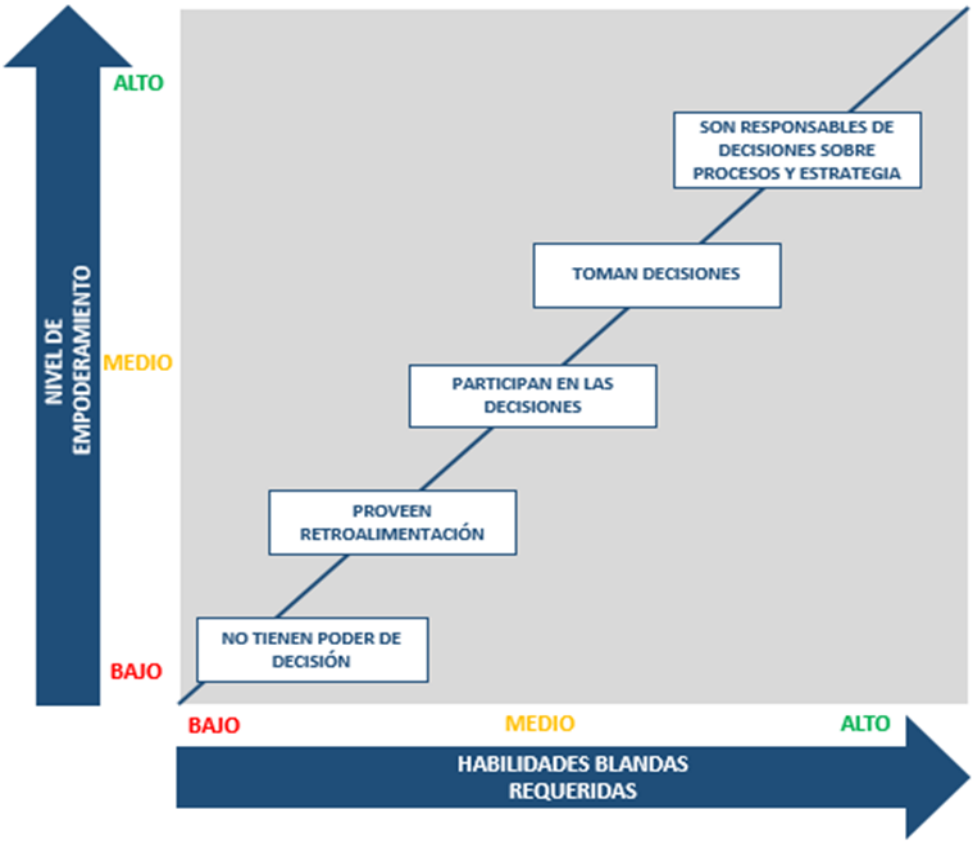


Figura 17. Matriz de evaluación del nivel de empoderamiento del equipo de proyecto con respecto a nivel de madurez de las habilidades blandas requeridas. Fuente: An Approach to Technology Risk Management – Ricardo Valerdi – Ron Khol.

Una vez se tenga la evaluación del empoderamiento del equipo, se retoma la matriz de la figura 16 del numeral 2.3.2 Variables y herramientas de evaluación del equipo y la entidad, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del software de la siguiente manera:

- Si el resultado indica que el equipo es empoderado, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde

- Si el resultado indica que el equipo es dependiente, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.2.2 Evaluación del tamaño de un equipo del proyecto.

En el contexto de proyectos de desarrollo de *software* hay rangos particulares para determinar cuándo un equipo de trabajo es grande o pequeño. Los grupos de más de diez personas son considerados equipos grandes, que significan esfuerzos adicionales en gestión de comunicación, incentivos, etc. Los grupos de diez o menos personas son considerados equipos pequeños en los que la información fluye con rapidez y la rotación es un riesgo de muy alto impacto. El tamaño del equipo permite predecir algunos de los desafíos y ventajas que tendrá la dinámica de trabajo.

Una vez se tenga la evaluación del tamaño de un equipo del proyecto, se retoma la matriz de la figura 16 del numeral 2.3.2 Variables y herramientas de evaluación del equipo y la entidad, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del software de la siguiente manera:

- Si el resultado indica que es pequeño, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que es grande, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.2.3 Evaluación de la cercanía al cliente.

Esta variable busca evaluar la sincronía en la comunicación y la dispersión cultural que existe entre los usuarios y el equipo de desarrollo. Ambos parámetros determinan la dinámica de trabajo del equipo del proyecto y, por lo tanto, son un factor relevante en la selección de la metodología de desarrollo.

La siguiente gráfica ilustra una forma de evaluar la cercanía al cliente según el medio de contacto que se usa con mayor frecuencia en la comunicación. En los casos en que los documentos y el correo electrónico dominan la comunicación entre usuario y desarrollador, se puede decir que la relación es lejana. En los casos en que las reuniones personales y virtuales dominan la comunicación entre usuario y desarrollador se puede decir que la relación es mucho más cercana

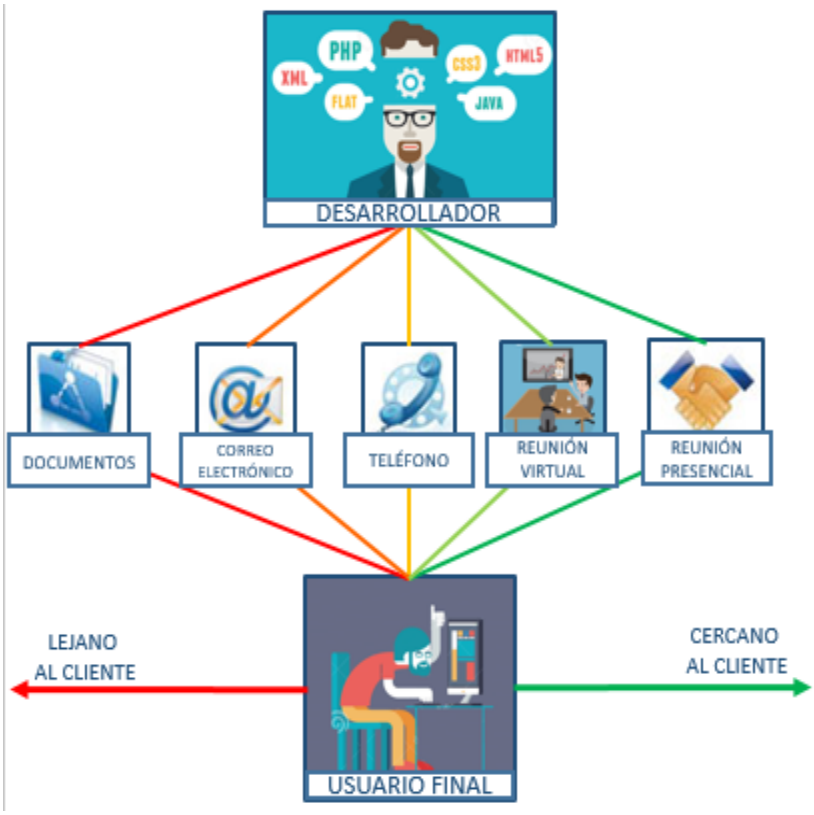


Figura 18. Medios de contacto dominantes en la comunicación entre el equipo desarrollador y usuario final. Fuente: Organizational Behaviour Huczynski y Buchanan.

Otra herramienta que permite evaluar la sincronía en la comunicación es la matriz que se muestra a continuación y que considera las dimensiones de espacio y tiempo que determinan la relación entre ambas partes:

	Espacio

		Coincide	No coincide
Tiempo	**Coincide**	Síncrono – Misma ubicación Reuniones presenciales	Síncrono – Ubicación diferente Llamadas telefónicas Video conferencias
	No coincide	Asíncrono – Misma ubicación Salas de trabajo.	Asíncrono– Ubicación diferente Correo electrónico. Intranet

Tabla 4. Matriz de intercambio de información teniendo en cuenta el espacio y tiempo. Fuente: Corporación Colombia Digital.

Finalmente, la evaluación de la dispersión cultural del grupo puede ser hecha calificando nueve aspectos que determinan la cercanía entre usuarios finales y el equipo desarrollador. El evaluador se puede basar en la siguiente matriz que califica la cercanía entre tres niveles: bajo o lejano, medio y alto o lejano.

	Bajo	Medio	Alto
Cercanía entre las diferentes disciplinas			
Entendimiento de la tecnología			
Afinidad en el estilo de trabajo			
Cercanía cultural			
Sincronía de la comunicación			
Frecuencia en la comunicación			
Calidad de la comunicación			
Cercanía física (frecuencia con que trabajan en el mismo espacio)			
Cohesión de grupo			

Tabla 5. Matriz de evaluación de la dispersión cultural entre usuarios finales y el equipo desarrollador. Fuente: Organizational Behaviour Huczynski y Buchanan.

Una vez se tenga la evaluación de la cercanía al cliente, se retoma la matriz de la figura 16 del numeral 2.3.2 Variables y herramientas de evaluación del equipo y la entidad, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del software de la siguiente manera:

- Si el resultado indica que es cercano al cliente, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que es lejano al cliente, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.2.4 Evaluación de la respuesta del equipo frente a los problemas.

En los equipos de proyecto se puede observar que frente a los problemas pueden existir dos tipos de enfoque: preventivo y correctivo; aunque según la situación, un mismo grupo puede usar cualquiera de los dos enfoques, sin embargo se observa que uno de los dos domina la dinámica de trabajo. A continuación se presentan las actividades y características que predominan en cada caso.

- Enfoque preventivo:

☑ El equipo identifica los riesgos
☑ El equipo diseña estrategias de protección
☑ El equipo implementa las estrategias de prevención
☑ El equipo prueba las estrategias de prevención
☑ El equipo asegura el uso y apropiación de las estrategias preventivas
☑ El equipo cuenta con un plan de mejoramiento continuo para esas estrategias

- Enfoque correctivo:

☑ El equipo identifica y describe el problema
☑ El equipo controla las consecuencias no deseadas

- ☑ El equipo identifica la causa del problema
- ☑ El equipo diseña la acción correctiva
- ☑ El equipo implementa la acción correctiva
- ☑ El equipo evalúa la efectividad de la acción correctiva

Una vez se tenga la evaluación de la respuesta del equipo frente a los problemas, se retoma la matriz de la figura 16 del numeral 2.3.2 Variables y herramientas de evaluación del equipo y la entidad, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del software de la siguiente manera:

- Si el resultado indica que es correctivo, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que es preventivo, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.2.5 Evaluación del estilo de liderazgo en el equipo.

Uno de los principales factores que define la dinámica de trabajo de un equipo es el estilo de liderazgo bajo el cual se toman las decisiones en el grupo. Puede comprender desde actitudes con la que las decisiones son completamente controladas por los líderes de la organización o proyecto hasta estilos en los que la participación del líder es reducida y el protagonismo lo tienen quienes integran el proyecto.

La siguiente figura muestra en el extremo izquierdo las características de un liderazgo autocrático y en el extremo derecho las características de un liderazgo participativo. Entre los dos extremos existen combinaciones de ambos estilos y de acuerdo con la descripción, se puede evaluar cuál de los dos domina la cultura organizacional del grupo o entidad.

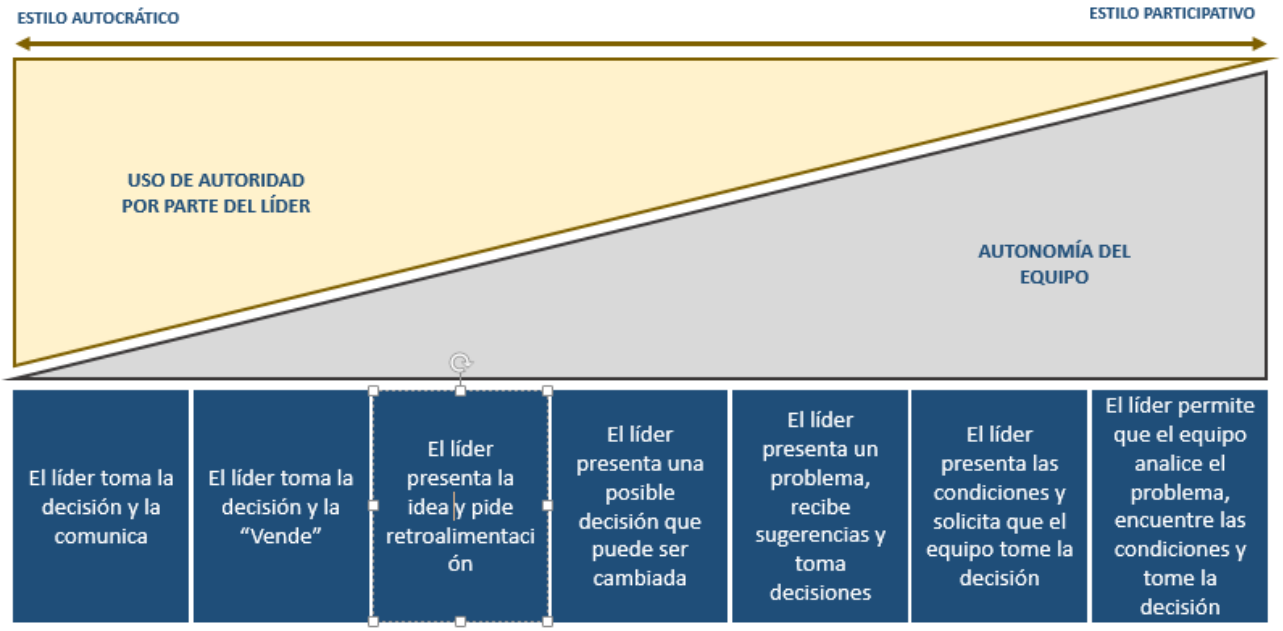


Figura 19. Estilo de liderazgo en los equipos de proyecto. Fuente: Modelo de Vroom – Yetton.

Una vez se tenga la evaluación del estilo de liderazgo en el equipo, se retoma la matriz de la figura 16 del numeral 2.3.2 Variables y herramientas de evaluación del equipo y la entidad, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del software de la siguiente manera:

- Si el resultado indica que es participativo, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que es autocrático, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.2.6 Evaluación de la actitud del equipo frente al cambio.

La actitud que los individuos y grupos tienen frente al cambio determina la dinámica de trabajo del proyecto y el tipo de esfuerzos que se deben hacer para obtener los resultados planeados. En términos generales los individuos pueden tener tres actitudes frente al cambio: buscan sabotear, son observadores pasivos o participan activamente en la

construcción del cambio. Cada una de esas tres actitudes está fuertemente ligada al nivel de entendimiento de las motivaciones que suscitaron la necesidad de cambio y las consecuencias que tiene el proceso para los interesados y la entidad.

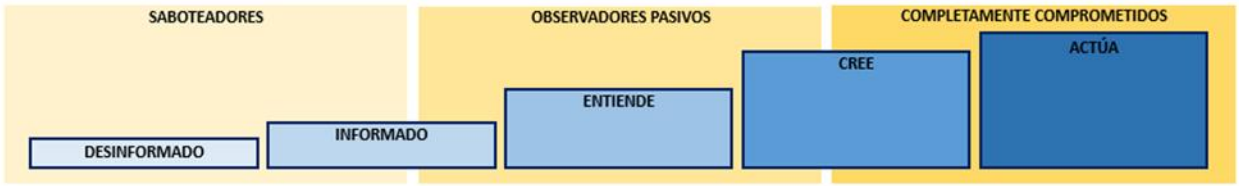


Figura 20. Actitud de los individuos frente al cambio. Fuente: Change management - Harvard Business Review.

Las características que permiten identificar la actitud que tiene un equipo frente al cambio se describen a continuación para cada una de las tres categorías definidas:

Tipos de individuos	Características	Sugerencias
Saboteadores	• Creen que el cambio no es necesario y que empeorará la situación. • No confían en los líderes que están generando y gestionando el programa de cambio. • No les gusta la forma en que el cambio está iniciando en la organización. • No confían en que el programa de cambio va a ser exitoso. • No proveen retroalimentación ni participan en la planeación e	Aprovechar el lanzamiento (kick off) del proyecto para socializar la importancia de todos y cada uno de los miembros del equipo y resaltar su participación. Mantener informado al equipo conforme a su nivel de jerarquía y conveniencia de socialización de información, para ir presentando avances, importancia de los aportes y la colaboración e impacto que ellos tienen en la consecución del objetivo. Asignar claramente las responsabilidades

	implementación de los procesos de cambio. • Sienten que el cambio significará pérdidas personales (seguridad, dinero, estatus, amigos, etc.) • Creen en el statu quo. • Han sufrido muchos cambios recientemente y no pueden manejar cambios adicionales.	
Observadores Pasivos	• No participan de ninguna forma en los procesos de planeación e implementación del cambio. • No tienen una opinión en favor o en contra del cambio. • No sienten que el cambio los afecte ni positiva ni negativamente. • Evitan involucrarse en el programa de cambio.	Motivar a los observadores pasivos para hacerlos participes activos y colaboradores del proyecto y del proceso de cambio. Exponer los beneficios directos o indirectos que presentan los cambios a los que se enfrenta el área y/o la organización.
Comprometidos	• Creen que el cambio es el enfoque correcto. • Respetan y confían en las personas que lideran el cambio. • Esperan que nuevas oportunidades y retos surjan a partir del cambio.	Este tipo de individuos se pueden comprometer como líderes y hacer que los saboteadores y los observadores pasivos se involucren positivamente en los programas de cambio.

	• Se involucran en la planeación e implementación del programa de cambio. • Confían en que el cambio traerá ganancias personales. • Disfrutan los retos y la emoción que implican los programas de cambio.	

Tabla 6. Características de un equipo frente al cambio. Fuente: Corporación Colombia Digital.

Una vez se tenga la evaluación de la actitud del equipo frente al cambio, se retoma la matriz de la figura 16 del numeral 2.3.2 Variables y herramientas de evaluación del equipo y la entidad, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del software de la siguiente manera:

- Si el resultado indica que acepta el cambio, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que es resistente al cambio, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.2.7 Evaluación de la estructura organizacional.

La estructura organizacional de la entidad y la forma como se estructuran los equipos determinan la dinámica de trabajo del grupo y por lo tanto son un factor importante dentro de la selección de la metodología de desarrollo de *software*. La estructura organizacional puede tener varias configuraciones, pero existen dos modelos principales que influyen drásticamente en la forma en que se comunica el equipo y la forma en que se realizan las actividades. Estos dos tipos de estructura organizacional son: estructura de modelo mecánico y estructura de modelo orgánico. La siguiente grafica muestra la configuración característica de cada una de esas estructuras:

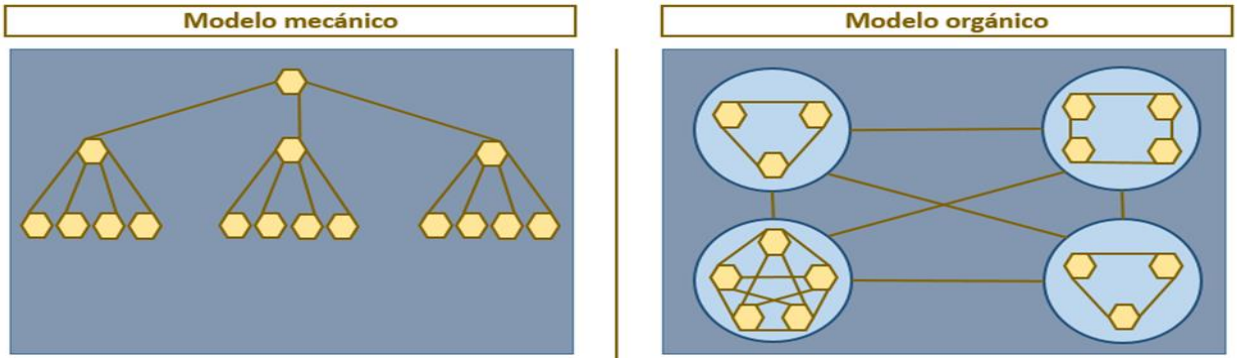


Figura 21. Modelos que describen las dos principales configuraciones de la estructura organizacional. Fuente: Organizational Behaviour – Robbins Judge.

Las características principales que permiten identificar a cada uno de estos modelos se presentan a continuación:

Modelo mecánico.

- ☑ Alta especialización de los funcionarios.
- ☑ Jerarquía rígida.
- ☑ Clara línea de mando.
- ☑ Alta formalización.
- ☑ Centralización.
- ☑ Niveles de control muy acodados.

Modelo orgánico.

- ☑ Equipos cros-funcionales
- ☑ Jerarquía flexible.
- ☑ Libre flujo de información.
- ☑ Baja formalización.
- ☑ Descentralización.
- ☑ Niveles de control muy amplios

Una vez se tenga la evaluación de la estructura organizacional, se retoma la matriz de la figura 16 del numeral 2.3.2 Variables y herramientas de evaluación del equipo y la entidad, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del software de la siguiente manera:

- Si el resultado indica que es una estructura auto-organizada, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que es una estructura jerárquica rígida, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.2.8 Evaluación del ambiente de trabajo del equipo.

El estilo de trabajo del equipo normalmente tiende a ser dominado por dos tipos de conductas: individualismo o colectivismo; estos además determinan la dinámica de trabajo y la selección de la metodología de desarrollo de *software*. Las características principales de ambos estilos son:

- Enfoque individualista:

☑ Prevalecen los intereses personales y de su círculo inmediato dentro de la organización
☑ Alta orientación al YO
☑ Privilegia el derecho a la privacidad
☑ Prevalece la opinión individual
☑ Los incentivos están orientados a premiar: logros personales, independencia, competencia, respeto a la jerarquía y capacidad para cambiar las circunstancias
☑ El incumplimiento de los lineamientos se traduce en sentimientos de culpabilidad

- Enfoque colectivo:

☑ Prevalecen los intereses colectivos de toda la organización
☑ Alta orientación a NOSOTROS

☑ Privilegia el sentido de pertenencia
☑ Prevalece la armonía colectiva
☑ Los incentivos están orientados a premiar: logros colectivos, interdependencia, cooperación, debate, capacidad de adaptación a las circunstancias
☑ El incumplimiento de los lineamientos se traduce en sentimientos de vergüenza

Una vez se tenga la evaluación del ambiente de trabajo del equipo, se retoma la matriz de la figura 16 del numeral 2.3.2 Variables y herramientas de evaluación del equipo y la entidad, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del software de la siguiente manera:

- Si el resultado indica que es colaborativo, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que es individualista, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.2.9 Evaluación de la flexibilidad de los roles en el equipo.

La flexibilidad que tienen los roles en un equipo de trabajo u organización es un factor que contribuye a definir el espectro de posibilidades para enfrentar los desafíos que supone un proyecto de desarrollo de *software*. Por tal razón, es importante tener claro desde un inicio qué flexibilidad ofrece la organización y sus integrantes para asumir retos diferentes a los formalmente establecidos. Para hacer esta evaluación a continuación se listan los aspectos que permiten diferenciar una organización con roles rígidos de una con roles flexibles:

- Características de una organización con roles rígidos:

☑ Bajo tiempo de respuesta: el proceso de evaluar y aprobar cambios es extenso debido a que existen líneas muy complejas de decisión y reporte.

☑ Alta complejidad en la gestión de las cargas laborales: los recursos están disponibles tiempo completo en la organización y la disponibilidad no es aprovechada eficientemente para el proyecto.

☑ Los recursos contratados son altamente especializados: Cada posición en la organización tiene formalmente definida las responsabilidades y tareas por desarrollar. Las personas son contratadas de acuerdo con perfiles especializados definidos según el rol. Se observa alta resistencia y dificultades para asumir responsabilidades y tareas que estén fuera del rol.

- Características de una organización con roles flexibles:

☑ Modelo de la estructura organizacional: normalmente se observa una estructura tipo grafo con diferentes nodos de decisión. Existen líneas de reporte y comunicación alternativas.

☑ Flexibilidad en la asignación de la carga laboral: semiflexible: Al inicio del proyecto se distribuyen las responsabilidades y tareas. La estructura definida no se modifica durante el proyecto. Flexible: las responsabilidades y tareas asignadas cambian durante el proyecto según la necesidad.

☑ La organización y los integrantes del equipo reconocen y adoptan con facilidad el cambio: se crean nuevos equipos y roles para adaptarse a las necesidades del proyecto. Se observan fusiones de equipos y roles según los requerimientos. Se eliminan equipos y roles según los cambios.

Una vez se tenga la evaluación de la flexibilidad de los roles en el equipo, se retoma la matriz de la figura 16 del numeral 2.3.2 Variables y herramientas de evaluación del equipo y la entidad, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del software de la siguiente manera:

- Si el resultado indica que son roles flexibles, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde

- Si el resultado indica que son roles rígidos, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.3.2.10 Trabajo en equipo lineal vs trabajo en equipo cros-funcional.

El último factor que determina la dinámica de trabajo y que resulta ser un factor más para la selección de la metodología de desarrollo es la composición de los equipos. De acuerdo con la naturaleza del proyecto, los equipos pueden ser predominantemente cros-funcionales o lineales. Las características de ambos tipos de equipo se presentan a continuación.

- Equipos cros-funcionales:

☑ Grupos formales de trabajo que están constituidos por funcionarios de igual jerarquía pertenecientes a diferentes áreas de la organización, para cumplir con los objetivos de un proyecto.

☑ Tienen las habilidades y conocimiento para abordar un problema o necesidad desde múltiples perspectivas, ofrecen soluciones robustas a proyectos de alta complejidad.

☑ Representan retos importantes en términos de la gestión de la comunicación, ya que sus integrantes manejan diferentes leguajes especializados.

☑ Experimentan mayores dificultades para armonizar los diferentes estilos de trabajo, por lo que requieren de una curva de aprendizaje prolongada para trabajar conjuntamente.

- Equipos lineales:

☑ Grupos formales de trabajo que están constituidos por funcionarios que pertenecen a una misma área de la organización, para cumplir con los objetivos de un proyecto.

☑ Son altamente especializados en un área de conocimiento.

☑ Manejan un mismo lenguaje del área de conocimiento que dominan, por lo que la comunicación fluye con mayor facilidad.

☑ Tienen mayor afinidad en términos del estilo de trabajo y por lo tanto generar resultados de forma más rápida.

Una vez se tenga la evaluación si el trabajo en equipo es lineal o cros-funcional se retoma la matriz de la figura 16 del numeral 2.3.2 Variables y herramientas de evaluación del equipo y la entidad, para ubicar el resultado que permite establecer el tipo de metodología que favorece el desarrollo del software de la siguiente manera:

- Si el resultado indica que es cros-funcional, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color verde
- Si el resultado indica que es lineal, en la matriz se ubica en el rango con el mismo nombre, el cual se identifica con el color rojo

2.4 ESTRATEGIA PROPUESTA

La estrategia propuesta tiene como objetivo optimizar el modelo de gestión de proyectos de desarrollo de *sistemas de información* a través del uso de una ficha tipo que contiene información sobre la problemática actual, herramientas y técnicas que sirven como base para la buena gestión de los desarrollo de sistemas de información, para los equipos involucrados en este tipo de proyectos.

El propósito final es que las entidades del Estado cuenten con sistemas de información que se conviertan en fuente única de datos útiles para la toma de decisiones, que garanticen la calidad de la información, dispongan recursos de consulta de información, sean fáciles de mantener, escalables, interoperables, seguros, funcionales y sostenibles, tanto financiera como técnicamente.

Las fichas tipo se crean a partir del modelo conceptual que se ilustra a continuación:

Figura 22. Modelo Conceptual. Fuente: Corporación Colombia Digital.

El modelo comienza con una etapa de análisis preliminar en la que se diagnóstica la demanda, para conocer y modelar las necesidades de la entidad. Seguidamente, se diagnóstica la oferta para conocer los esquemas de servicio de los proveedores y las oportunidades de mejora al respecto. Finalmente, se estudia la experiencia de otros países para optimizar el desarrollo de proyectos de *software* y se determinan las iniciativas que pueden ser aplicadas en el contexto colombiano.

La siguiente etapa consiste en construir las fichas tipo teniendo como marco los hallazgos del análisis preliminar, la evaluación de la idoneidad de la metodología y el ciclo de vida de un proyecto de desarrollo de *software* como se muestra en la siguiente gráfica. Es importante resaltar que para el uso de las fichas tipo se debe tener en cuenta la metodología seleccionada para el proyecto, a partir de esto se escogen las herramientas y mejores prácticas.

Figura 23. Ciclo de vida de un proyecto de desarrollo de software. Fuente: Corporación Colombia Digital.

En consecuencia, la estructura definida para la ficha tipo integra los siguientes elementos: el ciclo de vida de los proyectos de *software*, la naturaleza iterativa o secuencial de cada una de sus fases, los problemas frecuentes y oportunidades de mejora encontradas en la etapa de diagnóstico del modelo conceptual, los puntos clave en cada fase, las mejores prácticas de la industria y los componentes esenciales que deben considerarse en los elementos transversales.

Estructura documento de las fichas tipo

	Capítulo / Subcapítulo	Problemas frecuentes	Puntos clave	Mejores prácticas	Elementos transversales
Ciclo de vida	Alcance				
	Requerimientos				
	Diseño/Arquitectura				
	Codificación				
	Pruebas				
	Puesta en producción				
	Uso y apropiación				
	Mantenimiento				

Figura 24. Estructura documento de las fichas tipo. Fuente: Corporación Colombia Digital

3 ALCANCE

3.1 PROBLEMAS FRECUENTES

- No se realiza un adecuado estudio de viabilidad de los proyectos, con el que se podría analizar una serie de necesidades para proponer una solución a corto o largo plazo, que tenga en cuenta restricciones económicas, técnicas, legales y operativas.
- El proceso de planeación de los proyectos de desarrollo de sistemas de información es deficiente, lo que afecta el alcance, cronograma, presupuesto, recursos, etc. Además, no se contemplan curvas de aprendizaje de los proveedores, uso y apropiación de las entidades y fases de estabilización de los productos desarrollados.
- Las áreas funcionales de las entidades involucradas en los proyectos tienen poca comunicación entre ellas, por lo que es necesario crear sinergias entre las mismas, con el fin de evitar duplicar esfuerzos en el levantamiento de información con el que se define el alcance y se conocen las necesidades.
- Se detecta con frecuencia que el alcance inicialmente definido no corresponde con el alcance real que requiere el desarrollo exitoso de la iniciativa, esto es resultado de los largos tiempos en el proceso de contratación, que desactualiza la necesidad inicialmente identificada
- Los altos niveles de rotación de personal en las entidades pueden conllevar a cambios en el alcance del proyecto, por la pérdida de la cronología y la curva de aprendizaje adquirida.

3.2 ASPECTOS GENERALES.

Definir el alcance del proyecto es el proceso mediante el cual se desarrolla una descripción detallada del objetivo, fases y entregables. Durante este proceso, se analizan los riesgos, los supuestos y las restricciones existentes. Definir de manera acertada el alcance del

proyecto es fundamental para su éxito, ya que proporciona un entendimiento común entre los interesados.

Antes de dar inicio al proyecto es indispensable identificar:

1. Los factores ambientales de la entidad: cultura, sistemas, recursos humanos, etc.

2. Los activos de los procesos de la organización: políticas, procesos, normas, información histórica y lecciones aprendidas

3. Los disparadores del proyecto: problema, oportunidad de mercado, requisito o reglas del negocio, cambio tecnológico, legislación, etc.

Mejor Práctica: *La planificación determinará si es factible o no llevar a cabo lo anunciado en el alcance. En caso que sea posible, la planificación deberá detallar cómo se desarrollará el proyecto para cumplir con los objetivos. Esta planificación es gradual, por lo que desarrolla actividades repetitivas e iterativas.*

3.2.1 Proceso de gestión del alcance

Para alcanzar un proyecto exitoso es necesario implementar procesos de gestión del alcance, con la seguridad de que todo el trabajo se llevará a cabo. (Lledo, Pablo, 2013. Director de proyectos).

Procesos	Descripción
Declaración del problema o necesidad	Se identifican los problemas o necesidades
Recopilación de información	Se hace el levantamiento de la información relevante
Restricciones y supuestos de alto nivel	Se determinan cuáles son las limitaciones y variables que afectan el proyecto
Análisis de alternativas	Se busca entender la naturaleza de la necesidad y generar alternativas
Objetivos del proyecto	Se identifica qué se quiere realizar o alcanzar con el proyecto

Riesgos de alto nivel	Se detectan los eventos que pueden afectar positivamente o negativamente el normal desarrollo del proyecto
Resúmenes del cronograma y presupuesto	Se documenta el cronograma y el presupuesto
Identificación y gestión de los interesados	Se conoce a los integrantes e interesados del proyecto
Requisitos de aprobación	Se definen los criterios de aprobación
Acta de constitución	Se elabora un documento con el detalle del trabajo requerido a lo largo del proyecto
Plan de dirección	Se definen, preparan y coordinan todos los planes secundarios para incorporarlos en un solo plan
Planificar la gestión del alcance	Se planea cómo se llevarán a cabo los procesos (requisitos, definición, EDT, validación y control)
Recopilar requisitos	Se registran las necesidades de los interesados para convertirlas en requisitos del proyecto
Definir el alcance	Se documenta el alcance detallado
Línea base del alcance	Se enuncia el alcance del proyecto, la EDT y su diccionario
Crear la estructura de desglose del trabajo o EDT	Se descompone el proyecto en partes más pequeñas que generalmente definen hitos
Validar el alcance	Se obtiene la aceptación formal de la entidad sobre el alcance
Controlar el alcance	Se gestionan los cambios en el alcance

Tabla 7. Procesos de gestión del alcance. Fuente: Pinto, 2014.

3.2.1.1 Declaración del problema o necesidad.

Utilice los siguientes criterios para identificar problemas o necesidades:

- Las iniciativas técnicas u operativas para el mejoramiento de procesos
- La repotencialización de productos o servicios
- Los nuevos procesos de negocio para obtener mayor eficiencia
- La reconfiguración del portafolio de bienes y servicios
- Las nuevas alianzas estratégicas
- El mejoramiento de productos o servicios según las tendencias
- El mejoramiento de la comunicación organizacional
- El mejor gestión de las cadenas de suministro
- El mejor coordinación entre las áreas que integran la organización

3.2.1.2 Recopilación de información.

Para realizar esta actividad tenga en cuenta lo siguiente:

- Los factores ambientales del proyecto: Cultura, sistemas, recursos, etc.
- Los activos de los procesos de la organización: políticas, procedimientos, normas, leyes, etcétera.
- Las fechas objetivo del proyecto
- Los disparadores del proyecto: problemas, necesidades, etc.
- La identificación de posibles proveedores
- Los portafolios de productos y servicios en el mercado: estudio de mercado
- El nivel de apoyo del equipo de liderazgo
- Las fuentes de financiación
- El apoyo requerido
- Los recursos necesarios

3.2.1.3 Restricciones y supuestos del proyecto.

RESTRICCIONES	SUPUESTOS
Limitaciones de tiempo	Disponibilidad de recursos
Limitaciones del presupuesto	Costos
Limitaciones del mercado	Inflación
Limitaciones de capacidad (recursos disponibles)	Variación TRM
Limitaciones legales	Depreciación
Limitaciones de la tecnología	Prioridades del negocio
Limitaciones de conocimiento	
Limitaciones de seguridad	

Tabla 8. Restricciones y supuestos del proyecto. Fuente: PMBOK Guide – 5th edition.

3.2.1.4 Análisis de alternativas.

El primer paso es entender la naturaleza de la necesidad o del problema, para esto se debe tener en cuenta:

- La determinación de la percepción del problema o necesidad
- El análisis de la necesidad o el problema con los datos e información disponible
- La división del problema o necesidad en elementos que faciliten su comprensión y disminuya su complejidad
- La identificación de causas y efectos para cada elemento
- La identificación de las relaciones entre elementos
- La evaluación del impacto de cada elemento en el problema o necesidad

El siguiente paso es generar alternativas de solución teniendo en cuenta:

- Una lluvia de ideas con las alternativas para resolver el problema o cubrir la necesidad
- La definición de los criterios de evaluación de las alternativas: riesgo, costo, tiempo, capacidad, indicadores y recursos requeridos
- La definición del procedimiento de evaluación de las alternativas
- La definición del procedimiento de selección de las alternativas

El esquema de trabajo para realizar este proceso se muestra en la siguiente figura:

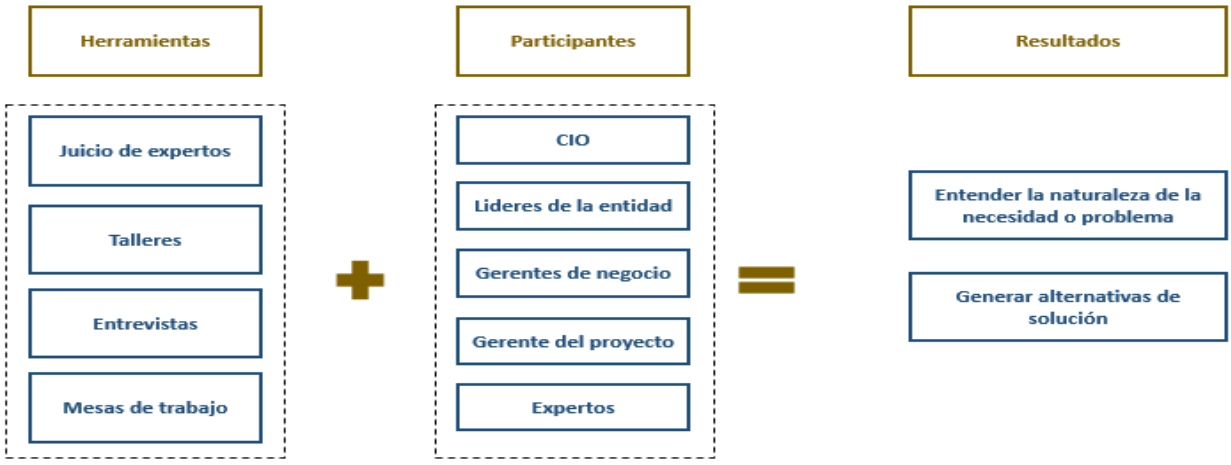


Figura 25. Esquema de trabajo. Fuente: PMBOK Guide – 5th edition.

3.2.1.5 Objetivos del proyecto.

Los objetivos deben cumplir con lo que se describe en la siguiente figura.

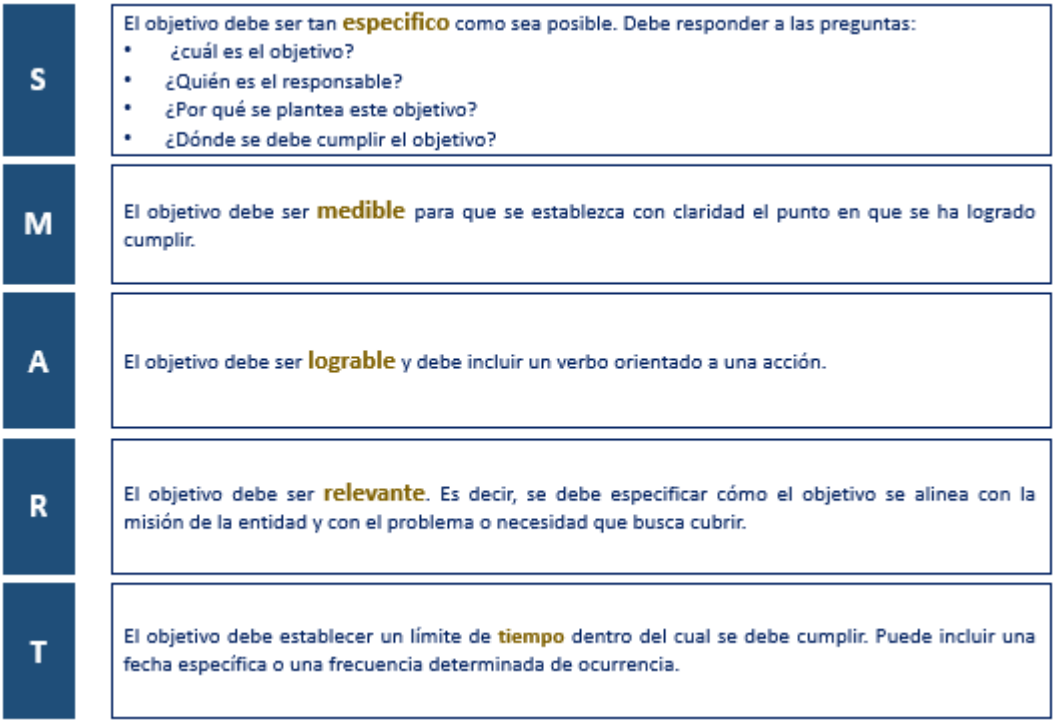


Figura 26. Objetivos del proyectos. Fuente: Corporación Colombia Digital.

3.2.1.6 Riesgos del proyecto.

La gestión de los riesgos se refiere al proceso sistemático de planificar, identificar, analizar, responder y controlar los riesgos del proyecto. Se trata de maximizar la probabilidad de ocurrencia de los sucesos positivos y minimizar la probabilidad de los sucesos negativos.

Procesos	Descripción
Planificar la gestión de riesgos	Se planea cómo se planificarán y ejecutarán las actividades de identificación, análisis, respuesta y seguimiento de los riesgos.
Identificar los riesgos	Se identifica qué riesgos afectan al proyecto
Realizar un análisis cualitativo de riesgos	Se estima de manera cualitativa la probabilidad y el impacto de cada riesgo para hacer una priorización de los mismos.

Realizar un análisis cuantitativo de riesgos	Se estima numéricamente la probabilidad y el impacto, para priorizar los riesgos con mayor precisión.
Planificar la respuesta a los riesgos	Se planifican las acciones que se llevarán a cabo para mejorar las oportunidades y reducir las amenazas.
Controlar los riesgos	Se monitorean y ejecutan los planes de respuesta al riesgo.

Tabla 9. Procesos de gestión de riesgos. Fuente: PMBOK Guide – 5th edition.

El riesgo representa el impacto potencial de todas las amenazas u oportunidades de mejora que podrían afectar los logros de los objetivos del proyecto. Durante el proceso de planificar los riesgos se debe dar respuesta a los siguientes interrogantes:

- ¿Quiénes van a identificar los riesgos?
- ¿Cuándo se llevará a cabo la identificación?
- ¿Qué escala se utilizará para el análisis cualitativo de riesgos?
- ¿Cómo se priorizarán los riesgos?
- ¿Qué herramientas se utilizarán para el análisis cuantitativo?
- ¿Cuáles serán las estrategias que se implementarán para cada riesgo?
- ¿Con qué frecuencia se realizará el seguimiento?

Una vez realizado el plan de gestión de riesgos, es necesario comenzar con la identificación de los eventos que pudiesen afectar el resultado del proyecto, ya sea de modo positivo o negativo. Se debe prestar especial atención a la identificación de los sucesos que puedan afectar seriamente al proyecto, aun cuando su probabilidad de ocurrencia fuese muy baja. Las herramientas para realizar esta actividad pueden ser: la revisión de documentación, recolección de información (entrevistas, lluvias de ideas, análisis de causa raíz, etc.), listas de chequeo, análisis de supuestos, diagramas causa – efecto, análisis DOFA y los juicios de expertos.

Mejor Práctica: *Hay ocasiones en que no se conoce con precisión la probabilidad de ocurrencia de un evento riesgoso, por lo contrario lo único que se tiene es una percepción basada en una opinión o una investigación que tal vez no es del todo correcta. En estos*

casos, se puede utilizar un rango de probabilidad estimado y analizar la sensibilidad del posible impacto de cada escenario sobre los objetivos del proyecto.

Ejemplos de preguntas sugeridas para identificar riesgos:

- ¿Son estables los requisitos de proyecto?
- ¿El diseño depende de supuestos optimistas?
- ¿Estarán oportunamente los recursos disponibles?
- ¿El desarrollo está soportado por la infraestructura requerida?
- ¿El cronograma del proyecto depende de otros proyectos?
- ¿Los procedimientos de estimación de costos son confiables?
- ¿La cultura organizacional está alineada con la metodología del proyecto?
- ¿Se tiene experiencia en este tipo de proyectos?
- ¿Qué dificultades externas pueden encontrarse?

La experiencia indica que los imprevistos son los tipos de riesgos más peligrosos para la viabilidad de un proyecto, debido a que son desconocidos es muy fácil omitirlos. De allí que una de las tareas más importantes durante la gestión de riesgos es la identificación de la mayor cantidad posible de eventos riesgosos, a pesar de la indudable dificultad que presenta esta tarea para el caso de los imprevistos.

Evaluación de riesgos

El proceso para realizar la evaluación cualitativa de riegos se muestra en la siguiente figura, cuyo objetivo es identificar en la matriz el nivel de riesgo según su impacto y probabilidad de ocurrencia, para luego tomar los correctivos necesarios para mitigarlo. (PMBOK Guide – 5th edition).

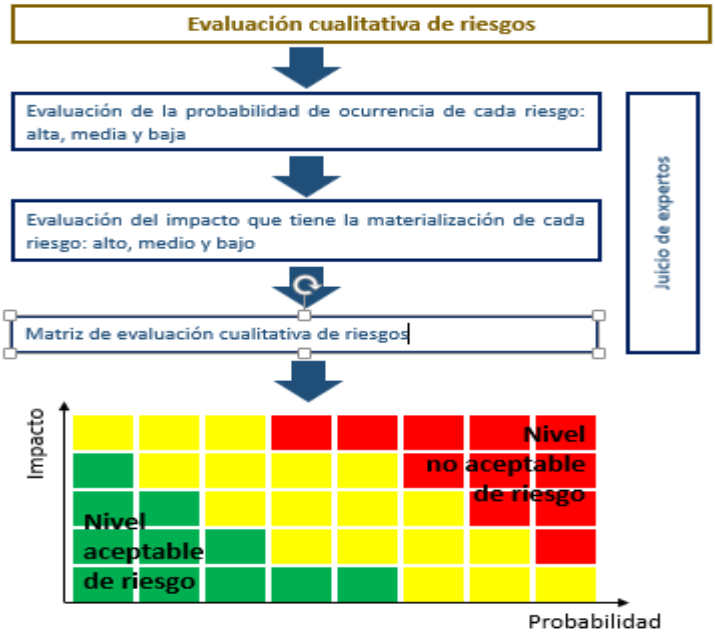
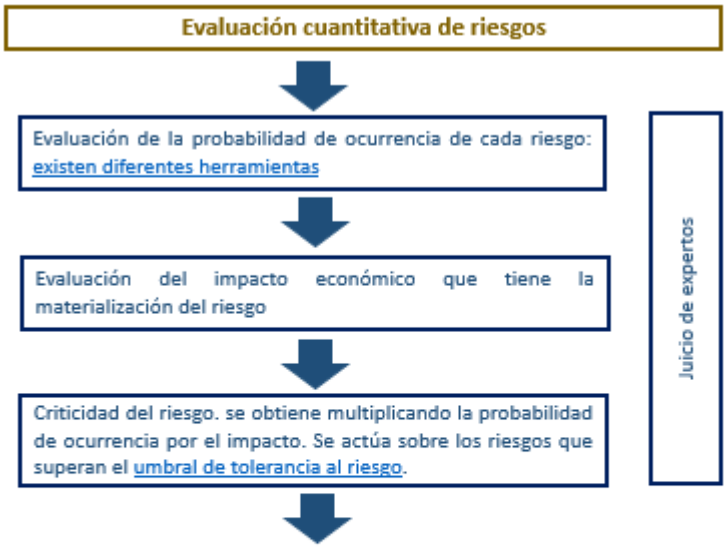


Figura 27. Evaluación cualitativa de riesgos. (PMBOK Guide – 5th edition).

El proceso para la evaluación cuantitativa de riegos se muestra en la siguiente figura, para lo que puede utilizar herramientas como el análisis de Monte Carlo, análisis del árbol de decisiones, el valor monetario esperado, el análisis de sensibilidad y la estimación por tres valores:

El análisis de decisiones y la evaluación de riesgos se fundamenta en modelos matemáticos, pero es igualmente importante su componente artesanal en el sentido de que cada problema de decisión y cada escenario de riesgo es único y exige una importante labor de construcción que no sigue procedimientos mecánicos y que requiere profesionales con mente analítica, flexible y creativa.



Riesgo	Probabilidad	Impacto	Criticidad
Riesgo A	Probabilidad A	Impacto A	Probabilidad A x Impacto A
Riesgo B	Probabilidad B	Impacto B	Probabilidad B x Impacto B
Riesgo C	Probabilidad C	Impacto C	Probabilidad C x Impacto C
Riesgo D	Probabilidad D	Impacto D	Probabilidad D x Impacto D

Figura 28. Evaluación cuantitativa de riesgos. Fuente: PMBOK Guide – 5th edition.

El riesgo no se cuantifica sólo por su probabilidad de ocurrencia, sino también por su impacto sobre los objetivos del proyecto: alcance, tiempo, costo y calidad.

La probabilidad de ocurrencia se puede clasificar de acuerdo con la siguiente tabla:

Nivel	Definición
Alta = 5	La amenaza está altamente motivada y es capaz de llevarse a cabo
Media - Alta =4	La amenaza es fundamentada y posible
Media = 3	La amenaza es posible
Media - Baja = 2	La amenaza no posee la suficiente capacidad
Baja = 1	La amenaza no posee la suficiente motivación y capacidad

Tabla 10. Probabilidad de ocurrencia de un evento determinado. Fuente: PMBOK Guide – 5th edition.

Mejor Práctica: *Se puede obtener una buena estimación de los beneficios o costos esperados de un evento riesgoso si se multiplica su probabilidad de ocurrencia por el impacto.*

Para los riesgos identificados y cuantificados, se puede estimar una reserva monetaria para contingencias, estas últimas forman parte de la línea base de costo del proyecto. Por otra parte, los riesgos desconocidos no se pueden gestionar de manera proactiva, sin embargo para estos se podría asignar una reserva de gestión general, que no forma parte de la línea base de costo, pero que se incluye en el presupuesto total del proyecto.

A continuación se listan los riesgos más comunes en los proyectos de desarrollo de sistemas de información. Esto se hace con carácter informativo, la matriz debe ser definida por la entidad de acuerdo con el proyecto:

- Errores en la estimación del presupuesto
- Retrasos en el cronograma
- Que los costos sobrepasen los límites establecidos
- Cambios en la definición del alcance
- Cambios en la planeación del proyecto
- Tener recursos técnicos insuficientes
- Problemas de confiabilidad
- Cambios en las especificaciones
- No ejercer a tiempo los controles correctivos
- Cambios en precios iniciales
- Realizar una planeación demasiado optimista
- No tener recursos disponibles de acuerdo con el plan
- Que las tareas precedentes estén incompletas
- Cambios en la regulación o generados por la entidad

- Cambios de personal: se refiere a la gente con experiencia que abandona el proyecto antes de que finalice
- La incertidumbre: se da cuando no se conoce la probabilidad de ocurrencia de un evento.
- Los imprevistos: son aquellos riesgos desconocidos que pueden ocurrir sin anticipar su ocurrencia.

Mejor Práctica: *Utilice una metodología marco, guía de buenas prácticas o directrices como el PMBOK, pero adapte los insumos, herramientas, técnicas y salidas según las necesidades y la complejidad del proyecto.*

Umbral de tolerancia al riesgo y factores que lo definen

Las siguientes figuras muestran los factores que definen el riesgo y el comportamiento frente al nivel de tolerancia del riesgo:

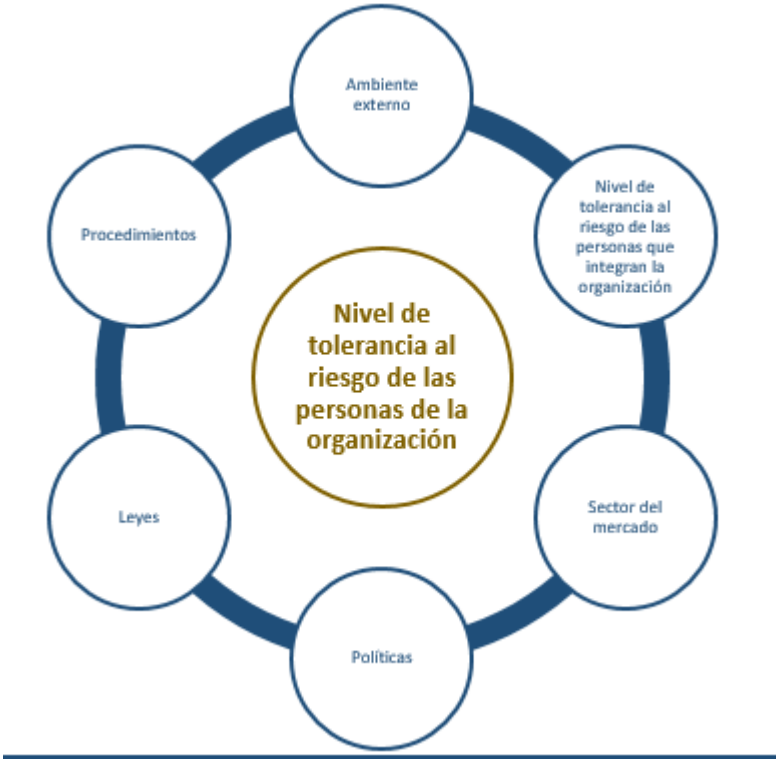


Figura 29. Factores que definen el nivel de tolerancia al riesgo. Fuente: PMBOK Guide – 5th edition.

Figura 30. Comportamiento frente al nivel de tolerancia al riesgo. Fuente: PMBOK Guide – 5th edition.

Con el resultado obtenido se identifican cuales riesgos requieren alguna acción de mitigación.

Documentación de los riesgos

Cuando un riesgo es identificado, es indispensable documentarlo para tomar las acciones correspondientes, para esto se sugiere tener en cuenta la siguiente tabla.

Riesgo	Criticidad	Estrategia	Plan	Disparador	Responsable
Descripción del riesgo	Resultado de la evaluación cuantitativa o cualitativa	Tipo de estrategia seleccionada para gestionar el riesgo	Descripción detallada del plan de acción ante la materialización de la amenaza.	Factores que disparan el plan de acción.	Persona responsable de monitorear y gestionar el riesgo.

Tabla 11. Documentación de los riesgos. Fuente: PMBOK Guide – 5th edition.

Monitoreo y control de riesgos

El proceso final es el monitoreo y control de los riesgos para lo que debe realizar las siguientes actividades:

- Implementar planes de acción frente a los riesgos
- Seguimiento a los riesgos identificados

- Monitoreo de los riesgos residuales
- Monitoreo de nuevos riesgos
- Auditoria sobre efectividad de la gestión de riesgos
- Gestión de los costos asociados a los riesgos
- Gestión de los responsables de los riesgos

3.2.1.7 Resumen del cronograma y presupuesto.

La siguiente figura muestra la información relevante que debe tener la documentación del cronograma y del presupuesto:

Figura 31. Información de cronograma y presupuesto. Fuente: PMBOK Guide – 5th edition.

3.2.1.8 Identificación de interesados.

En la fase de definición del alcance es importante hacer un mapeo inicial de los interesados relevantes, el ciclo de gestión de estos se presenta en la siguiente figura:

Figura 32. Ciclo de gestión de los intersados. Fuente: Shwalbe, 2014 - Wiegers y Beatty, 2013.

Con el fin de facilitar dicha identificación, la siguiente es una lista de los posibles interesados para un proyecto de esta naturaleza. Es importante resaltar que este listado no es exhaustivo y puede estar sujeto a los cambios que el usuario de esta ficha tipo considere pertinentes:

Asociados al proceso de adquisición		
Gerente de compras	Compradores	Personal de soporte legal
Gerente de cuentas por pagar	Analistas de cuentas por pagar	Gerente logístico
Gerente financiero		
Asociados a la gestión del proyecto		
Gerente de proyecto	Analista de negocio	Oficina de gerencia de proyectos
Auditores/supervisores/interventores		
Asociados al desarrollo		
Arquitecto de la aplicación	Diseñador	Desarrollador
Modelador de datos	Analista de procesos	Analista de pruebas
Analista de calidad	Documentador	Administrador de la base de datos
Ingeniero de *hardware*	Arquitecto de sistemas de información	Arquitecto de la solución de negocio

Analista de infraestructura	Gerente de desarrollo	Contratistas
Subcontratistas	Proveedores	Certificadores
Analista de seguridad		
Asociado a la gestión del desarrollo		
Dueño/responsable del desarrollo	Instaladores	Personal de mantenimiento
Personal de soporte	*Chief Information Officer* - CIO	Personal de entrenamiento

Tabla 12. Posibles grupos de interesados en un proyecto de desarrollo de software. Fuentes: Wiegers & Beatty, 2013 - Sharp, Galal, & Finkelstein, 1999.

Adicionalmente, se recomienda determinar qué tan involucrados están cada uno de los interesados con el proyecto, según la siguiente tabla:

Nivel de involucramiento de los interesados	
Desinformado sobre el cambio	No sabe del proyecto y no conoce sus potenciales efectos.
Resistente al cambio	Sabe del proyecto, pero se resiste al cambio.
Neutral al cambio	Sabe del proyecto, pero no le interesa. No ofrece resistencia, pero tampoco apoya el proyecto.
Apoya el cambio	Conoce el proyecto y lo apoya.
Lidera el cambio	Conoce el proyecto y se involucra para que otros lo apoyen.

Tabla 13. Nivel de involucramiento de los interesados. Fuentes: Schwalbe, 2014.

Una vez los interesados en el proyecto han sido identificados, deben ser evaluados en términos de su poder dentro del proyecto y de su nivel de interés. La siguiente figura muestra la estrategia para seguir con cada interesado de acuerdo a esta evaluación:

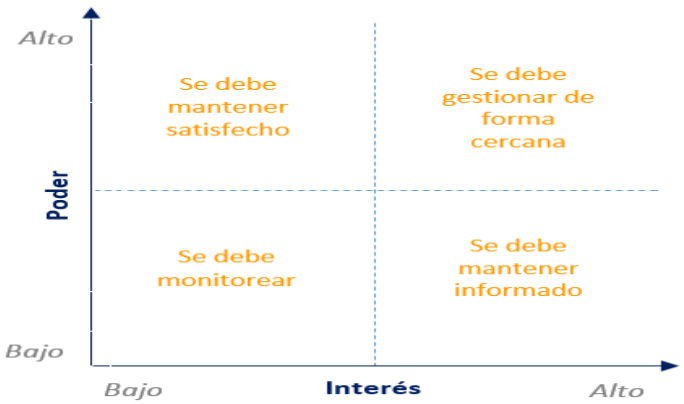


Tabla 14. Matriz de gestión de interesados de acuerdo a su Interés/Poder. Fuente: Schwalbe, 2014.

__Mejor Práctica:__ Al enfrentarse a situaciones difíciles con los interesados se sugiere ser claro desde el comienzo, explicar detalladamente las consecuencias, diseñar planes de contingencia, evitar sorpresas y tener siempre definida una posición sobre el asunto por discutir.

3.2.1.9 Requisitos de aprobación del proyecto.

Se deben tener en cuenta los siguientes aspectos

Comprender el proceso de aprobación de la entidad:

- El gerente de proyecto debe tener claros los procesos definidos por la entidad para aprobar un proyecto.
- En el caso que no existan procesos formalmente definidos, se debe proponer un procedimiento de aprobación.

Entender la alineación estratégica del proyecto con los objetivos de la entidad:

- El gerente de proyecto debe conocer la visión, estrategia y objetivos de mayor nivel de la entidad.
- El proyecto debe contribuir a alcanzar uno o varios de los objetivos de mayor nivel de la entidad.
- El gerente debe tener la capacidad de explicar cómo el proyecto contribuye a alcanzar esos objetivos de mayor nivel.

Identificar los interesados con poder de influir en la aprobación y alinearlos:

- El gerente de proyecto debe identificar los actores claves en el proceso de aprobación de un proyecto.
- El gerente debe “vender” la iniciativa a quienes verifican el proyecto, con el fin de obtener su apoyo en el proceso de aprobación.

- Para “vender” la iniciativa, el gerente debe explicar los beneficios y el impacto que tienen en los indicadores operativos o financieros, dependiendo de la audiencia objetivo.

Conocer las necesidades de información para surtir el proceso de aprobación:

- El gerente de proyecto debe conocer a fondo las etapas del proyecto para explicar el impacto en cada área y el apoyo requerido.
- El gerente debe presentar la información de costos y recursos requeridos a lo largo del proyecto.
- El gerente debe tener claro el retorno de la inversión (ROI), el valor presente neto (VPN) o cualquier otro indicador financiero relevante que soporte la conveniencia del proyecto.

3.2.1.10 Acta de constitución del proyecto.

Es una narración detallada de todo el trabajo requerido a lo largo del proyecto. La información que debe incluir este documento es:

- Los antecedentes, necesidades y problemas de la entidad
- Los objetivos del proyecto
- La descripción breve del proyecto, las condiciones bajo las que se realiza, los beneficios esperados y las prioridades
- Un listado preliminar y la descripción breve de las tareas que se ejecutarán
- La línea temporal y los hitos inicialmente definidos
- Los resultados esperados del proyecto: entregables, seguridad, lugar de ejecución, plazo de ejecución, etc.
- Las fuentes de financiación del proyecto
- El resumen de las restricciones del proyecto

La estructura sugerida para el acta de constitución del proyecto se muestra en la siguiente figura:

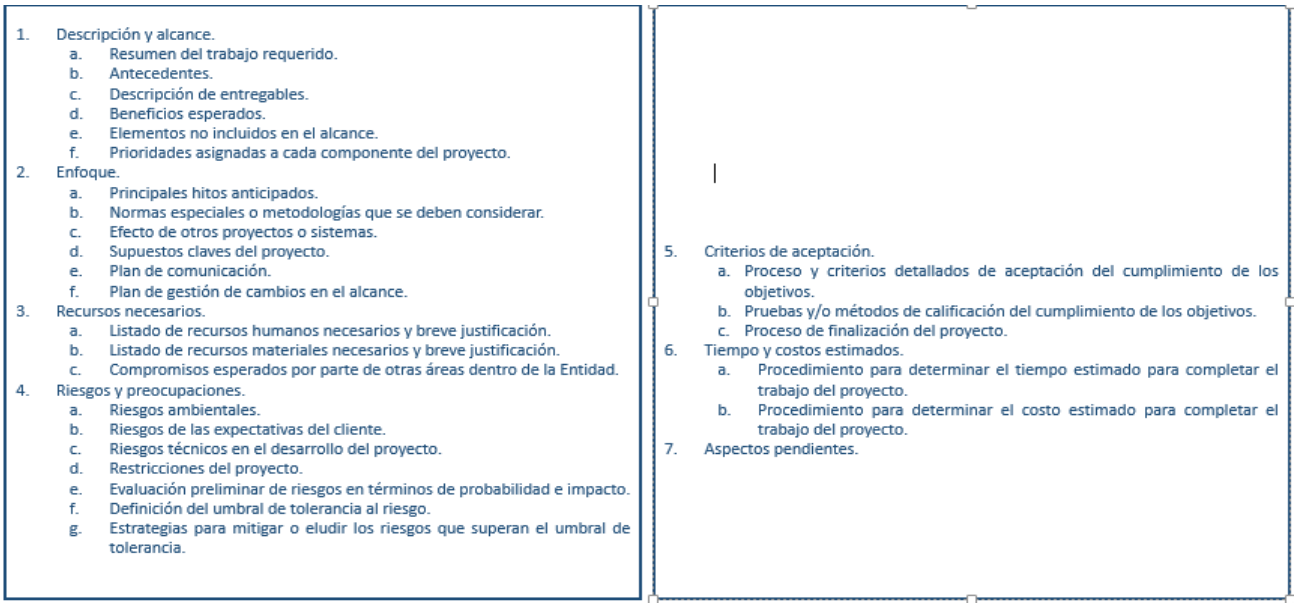


Figura 33. Estructra acta de constitución. Fuente: PMBOK Guide – 5th edition.

3.2.1.11 Plan de dirección del proyecto.

Cuando planifique los proyectos tenga en cuenta las siguientes tareas:

- Definir los roles y responsabilidades de cada miembro del equipo
- Desarrollar el cronograma y presupuesto en conjunto con el equipo de trabajo
- Desarrollar líneas base (alcance, tiempo y costo) y confirmar con el equipo que los objetivos podrán cumplirse
- Determinar estándares y establecer métricas de calidad
- Gestionar los riesgos: identificación, análisis cualitativo y cuantitativo, y el plan de respuesta
- Planificar la evaluación del desempeño del proyecto
- Planificar las necesidades de comunicación de los interesados
- Recopilar los requisitos del proyecto antes de comenzar con la planificación
- Utilizar técnicas de estimación de esfuerzo que permitan ahorrar costos y tiempo en el desarrollo

El plan para la dirección del proyecto debe ser realista y aprobado por los principales interesados. Suele incluir lo siguiente:

- Ciclo de vida del proyecto
- Procesos que se utilizarán en cada fase del proyecto
- Herramientas y técnicas que se emplearán
- Detalle de cómo se ejecutará y controlará el trabajo
- Plan de gestión del cambio
- Registro de cómo se realizará la gestión de la configuración
- Líneas base: alcance, tiempo y costo
- Registro de riesgos, análisis de vulnerabilidad y mitigación
- Todos los planes: alcance, requisitos, tiempo, costo, calidad, mejora de procesos, recursos humanos, comunicaciones, riesgos, adquisiciones y grupo de interesados

Componentes del plan para la dirección del proyecto

Plan de gestión del alcance

El plan de gestión del alcance es un documento donde se definen los procedimientos que se llevarán a cabo para:

- Preparar el enunciado o declaración del alcance
- Crear y aprobar la EDT
- Realizar la validación del alcance
- Procesar y aprobar los cambios en el alcance

Los responsables de implementar las tareas deberían participar en la elaboración del plan del alcance. (PMBOK Guide – 5th edition).

Aspectos a considerar en el desarrollo del plan de gestión del alcance

- Factores ambientales de la organización: Variables políticas, variables económicas, variables culturales, tecnología, variables socio-culturales, tendencias mundiales,

clientes, competencia, proveedores, comunidades, asociaciones, empleados, gerencia, etcétera.

- Activos de los procesos de la organización: Políticas, procedimientos, leyes, decretos, información histórica, lecciones aprendidas.
- Juicio de expertos: Aportes hechos por un grupo de personas conocedora del tema y con amplia experiencia.
- Reuniones: Encuentros del equipo de trabajo que tiene la responsabilidad de desarrollar el alcance del proyecto.

Plan de gestión de los requisitos

Describe los procesos asociados al análisis, documentación y gestión de los requisitos durante el proyecto como son requisitos de negocio, de los interesados, del *software,* del proyecto, de transición, supuestos, dependencias y restricciones de los requisitos

Plan de gestión del cronograma

Describe los procesos, políticas y procedimientos asociados a desarrollar, gestionar, ejecutar y controlar el cronograma. Lo componen lista de actividades y estimación de su duración, atributos de las actividades, lista de hitos, diagrama de red del cronograma del proyecto, calendario de recursos, análisis de ruta crítica y optimización de recursos.

Plan de gestión de costos

Describe los procesos, políticas y procedimientos asociados a planear, estimar, financiar, gestionar y controlar los costos de un proyecto.

Plan de gestión de calidad

Describe los procesos, políticas y procedimientos asociados a definir los objetivos y responsabilidades de calidad en el proyecto, cuyo enfoque es la satisfacción del cliente, la prevención antes que la inspección, mejora continua, involucramiento de la alta dirección y costo de la calidad.

Herramientas: Análisis de costo beneficio, costo de la calidad, diagramas causa efecto, diagramas de flujo, listas de verificación, diagramas de Pareto, histogramas, diagramas de control, diagramas de dispersión, estudios comparativos, muestreo estadístico, tormenta de ideas, reuniones, etcétera.

Plan de gestión de los recursos humanos

Describe los procesos, políticas y procedimientos asociados para organizar, gestionar y conducir el equipo de trabajo del proyecto.

Figura 34. Herramientas para la gestión de los recursos humanos. Fuente: PMBOK Guide – 5th edition.

Plan de gestión de las comunicaciones

Describe los procesos, políticas y procedimientos asociados para planear, crear, recopilar, distribuir, almacenar, recuperar, controlar y disponer finalmente de la información.

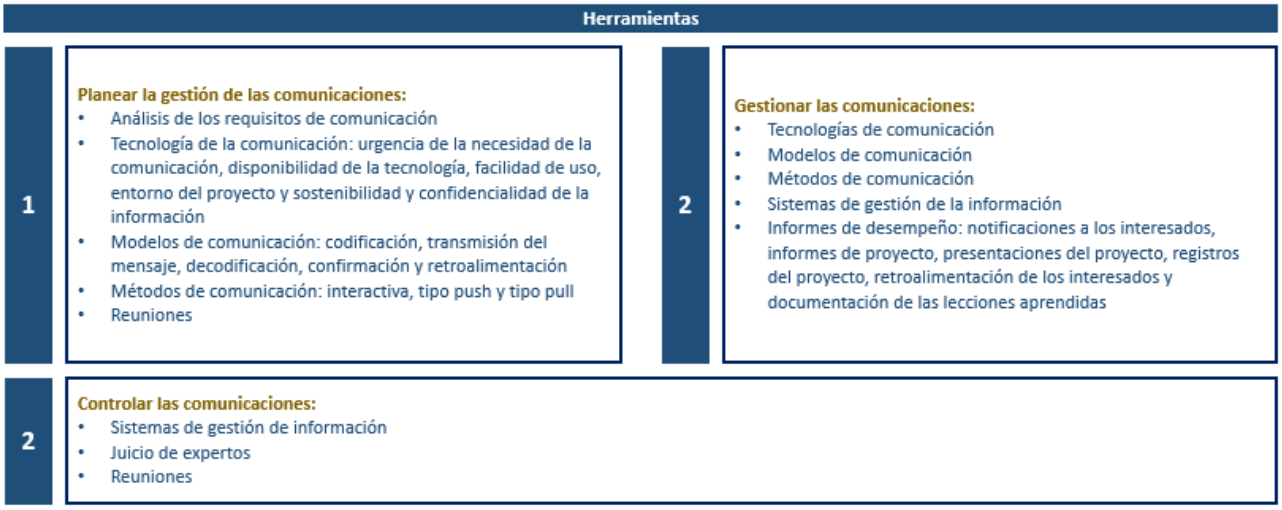
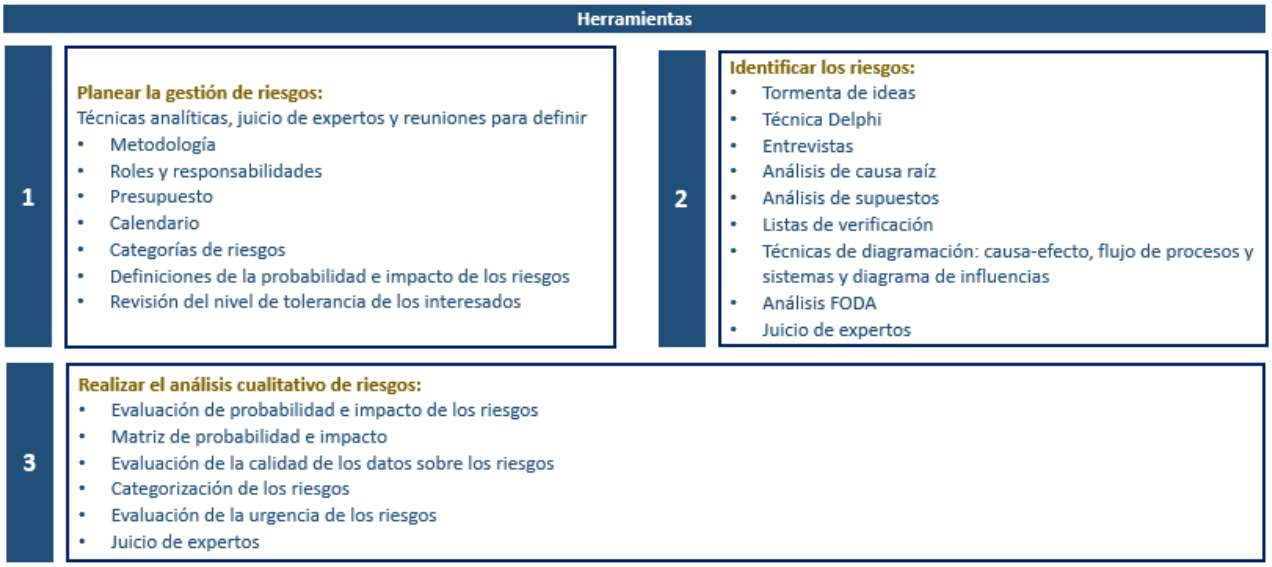


Figura 35. Herramientas para la gestión de las comunicaciones. Fuente: PMBOK Guide – 5th edition.

Plan de gestión de riegos

Describe los procesos, políticas y procedimientos asociados para planear, crear, recopilar, distribuir, almacenar, recuperar, controlar y disponer finalmente de la información.



4	**Realizar el análisis cuantitativo de riesgos:** • Técnicas de recopilación y representación de datos: Entrevistas y distribuciones de probabilidad • Técnicas de análisis cuantitativo y modelamiento: Análisis de sensibilidad, análisis de valor monetario esperado, modelamiento, simulación, análisis probabilístico y análisis de tendencias • Juicio de expertos
5	**Planear la respuesta a los riesgos:** • Estrategias para riesgos negativos o amenazas: Evitar, transferir, mitigar y aceptar • Estrategias para riesgos positivos u oportunidades: Explotar, mejorar, compartir y aceptar • Estrategias de respuesta a contingencias • Juicio de expertos
6	**Controlar los riesgos:** • Re-evaluación de riesgos • Auditorias de riesgos • Análisis de variación y de tendencias • Medición del desempeño • Análisis de reservas • Reuniones

Figura 36. Herramientas para el plan de gestión de riesgos. Fuente: PMBOK Guide – 5th edition.

Plan de gestión de adquisiciones

Describe los procesos, políticas y procedimientos asociados para comprar o adquirir productos y servicios que son necesarios para cumplir el objetivo del proyecto.

Herramientas	
1	**Planear la gestión de las adquisiciones:** • Análisis de productos o servicios que deben ser adquiridos: Comprensión de la necesidad, costo total, ciclo de vida, capacidad técnica, riesgo, enfoque de gestión, enfoque técnico, garantías, capacidad financiera, capacidad de producción, derechos de propiedad intelectual, entre otros • Juicio de expertos • Investigación de mercado • Reuniones
3	**Controlar las adquisiciones** • Sistema de control de cambios a los contratos • Revisiones de desempeño según lo pactado en el contrato • Inspecciones y auditorias • Informes de desempeño • Sistemas de pago • Penalizaciones • ANS • Gestión de inconformidades
2	**Efectuar las adquisiciones:** • Conferencias de oferentes • Técnicas de evaluación de propuestas • Estimaciones independientes • Juicio de expertos • Publicidad • Técnicas analíticas • Negociación de adquisiciones
4	**Cerrar las adquisiciones:** • Auditorias de adquisición • Negociaciones

Figura 37. Herramientas para el plan de gestión de adquisiciones. Fuente: PMBOK Guide – 5th edition.

Plan de gestión de los interesados

Describe los procesos, políticas y procedimientos asociados para identificar y gestionar a las personas, grupos u organizaciones que puedan verse afectados por el proyecto de forma directa o indirecta.

Figura 38. Herramientas para la gestión de los interesados. Fuente: PMBOK Guide – 5th edition.

3.2.1.12 Recopilación de requisitos.

Es el proceso de determinar, documentar y gestionar las necesidades de todos los actores, con el fin de cumplir los objetivos del proyecto, como se muestra en la siguiente figura:

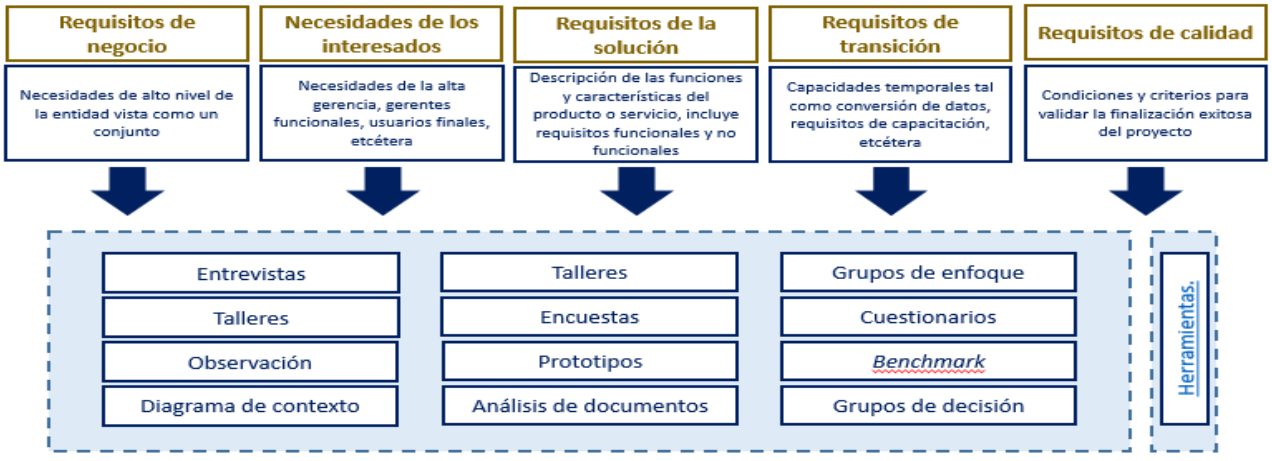


Figura 39. Recopilación de requisitos. Fuente: PMBOK Guide – 5th edition.

3.2.1.13 Definición del alcance.

En este paso la idea es desarrollar una descripción detallada del proyecto y del *software*, teniendo en cuenta:

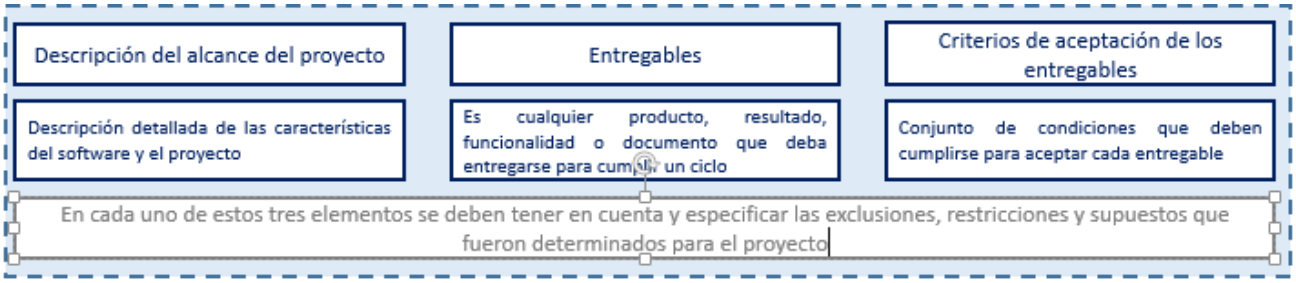


Figura 40. Descripción detallada del proyecto. Fuente: PMBOK Guide – 5th edition.

El plan de gestión del alcance es un documento que define cómo se definirá, validará y controlará dicho alcance.

Figura 41. Planificación del alcance. Fuente: Corporación Colombia Digital.

Las herramientas para crear el plan de gestión del alcance son: juicio de expertos, análisis del producto, lluvia de ideas para generar alternativas, sesiones de trabajo y casos de uso.

Mejor Práctica: *Al planificar el alcance, seguramente el plan del proyecto tendrá poco detalle, pero debería incluir como mínimo: las fases o el ciclo de vida del proyecto, qué procesos y herramientas se van a utilizar y cómo se realizará la gestión de la configuración.*

El análisis del producto se utiliza para aquellos proyectos en los que el entregable es un producto. Por otro lado, la identificación de alternativas se emplea para obtener diferentes enfoques para ejecutar y desarrollar el trabajo necesario, en esta actividad participan el gerente del proyecto, analistas y gerentes de TI.

Elemento transversal – documentación: *La documentación o entregables que debe producir la gestión del alcance son:*

- ***Objetivos del proyecto:*** *incluyen los criterios de éxito que se puedan medir. Los proyectos pueden tener una amplia variedad de objetivos de negocio, de costos, de cronograma, técnicos y de calidad.*
- ***Requisitos del proyecto:*** *Condiciones que deben cumplir o las capacidades que deben tener los productos entregables del proyecto. Se refieren a los requisitos necesarios para satisfacer un contrato, norma, especificación o cualquier otro documento formalmente impuesto.*
- ***Límites del proyecto:*** *Identifican las restricciones de tiempo, recursos y procedimientos.*
- ***Productos entregables del proyecto:*** *Incluyen las salidas que constituyen el producto o servicio resultado del proyecto, y los materiales adicionales como informes y documentación relativa a la dirección del proyecto.*
- ***Criterios de aceptación de productos:*** *Definen el proceso y los criterios para que sean aceptados los productos completados a los largo del proyecto.*
- ***Restricciones:*** *Enumeran y describen las restricciones relacionadas con el alcance que limiten las opciones del equipo.*

- ***Supuestos:*** *Enumeran y describen las suposiciones del proyecto asociadas al alcance del proyecto y al potencial impacto si resultan ser falsas.*
- ***Riesgos iniciales identificados:*** *Describen los riesgos conocidos y previstos que hay que considerar antes de comenzar el proyecto.*
- ***Hitos del cronograma:*** *La entidad ejecutante puede identificar puntos clave en el cronograma y establecer fechas para los mismos. Estos tiempos deben tratarse como restricciones del cronograma.*
- ***Requisitos de gestión de la configuración del proyecto:*** *Describen el nivel de control de cambios, identifican los documentos y elementos configurables.*
- ***Requisitos de aprobación:*** *Identifica los requisitos de aprobación de aspectos tales como objetivos, productos entregables, documentos y trabajos del proyecto.*
- ***Plan de gestión del alcance:*** *Documento en el que se definen los procedimientos que se llevarán a cabo para: preparar el enunciado o declaración del alcance, crear y aprobar la EDT, realizar la validación del alcance, procesar y aprobar los cambios en el alcance.*
- ***Descripción general del entorno tecnológico del sistema****: Documento en el que se define la arquitectura de infraestructura tecnológica, los servicios de conectividad, el servicio técnico y la mesa de ayuda requerida.*
- ***Identificación de los interesados:*** *Lista todas aquellas personas o departamentos que participan directamente o que se ven afectados por el proyecto.*

Mejor Práctica: *Defina el entorno tecnológico que se requiere para dar respuesta a las necesidades de información, especifique los posibles condicionantes y restricciones. Esta información se obtiene mediante sesiones de trabajo con los usuarios y con el apoyo de los responsables de las áreas de TI.*

3.2.1.14 Estructura de desglose del trabajo (EDT).

Es una agrupación de trabajo orientada a la entrega de los elementos del proyecto, organiza y define el alcance. Tiene los siguientes propósitos:

- Reforzar los objetivos planteados por el proyecto al identificar las actividades principales que se deben cumplir.
- Identificar las tareas clave que requieren atención, las subtareas y el flujo lógico que las relaciona.
- Crear una guía de seguimiento de los elementos que componen el proyecto.
- Comunicar la situación del proyecto en cualquier momento, a partir de la información contenida.
- Suministrar lineamientos sobre el proceso de control.
- Facilitar el proceso de delegación.

Componentes de la EDT

- Niveles de la EDT: determinan la jerarquía de trabajo dentro del proyecto, como se muestra en la siguiente figura:

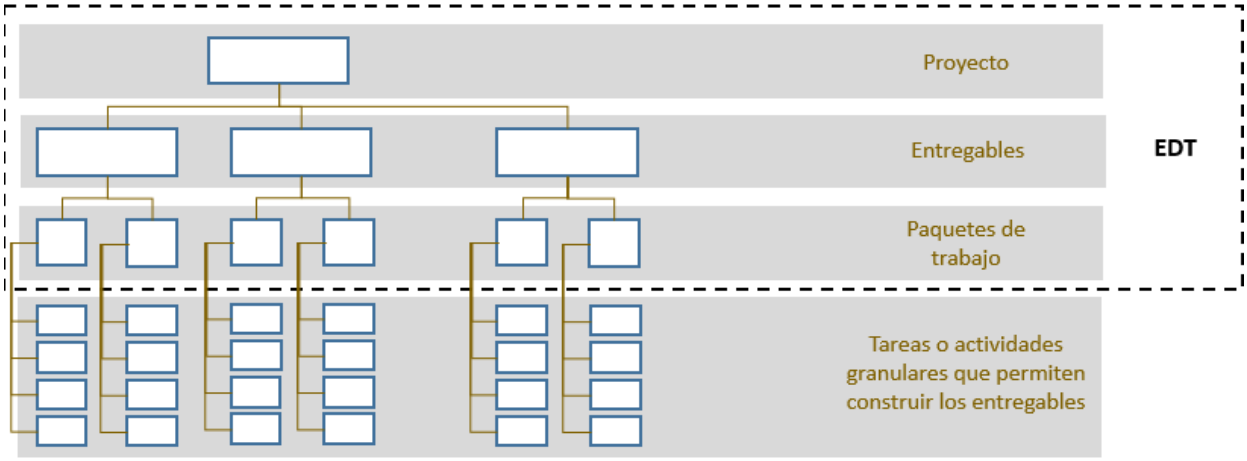


Figura 42. Niveles de la EDT. Fuente: PMBOK Guide – 5th edition.

- Diccionario de la EDT: Suministra detalles sobre los elementos especificados en la EDT, para que no haya lugar a ambigüedades. Estos son: información adicional sobre el trabajo que debe ser realizado, actividades e hitos, procedimientos para estimar los costos, recursos requeridos y la información contractual para cada elemento.

- Códigos de identificación de los elementos de la EDT: Definen una metodología de identificación y numeración de cada uno de los elementos de la EDT para facilitar la trazabilidad de cada elemento.
- Elementos de la EDT: Se refiere al proyecto, entregables, paquetes de trabajo y actividades especificadas en la EDT.
- Paquetes de trabajo de la EDT: Agrupan actividades con características similares. Se sugiere que el tamaño de cada paquete de trabajo, en términos de tiempo, esté entre 8 y 80 horas.

3.2.1.15 Línea base del alcance

Es un documento formalmente aprobado por los interesados, designados con la autoridad de aceptar las condiciones del alcance. El documento contiene la siguiente información:

Enunciado del alcance del proyecto: incluye la descripción del alcance del proyecto y del *software*, los principales entregables, los supuestos y las restricciones consideradas.
Estructura de desglose del trabajo (EDT): Descomposición jerárquica del alcance del proyecto que contiene los entregables y los paquetes de trabajo necesarios para cumplir con los resultados propuestos.
Diccionario de la EDT: Descripción detallada que proporciona información sobre los entregables, actividades, planes, supuestos y restricciones de cada uno de los componentes de la EDT.
Ajustes aprobados al alcance: Dentro de los procedimientos definidos, se considera la eventual gestión de cambios en el alcance del proyecto. Las modificaciones que sean pactadas y aprobadas según el procedimiento definido, deben ser incluidas en este documento para garantizar su trazabilidad.

3.2.1.16 Validar el alcance

Es el proceso de formalizar la aceptación de los entregables que hasta el momento se encuentren listos, como se muestra en la siguiente figura:

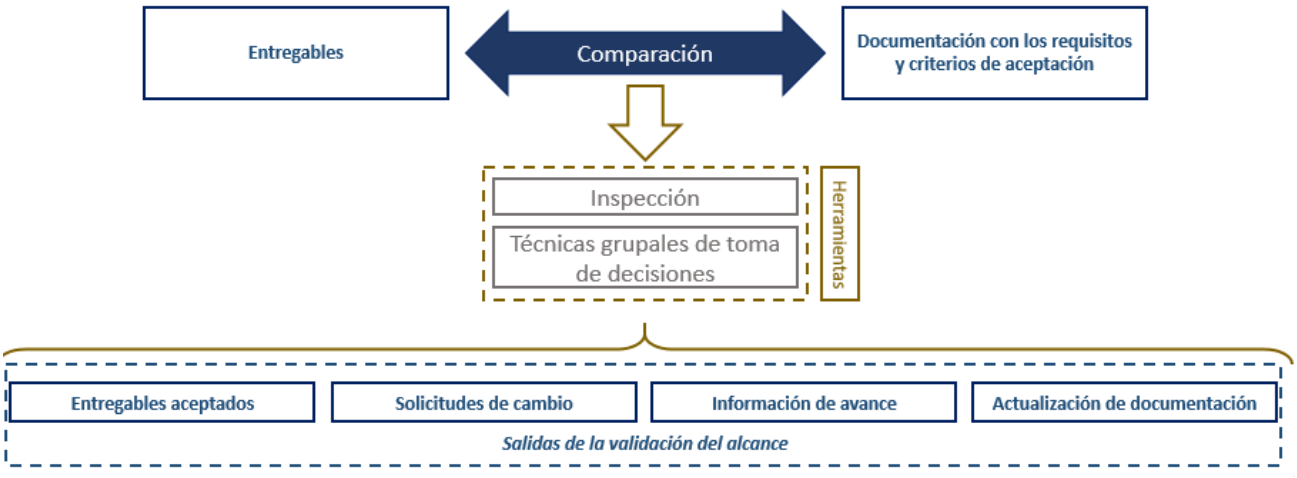


Figura 43. Proceso de validar el alcance. Fuente: PMBOK Guide – 5th edition.

Elemento transversal – Talento TI: *El personal asociado a la gestión del proyecto debe tener las siguientes capacidades:*

Las habilidades para conocer el negocio y entender claramente la necesidad real de la entidad, en función de la estrategia y de las iniciativas de negocio. En el plano táctico, se debe tener las capacidades para controlar y administrar adecuadamente los requerimientos, y para poder especificarlos con el suficiente nivel de calidad y de detalle que permitan asegurar la realización del trabajo.

El analista de negocio es un rol que se requiere con mayor frecuencia cuando se tiene que actuar como enlace entre un grupo de stakeholders (cualquier persona o grupo de personas afectados por una iniciativa de negocio en una entidad) y un gerente del proyecto, con el fin de analizar cómo reducir cuestiones complejas a un nivel más manejable.

El Gerente del Proyecto y otros miembros del equipo de dirección de proyectos serán los responsables de monitorear y controlar las actividades del proyecto durante todo el grupo de procesos. Monitorear es observar lo que está ocurriendo en el proyecto y controlar es

implementar acciones correctivas cuando algo está fuera de lo normal. (Lledo, Pablo, (2013). Director de proyectos).

Dentro del proyecto de desarrollo de sistemas de información es muy importante tener claro cómo se tomarán las decisiones. La siguiente tabla resume los estilos de toma de decisiones que se observan usualmente. Es importante definir desde un comienzo quienes serán los encargados de decidir y qué metodología será usada para esto:

Estilos de toma de decisiones
El líder de la entidad, el área o el proyecto toma la decisión sin consultarla.
El líder de la entidad, el área o el proyecto toma la decisión y la consulta con algún asesor o grupo de trabajo.
El área responsable toma la decisión votando y la mayoría gana.
El área responsable toma la decisión votando y el resultado debe ser unánime.
El área responsable discute y negocia hasta tomar una decisión.
El líder de la entidad delega la responsabilidad de tomar la decisión a una persona o grupo de personas.
El área responsable llega a una decisión pero existen otros actores que pueden vetar esa decisión.

Tabla 15. Estilos de toma de decisiones. Fuente: Wiegers & Beatty, 2013.

Elemento transversal – Supervisión: *La gerencia de proyecto tiene actividades que debe desarrollar en este elemento. Durante el proceso de dirigir y gestionar la ejecución del proyecto, el gerente y el equipo de trabajo llevarán a cabo lo acordado en el plan para la dirección del proyecto. Además, deben implementar los cambios aprobados (acciones correctivas, acciones preventivas y reparación de defectos) y revisar de manera periódica el impacto de los cambios sobre el proyecto.*

Durante la ejecución del proyecto, el gerente y su equipo, entre los que se encuentran analistas y personal experto, deben reunirse para intercambiar información, evaluar alternativas y tomar decisiones.

Mejor Práctica: *El gerente del proyecto no debería aprobar cambios, sino que él generalmente solicita cambios al comité de cambios. Ahora bien, el gerente suele tener autoridad para aprobar algunos cambios preestablecidos en la matriz de roles y responsabilidades, por ejemplo cambios de emergencia.*

Elemento transversal – documentación: *La documentación o entregables que debe producir el gerente del proyecto son:*

- ***Plan detallado de trabajo (PDT):*** *lista de actividades y su correspondiente detalle, enfocadas en cumplir con objetivos en cada etapa, además con los tiempos y recursos estimados.*
- ***Matriz de riesgos:*** *Identifica los riesgos y los valora por probabilidad e impacto para identificar el plan de respuesta.*
- ***Plan de gestión de los interesados:*** *Identifica las estrategias de gestión necesarias para involucrar a los interesados de manera eficaz.*

Elemento transversal – Talento TI: *El personal responsable de las actividades en la definición del alcance es el siguiente:*

- ***Gerente de Proyecto:*** *estima el esfuerzo necesario para llevar a cabo el proyecto, selecciona la estrategia de desarrollo, determina la estructura con los procesos principales del proyecto, fija el calendario de hitos y entregas, y establece la planificación. Es el encargado de dirigir el proyecto y para eso realiza las labores de seguimiento y control, revisa y evalúa los resultados y coordina el equipo. Se ocupa también de la gestión y resolución de incidencias que puedan surgir durante el desarrollo del proyecto, así como de la actualización de la planificación inicial. Entre sus funciones también se encuentra elaborar los informes de seguimiento y el archivo de la documentación de gestión del proyecto una vez que éste ha finalizado.*

- ***Expertos:*** *El perfil requerido para este grupo de participantes incluye a personas con un nivel alto en la dirección de la organización, de conocimiento de los objetivos estratégicos y de negocio que se persiguen, y de autoridad para validar y aprobar cada uno de los procesos realizados durante el desarrollo del sistema de información. Los expertos deben tener un conocimiento del entorno y de la organización para proporcionar, a lo largo del proyecto, unos requisitos del sistema adecuados, completos y suficientes para considerarlos en el catálogo definitivo de requisitos.*

  Los directores de las áreas funcionales y los usuarios afectados por el proyecto aportan información sobre las necesidades planteadas y validan los resultados con el fin de garantizar la identificación, comprensión e incorporación de todos los requisitos con las prioridades adecuadas. Esta misma función la desempeñan con mayor nivel de detalle los usuarios expertos de nivel directivo.

  El seguimiento y control del desarrollo del proyecto es responsabilidad del comité de seguimiento, que se ocupa de resolver cualquier contingencia durante la ejecución y asegura la disponibilidad de recursos humanos con los perfiles adecuados, además de participar en las actividades en las que sea necesaria su colaboración.

- ***Arquitecto de software:*** *En general, en él recae la responsabilidad de traducir las necesidades de la entidad, hacia una solución técnica preliminar, que es una pieza clave para producir una estimación del esfuerzo necesario para el desarrollo. El arquitecto puede participar en el trabajo de estimación del sistema, durante esta etapa debe hacer uso de habilidades técnicas (“duras”) y no técnicas (“suaves”). Como parte de las habilidades técnicas debe poder identificar estilos arquitectónicos y tecnologías que sean apropiadas para resolver el problema y proponer una solución preliminar. Como parte de las habilidades no técnicas debe ser capaz de analizar las necesidades de la entidad, especialmente desde una perspectiva de negocio, y poder explicar la solución técnica que propone a los distintos involucrados del proyecto.*

3.2.1.17 Control del alcance

Es el proceso de monitorear el avance del proyecto dentro de los parámetros establecidos en el alcance y gestionar cambios en la línea base del alcance, como se muestra en la siguiente figura:

Figura 44. Modelo del proceso de cambios. Fuente: PMBOK Guide – 5th edition.

Revisión del desempeño del proyecto

Comparación del desempeño del proyecto con respecto a la línea base. El reporte que se genera al aplicar este control debe incluir:

- Las variaciones detectadas categorizadas según la(s) etapas del ciclo de vida que afectan.
- Las causas de cada variación detectada.
- El plan de acciones correctivas o de adaptación a implementar.
- El impacto en el cronograma que tiene cada variación detectada.
- El impacto en el costo que tiene cada variación detectada.
- El pronóstico de desempeño futuro del proyecto

Solicitud de cambio

En este proceso se identifican dos tipos de cambios: mayores y menores

Mayores: Afectan los requisitos o actividades que se encuentran en la ruta crítica, tienen un impacto significativo en el proyecto y requieren recursos adicionales.

Menores: No tiene impacto en la ruta crítica y las actividades que afectan o las dependencias no generan mayor impacto en el desarrollo del proyecto. No requieren recursos adicionales.

La información requerida en una solicitud de cambio es: Descripción del cambio, motivo del cambio, impacto en el alcance, estimación de los recursos y el trabajo requerido, estimación del costo adicional, estimación del tiempo requerido y el cronograma, riesgos preliminares detectados y especificaciones de calidad que afectan los cambios.

Roles y responsabilidades en la gestión de cambios

En la siguiente figura se identifican los roles y responsabilidades de las personas involucradas en el proceso de gestión de cambios.

1	Gerente de proyecto.	• Debe facilitar el proceso de gestión de cambios. • Debe participar en la ejecución de gestión de cambios. • Debe participar en la evaluación de nuevos requisitos, alcance, cronograma y recursos. • Debe participar en el proceso de autorización de los cambios propuestos.
2	Patrocinador del proyecto.	• Debe revisar los cambios propuestos. • Debe participar en el proceso de autorización de los cambios propuestos. • Debe garantizar que los recursos que requieren los cambios aprobados estén disponibles.
3	Coordinador de cambios.	• Debe participar en el proceso de autorización de los cambios propuestos. • Debe documentar y dar seguimiento a los cambios propuestos. • Debe participar en la ejecución de gestión de cambios.
4	Comité asesor de cambios.	• Debe proveer un concepto para oponerse o apoyar los cambios de acuerdo a criterios expertos.
5	Comité de control de cambios.	• Esta compuesto por varios interesados. En particular deben integrarlo el gerente de proyecto, el patrocinador, el coordinador de cambios y cualquier otro actor que sea relevante en el proceso de toma de decisiones. • Debe aprobar o rechazar los cambios propuestos.

Figura 45. Roles y responsabilidades en la gestión de cambios. Fuente: PMBOK Guide – 5th edition.

Análisis de variación

Estudia si los desvíos en el alcance comparados con la línea base son significativos como para aplicar acciones correctivas. Las herramientas utilizadas se muestran en la siguiente figura:

Figura 46. Herramientas para análisis de variación. Fuente: PMBOK Guide – 5th edition.

Actualización de los documentos y planes

Los documentos que se actualizan en el proceso son el registro de cambios y línea base del alcance.

Los planes a actualizar son: Plan para la dirección del proyecto, gestión del alcance, gestión de los requisitos, gestión del cronograma, gestión de los costos, gestión de la calidad, mejoras de proceso, gestión de los recursos humanos, gestión de comunicaciones, gestión de riesgos, gestión de adquisiciones y gestión de los interesados.

3.3 GESTION DE LA CALIDAD

La gestión de la calidad es el conjunto de actividades que determinan la calidad, los objetivos y las responsabilidades del proyecto. Se implanta a través de mecanismos de planificación, control, aseguramiento y mejora, en el marco del sistema de calidad (Inteco, 2009. Guía de mejores prácticas de calidad de producto).

Estas actividades están relacionadas con el aseguramiento para que un producto consiga el nivel de calidad requerido. Incluyen la definición adecuada de estándares y procesos de calidad y el cumplimiento de los mismos. Estas acciones deberían estar dirigidas a desarrollar una cultura de calidad que otorgue responsabilidades a todo el mundo.

Entre las actividades que se llevan a cabo en la gestión de la calidad están el aseguramiento, la planificación y el control, entre otras.

Hay cuatro elementos que influyen de manera importante sobre la calidad:

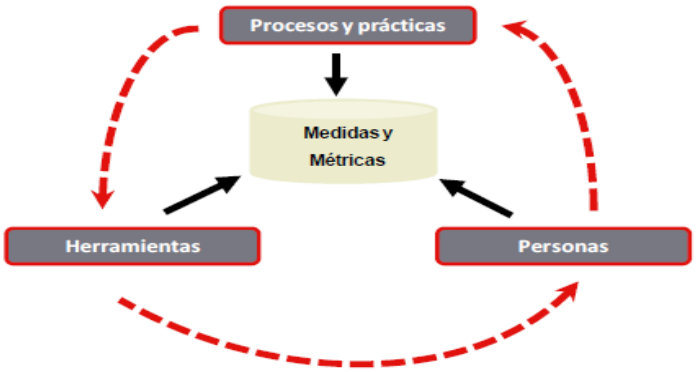


Figura 47. Elementos influyentes sobre la calidad. Fuente: Inteco. (2009). Guía de mejores prácticas de calidad de producto.

Los elementos clave relacionados con la calidad son:

Procesos y buenas prácticas: La calidad aumenta al aplicar una serie de procesos o metodologías y buenas prácticas, controlan el proceso para analizarlo y mejorarlo.
Herramientas: Proporcionan apoyo a la gestión de la calidad.
Personas: Son elementos clave como creadores y ejecutores.
Medidas y métricas: Son los datos que permiten evaluar el estado actual y ejecutar acciones para mejorar.

La calidad en la ingeniería del *software* depende en gran medida de la pericia del equipo que desarrolla, se debe tener en cuenta un conjunto de características o cualidades, tales como: eficiencia, fiabilidad, usabilidad, funcionalidad, mantenibilidad, portabilidad, etc. La importancia de cada una de ellas varía de un producto a otro.

La calidad de un producto *software* puede medirse en diferentes aspectos:

- Interna: calidad medible a partir de las características intrínsecas, como el código fuente.
- Externa: calidad medible en el comportamiento del producto, como en una prueba.
- En uso: durante la utilización efectiva por parte del usuario.

Para poder ofrecer un producto *software* de calidad, las entidades deben hacer hincapié en todo el ciclo de vida del producto, tal como lo muestra la siguiente figura:

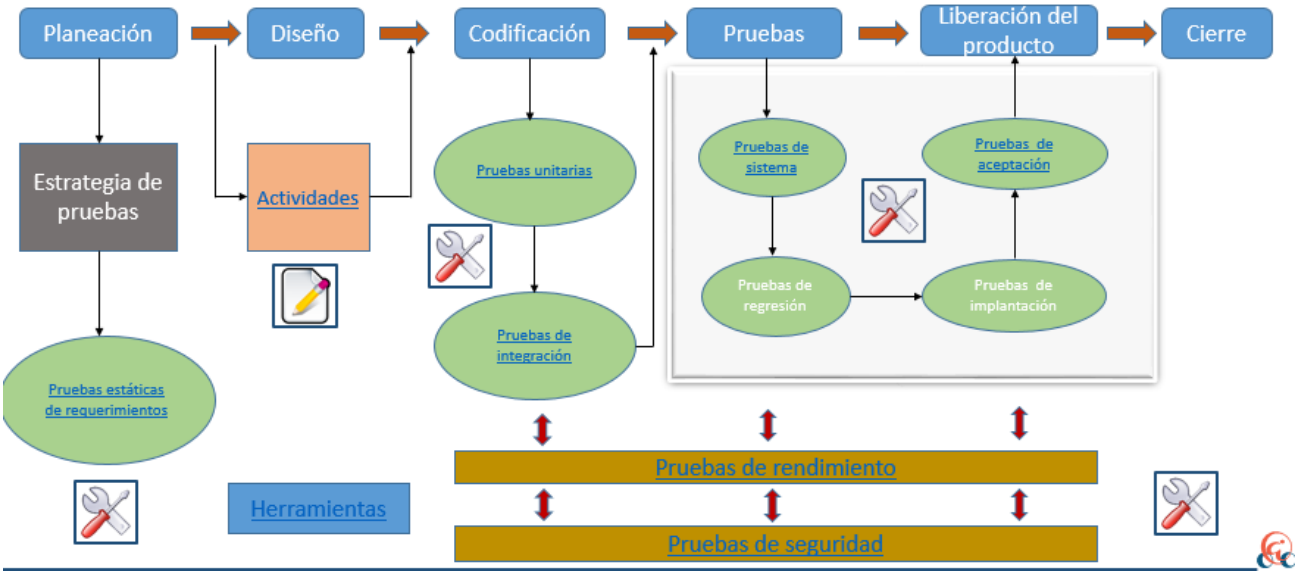


Figura 48. Ciclo de vida de la calidad del producto. Fuente: Corporación Colombia Digital.

El proceso debe empezar con un buen entendimiento y recolección de los requisitos, hasta llegar a pruebas de aceptación del usuario una vez que el *software* está desarrollado.

Mejor Práctica: *La gestión de los defectos es un proceso que se ejecuta durante todo el ciclo de vida. Se debe hacer hincapié en la importancia de detectar estos defectos en etapas tempranas para que el costo de su corrección sea menor.*

3.3.1 Aseguramiento de la calidad vs Control de calidad

El control de la calidad (QC) y el aseguramiento de la calidad (QA) son dos conceptos vitales para la gestión de la calidad que no siempre están bien definidos y que no se diferencian con claridad.

Al hablar del control de la calidad (QC) se hace referencia al conjunto de técnicas operativas y actividades utilizadas para cumplir con las demandas o requisitos de la calidad.

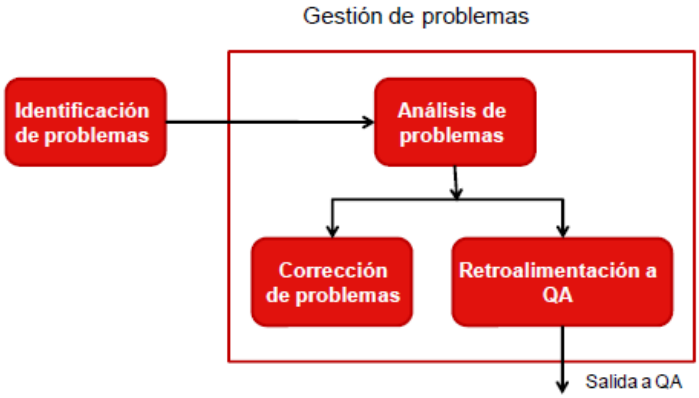


Figura 49. Pasos típicos del QC. Fuente: Inteco. (2009). Guía de mejores prácticas de calidad de producto.

Por otra parte, el aseguramiento de la calidad (QA) se entiende como el conjunto de actividades planificadas y sistemáticas necesarias para proporcionar la confianza suficiente de que un producto o servicio satisfará los requisitos de calidad.

Figura 50. Pasos típicos del QA. Fuente: Inteco. (2009). Guía de mejores prácticas de calidad de producto.

En el cuadro que se muestra a continuación aparece una comparación entre ambos conceptos y algunas actividades que se pueden llevar a cabo con base en ellos:

QA – Aseguramiento de la calidad	QC – Control de la calidad
• Preventivo y proactivo • Orientado a proceso • Responsabilidad a nivel de la organización • Identifica las debilidades de ciertos procesos y las mejora • Evalúa si QC funciona o no	• Reactivo • Orientado a producto o servicio • Responsabilidad a nivel del equipo de control • Verifica si los atributos especificados están presentes en el producto o no
ACTIVIDADES	
• Auditorías de procesos • Definiciones de procesos • Selección de herramientas • Formación	• Revisiones • Inspecciones • Ejecución de pruebas

Tabla 16. Comparativa QA vs QC. Fuente: Inteco. (2009). Guía de mejores prácticas de calidad de producto.

La entidad debe establecer los lineamientos para asegurar los niveles de calidad requeridos, técnicos y funcionales, para el desarrollo de las aplicaciones/*software*. Para esto se deben tener en cuenta las metodologías y estándares de desarrollo, de interoperabilidad y de integración, tales como CMMI, arquitectura basada en servicios, metodologías de desarrollo ágil o extremo, normas legales, etcétera.

3.3.2 Normas

A continuación se enuncia la familia de normas ISO/IEC 25000 que contiene las características de calidad para los desarrollos de *software*.

Es importante anotar que estas normas son solo una guía que puede cambiar de acuerdo con las condiciones particulares de cada proyecto, por lo tanto la implementación de este modelo debe ser evaluada por el lector.

ISO/IEC 25000, conocida como System and Software Quality Requirements and Evaluation (SQuaRE), es una familia de normas que tiene por objetivo la creación de un marco de trabajo común para evaluar la calidad del producto *software*. La familia ISO/IEC 25000 es el resultado de la evolución de otras normas anteriores, especialmente de las normas ISO/IEC 9126, que describe las particularidades de un modelo de calidad del producto *software*, e ISO/IEC 14598, que abordaba el proceso de evaluación de productos *software*. La ISO/IEC 25000 tiene cinco divisiones (http://iso25000.com/):

- **ISO/IEC 2500n – Gestión de Calidad:** definen todos los modelos, términos y definiciones comunes referenciados por todas las otras normas de la familia 25000.
- **ISO/IEC 2501n – Modelo de Calidad:** presentan modelos de calidad detallados que incluyen características sobre la calidad interna, externa y en uso del producto *software*.
- **ISO/IEC 2502n – Medición de Calidad:** incluyen un modelo de referencia de la medición de la calidad, definiciones de esas medidas (interna, externa y en uso) y guías prácticas para su aplicación.
- **ISO/IEC 2503n – Requisitos de Calidad:** estas normas ayudan a especificar requisitos de calidad que pueden ser utilizados en el levantamiento de requisitos del producto *software* que se desarrollará o como entrada del proceso de evaluación.
- **ISO/IEC 2504n – Evaluación de Calidad:** incluye normas que proporcionan requisitos, recomendaciones y guías para llevar a cabo el proceso de evaluación del producto *software*.

La calidad del producto *software* se puede interpretar como el grado en que dicho producto satisface los requisitos de los usuarios, lo que aporta valor a la entidad. Son precisamente

estos requisitos (funcionalidad, rendimiento, seguridad, mantenibilidad, etc.) los que se representan en el modelo de calidad, el cual categoriza la calidad del producto en características y subcaracterísticas.

El modelo de calidad del producto definido por la ISO/IEC 25010 se encuentra compuesto por las ocho características de calidad que se muestran en la siguiente figura:

Figura 51. Características de calidad. Fuente: http://iso25000.com/.

Adecuación funcional: representa la capacidad del producto *software* para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas. Esta característica se subdivide en:

- Completitud funcional: grado en que el conjunto de funcionalidades cubre todas las tareas y los objetivos especificados del usuario.
- Corrección funcional: capacidad del producto o sistema de proveer resultados correctos con el nivel de precisión requerido.
- Pertinencia funcional: capacidad del producto *software* de proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario específico.

Eficiencia de desempeño: esta característica representa el desempeño relativo del producto frente a la cantidad de recursos utilizados bajo determinadas condiciones. Se subdivide a su vez en:

- Comportamiento temporal: determina los tiempos de respuesta, procesamiento y rendimiento de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas, en relación con un banco de pruebas establecido (*benchmark*).
- Utilización de recursos: se refiere a las cantidades y tipos de recursos utilizados cuando el *software* lleva a cabo su función bajo condiciones determinadas.
- Capacidad: grado en que los parámetros de un producto o sistema *software* cumplen con los requisitos al ser llevados a condiciones extremas.

Compatibilidad: Capacidad de dos o más sistemas o componentes de intercambiar información y llevar a cabo las funciones requeridas cuando comparten el mismo entorno *hardware* o *software*. Esta característica se subdivide en:

- Coexistencia: capacidad de coexistir con otro *software* independiente, en un entorno común, compartiendo recursos comunes sin detrimento del intercambio de información.
- Interoperabilidad: capacidad de dos o más sistemas o componentes de intercambiar información y utilizarla.

Usabilidad: capacidad del producto *software* de ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se emplea bajo determinadas condiciones. Esta característica se subdivide en:

- Capacidad de reconocer su adecuación: permite al usuario entender si el *software* es adecuado a sus necesidades.
- Capacidad de aprendizaje: permite al usuario comprender su uso y aplicación.
- Capacidad para ser usado: facilita al usuario operarlo y controlarlo con facilidad.
- Protección contra errores de usuario: capacidad del sistema de protegerse de equivocaciones de los usuarios.
- Amigabilidad de la interfaz de usuario: capacidad de la interfaz de usuario de agradar y satisfacer la interacción con el usuario.
- Accesibilidad: capacidad del producto que le permite ser utilizado por usuarios con determinadas características y discapacidades.

Fiabilidad: hace referencia a la capacidad de un sistema o componente de desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo determinados. Esta característica se subdivide a su vez en:

- Madurez: capacidad del sistema de satisfacer las necesidades de fiabilidad en condiciones normales.
- Disponibilidad: capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiera.
- Tolerancia a fallos: capacidad del sistema o componente de operar según lo previsto, en presencia de fallos de *hardware* o *software* (continuidad del negocio).
- Capacidad de recuperación: permite recuperar los datos directamente afectados y reestablecer el sistema a una condición que permita la continuidad de la operación, en caso de interrupción o fallo.

Seguridad: capacidad de proteger la información y los datos para que personas o sistemas no autorizados no puedan leerlos o modificarlos, permite prevenir equivocaciones imprevistas. Esta característica se subdivide en:

- Confidencialidad: capacidad de proteger contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente.
- Integridad: capacidad del sistema o componente de prevenir accesos o modificaciones no autorizados a datos o programas.
- Responsabilidad: capacidad de rastrear de forma inequívoca las acciones de una entidad.
- Autenticidad: capacidad de demostrar la identidad de un sujeto o un recurso.

Mantenibilidad: esta característica representa la capacidad del producto *software* de ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas. Esta característica se subdivide en:

- Modularidad: capacidad de un sistema o programa que permite que un cambio en un componente tenga un impacto mínimo en los demás.
- Reusabilidad: capacidad de un activo que le permite ser utilizado en más de un sistema *software* o en la construcción de otros activos.
- Análisis: facilidad con la que se puede evaluar el impacto de un determinado cambio sobre la totalidad del *software*, diagnosticar las deficiencias o causas de fallos e identificar las partes para modificar.
- Capacidad para ser modificado: le permite ser modificado de forma efectiva y eficiente, sin introducir defectos o degradar el desempeño.
- Capacidad para ser probado: facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.

Portabilidad: capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno *hardware*, *software*, operacional o de utilización a otro. Esta característica se subdivide en:

- Adaptabilidad: capacidad del producto de ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de *hardware*, *software*, operacionales o de uso.
- Capacidad para ser instalado: facilidad con la que el producto se puede instalar y desinstalar de forma exitosa en un determinado entorno.
- Capacidad para ser reemplazado: le permite ser utilizado en lugar de otro producto *software* con el mismo propósito y en el mismo entorno.

3.4 METODOLOGÍAS PARA EL DESARROLLO DE SOFTWARE

Una metodología para desarrollo de *software* se refiere a un entorno o ambiente de trabajo de desarrollo que es usado para estructurar, planear y controlar el proceso de desarrollo en sistemas de información. Una gran variedad de estos marcos de trabajo ha evolucionado durante los últimos años; sin embargo, una metodología no tiene que ser necesariamente

adecuada para todos los proyectos. Cada una de las metodologías disponibles tiene ventajas y desventajas que debe adecuarse al tipo específico.

Desarrollar un buen *software* depende de un sinnúmero de actividades y etapas, en las que el impacto de elegir la mejor metodología para un equipo de trabajo, en un determinado proyecto, es trascendental para el éxito del producto. Elegir la metodología es el paso inicial para guiar y organizar actividades que conlleven a las metas trazadas por el grupo.

Son muchas las ventajas que puede aportar el uso de una metodología. A continuación se exponen algunas de ellas, clasificadas desde distintos puntos de vista:

Vista desde la gestión:

- Facilitar la tarea de planificación
- Facilitar la tarea del control y seguimiento de un proyecto
- Mejorar la relación costo/beneficio
- Optimizar el uso de recursos disponibles
- Facilitar la evaluación de resultados y el cumplimiento de los objetivos
- Facilitar la comunicación efectiva entre usuarios y desarrolladores

Desde el punto de vista de la ingeniería de *software*:

- Ayudar a la comprensión del problema
- Optimizar el proceso y las fases de desarrollo
- Facilitar el mantenimiento del producto final
- Permitir la reutilización de partes del producto

Vista del cliente o usuario:

- Garantizar un determinado nivel de calidad en el producto final

- Generar confianza en los tiempos fijados en la definición del proyecto
- Definir el ciclo de vida que más se adecue a las condiciones y características del desarrollo

Mejor Práctica: *desarrollar un buen software depende de un gran número de actividades y etapas, en las que elegir la metodología para un equipo de trabajo en un determinado proyecto es trascendental para el éxito del producto.*

Según la filosofía de desarrollo, se pueden clasificar las metodologías en dos grupos: las metodologías tradicionales, que se basan en una fuerte planificación durante todo el desarrollo, y las metodologías ágiles, en las que el desarrollo de *software* es incremental, cooperativo, sencillo y adaptado.

3.4.1 Metodologías tradicionales

Las metodologías tradicionales son orientadas por planeación. Inician el desarrollo de un proyecto con un riguroso levantamiento de requerimientos, previo a etapas de análisis y diseño. Con esto tratan de asegurar resultados circunscritos a un calendario o cronograma con alta calidad.

En las metodologías tradicionales se concibe un solo proyecto, de grandes dimensiones y estructura definida. Se sigue un proceso secuencial en una sola dirección y sin marcha atrás, que además es rígido y no cambia. Los requerimientos son acordados de una vez y para todo el proyecto, demanda largos plazos de planeación previa y poca comunicación con el cliente una vez ha terminado ésta.

Estas metodologías imponen una disciplina de trabajo con el fin de conseguir un *software* más eficiente, para ello se hace énfasis en la planeación total del trabajo, tras lo que se inicia el ciclo de desarrollo. Las metodologías tradicionales se centran especialmente en el

control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Otra de las características importantes dentro de este enfoque son los altos costos al implementar un cambio y la falta de flexibilidad para entornos volátiles, pues no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno donde los requisitos no pueden predecirse o pueden variar.

Ventajas:

- Se evalúa cada fase, lo que permite realizar cambios de objetivos
- Funciona bien en proyectos de innovación
- Es sencilla ya que sigue los pasos intuitivos necesarios para desarrollar el *software*
- Hace seguimiento detallado en cada una de las fases

Desventajas:

- La evaluación de riesgos es compleja
- Hay una excesiva flexibilidad para algunos proyectos
- El cliente debe ser capaz de describir y entender a un gran nivel de detalle para poder acordar un alcance del proyecto con él

La metodología tradicional más utilizada es la *Rational Unified Process* - Proceso de Desarrollo Unificado (RUP), la cual se describe a continuación:

El RUP es un marco de trabajo iterativo creado en 2003 por la *Rational Software Corporation*, una división de IBM. No es un proceso preceptivo concreto individual, sino un marco de trabajo de proceso adaptable, con la idea de ser utilizado por las organizaciones de desarrollo y los equipos de proyecto de *software* que seleccionarán los elementos del proceso que sean apropiados para sus necesidades.

El RUP está dirigido por casos de uso, centrado en la arquitectura, es iterativo e incremental. Se basa en un conjunto de módulos o elementos de contenido que describen lo que se producirá, las habilidades requeridas y la explicación paso a paso de cómo se consiguen los objetivos de desarrollo. Los módulos principales o elementos de contenido son:

- Roles (quién): definen un conjunto de habilidades, competencias y responsabilidades relacionadas.
- Productos de trabajo (qué): representan el resultado de una tarea, incluyen todos los documentos y modelos producidos durante el proceso.
- Tareas (cómo): describen una unidad de trabajo asignada a un rol que proporciona un resultado significante.

El RUP determina cuatro fases para el ciclo de vida, que permiten presentar el proceso en un alto nivel de forma similar a como sería presentado un proyecto basado en un estilo en cascada, aunque en esencia la clave del proceso recae en las iteraciones de desarrollo dentro de las fases. Además, cada fase tiene un objetivo clave y un hito al final que expresa el logro de ese objetivo.

Las cuatro fases en las que divide el ciclo de vida del proyecto son:

INICIO	ELABORACION	CONSTRUCCION	TRANSICION

Figura 52. Fases del ciclo de vida RUP. Fuente: Corporación Colombia Digital.

Fase de inicio: se busca ayudar al equipo de proyecto a decidir cuáles son los verdaderos objetivos del proyecto. Las iteraciones exploran diferentes soluciones y arquitecturas posibles. Es la fase más corta del proyecto, en la que se describe el producto final, se presenta el análisis del negocio y se identifican y priorizan los riesgos más importantes. La

fase finaliza con el hito de objetivos del ciclo de vida, el cual se alcanza cuando el equipo de proyectos y los interesados llegan a un acuerdo sobre:

- Cuál es el conjunto de necesidades del negocio y qué grupo de funciones las satisfacen
- La planificación preliminar de iteraciones
- La arquitectura preliminar

Fase de elaboración: se analizan con mayor detalle las necesidades del negocio, se definen sus principios arquitectónicos, se captura la mayoría de los requisitos del sistema y se identifican los riesgos. La fase de elaboración finaliza con el hito de la arquitectura del ciclo de vida, que se alcanza cuando el equipo de desarrollo y los interesados llegan a un acuerdo sobre:

- Los casos de uso que describen la funcionalidad del sistema
- La línea base de la arquitectura
- La mitigación de los mayores riesgos
- El plan del proyecto

Fase de construcción: es la fase más larga del proyecto, en ella se crea el diseño de la aplicación y el código fuente. Esta fase finaliza con el hito de capacidad operativa inicial, el cual se alcanza cuando el equipo de desarrollo y los interesados llegan a un acuerdo sobre:

- La estabilidad del producto para ser usado
- El producto provee alguna funcionalidad de valor
- El inicio de la transición para que todas las partes estén listas

Fase de transición: se entrega el sistema a los usuarios y se entrenan. En este momento el equipo se encuentra ocupado fundamentalmente en corregir y extender la funcionalidad del sistema desarrollado en la fase anterior. La fase de transición finaliza con el hito del lanzamiento del producto, el cual se alcanza cuando el equipo de desarrollo y los interesados llegan a un acuerdo sobre:

- El cumplimiento de los objetivos fijados en la fase de inicio
- La satisfacción del usuario

3.4.2 Metodologías ágiles

En los años noventa surgieron metodologías ligeras de desarrollo de *software*, luego las llamaron ágiles. Están dirigidas a reducir la probabilidad de fracaso de los proyectos por subestimación de costos, tiempos y funcionalidades. Se gestaron como alternativa a las metodologías tradicionales, específicamente para reducir la carga burocrática en proyectos de pequeña y mediana escala. A diferencia de las tradicionales, las metodologías ágiles son adaptativas no predictivas y están orientadas a las personas, no a los procesos. Por su flexibilidad, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto.

Los proyectos ágiles se subdividen en proyectos más pequeños mediante una lista ordenada de características. Cada proyecto es tratado de manera independiente y desarrolla un subconjunto de características durante un periodo corto, de entre dos y seis semanas. La comunicación con el cliente es constante, al punto de requerir un representante de él durante el desarrollo. Este tipo de proyectos es altamente colaborativo y se adapta mejor a los cambios, de hecho, el cambio en los requerimientos es una característica esperada y deseada, al igual que las entregas constantes al cliente y su retroalimentación. Tanto el producto como el proceso son mejorados frecuentemente.

Las metodologías ágiles se caracterizan por el desarrollo iterativo e incremental, la simplicidad de la implementación, las entregas frecuentes, la priorización de los requerimientos o características a cargo del cliente, y la cooperación entre desarrolladores y usuarios. Estas metodologías dan como un hecho que los requerimientos van a cambiar durante el proceso de desarrollo.

Mejor Práctica: *hoy por hoy, más del 90% de las iniciativas son desplegadas bajo Scrum. Recomendamos utilizar metodologías ágiles porque, con la experiencia, son más productivas, menos riesgosas y más económicas que otras aproximaciones "tradicionales". Como lo dice Jeff Sutherland, uno de los padres del movimiento ágil, "Scrum busca hacer el doble del trabajo en la mitad del tiempo".*

Las metodologías que buscan ser predictivas como las RUP o 'en cascada', las metodologías ágiles (Scrum o XP) no pretenden conceptualizar ni diseñar la totalidad de un *software*, previo a comenzar su construcción o codificación. Por el contrario, reconocen que un *software* es un emprendimiento flexible, que durante su ciclo de construcción sufrirá cambios, modificaciones y mejoras. En consecuencia, estas metodologías generan un ciclo de vida que permite incorporar nuevas ideas en cualquier momento, sin sacrificar la productividad o el tiempo de entrega del proyecto.

En esta misma línea, las aproximaciones ágiles apuntan a desarrollar *software* de manera iterativa e incremental, entregándole al cliente un producto ejecutable cada mes, el cual puede ser explorado y mejorado por el cliente. Como cada iteración (o Sprint) incorpora nuevos conocimientos, despliega un proceso de constante aprendizaje y refinamiento del producto final.

Adicionalmente, como se entrega *software* de manera temprana, el cliente puede comenzar a utilizar la herramienta y acelerar así el retorno sobre la inversión. Esta aproximación mantiene la moral del equipo en alto, puesto que el producto se logra hacer tangible y utilizar desde un principio, distinto a tantos desarrollos en los que después de meses lo único que se tiene son documentos técnicos en papel.

Para que el desarrollo ágil funcione, es necesario acompañar la etapa de codificación con prácticas de calidad (QA) tales como el *Test Driven Development* o TDD (desarrollo

orientado a pruebas) y la integración continua. Ambas garantizan que el proceso de control de calidad de una aplicación se lleve a cabo de manera paralela (y constante) al proceso de desarrollo o codificación, para que el código de cada *sprint* pueda ser probado por el cliente.

La metodología ágil más utilizada es Scrum: Es un marco de trabajo diseñado para lograr la colaboración eficaz de equipos en proyectos, emplea un conjunto de reglas y artefactos, y define roles que generan la estructura necesaria para su correcto funcionamiento.

La Scrum utiliza un enfoque incremental que tiene como fundamento la teoría de control empírico de procesos. Esta teoría se basa en la transparencia, inspección y adaptación; la transparencia que garantiza la visibilidad en el proceso de las cosas que pueden afectar el resultado; la inspección que ayuda a detectar variaciones indeseables en el proceso; y la adaptación que realiza los ajustes pertinentes para minimizar el impacto de las mismas

Es un proceso de desarrollo iterativo e incremental (o creciente) para la gestión y el desarrollo de proyectos de *software* para equipos pequeños (entre 3 y 9 personas) y con ciclos de entrega cortos, máximo de 4 semanas, como lo muestra la siguiente figura:

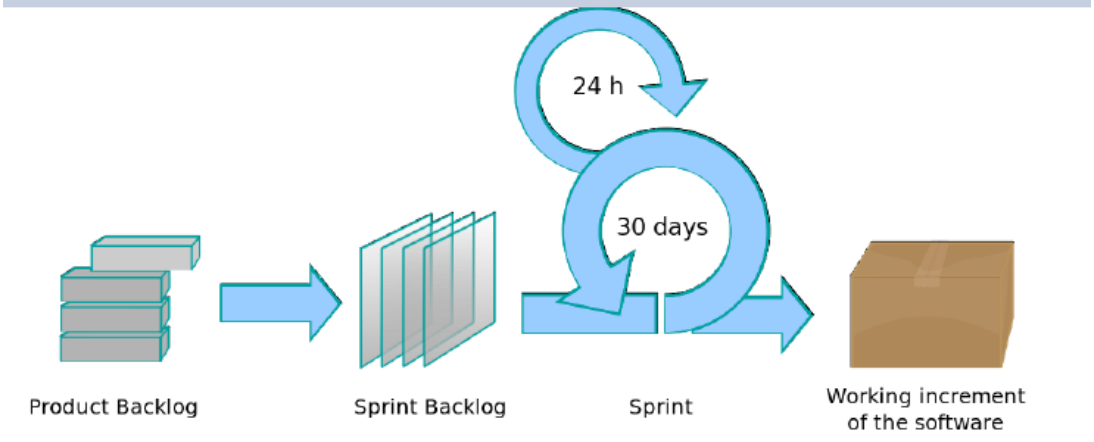


Figura 53. Proceso Scrum. Fuente: www.fattocs.com.

La Scrum se focaliza en priorizar el trabajo en función del valor que tenga para el negocio, lo que maximiza la utilidad de lo que se construye y el retorno de la inversión. Está diseñada especialmente para adaptarse a los cambios en los requerimientos, por ejemplo en un mercado de alta competitividad. Las necesidades y prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares, en tiempo real, según las necesidades del cliente. Se busca entregar *software* que realmente resuelva las necesidades y aumenten la satisfacción del cliente.

En Scrum, el equipo se focaliza en una única cosa: construir *software* de calidad. Por otro lado, la gestión se centra en definir cuáles son las características que debe tener el producto (qué construir, qué no y en qué orden) y en remover cualquier obstáculo que pueda entorpecer la tarea del equipo de desarrollo. Se busca que los equipos sean lo más efectivos y productivos posible.

Scrum tiene un conjunto de reglas muy pequeño y simple, está basado en los principios de inspección continua, adaptación, autogestión e innovación. Como resultado, el cliente se entusiasma y se compromete con el proyecto, dado que ve crecer el producto iteración a iteración y encuentra las herramientas para alinear el desarrollo con los objetivos de negocio de su empresa.

A continuación se muestra una comparación entre las metodologías tradicionales y las metodologías ágiles:

Metodologías Tradicionales	Metodologías Ágiles
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de prácticas de producción de código
Generan cierta resistencia a los cambios	Especialmente preparadas para cambios durante el proyecto
Son impuestas externamente	Impuestas internamente por el equipo
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Más artefactos	Pocos artefactos
Más roles	Pocos roles
Grupos grandes y posiblemente distribuidos	Grupos pequeños (menos de 10 integrantes) trabajando en el mismo sitio
La arquitectura del software es esencial y se expresa mediante modelos	Menos énfasis en la arquitectura del *software*
Existe un contrato prefijado	No existe contrato tradicional o al menos es bastante flexible

Tabla 17. Comparación entre metodologías. Fuente: Corporación Colombia Digital.

3.5 MEDICION Y ESTIMACIÓN DE ESFUERZO

En los proyectos de desarrollo de sistemas de información es indispensable controlar el alcance, plazo y costos para las entidades, esto se puede realizar a través de técnicas de medición y estimación de esfuerzos como las siguientes:

3.5.1 Técnicas Bottom – up

Las más conocidas y utilizadas son:

Juego de póker - *Planning poker* o *poker game*: permite calcular estimaciones basadas en el consenso de todo el equipo de trabajo. Principalmente es empleada para estimar el esfuerzo o el tamaño relativo de las tareas de desarrollo de *software*, mediante una variación del método Delphi. Al ser estimaciones colaborativas y consensuales entre todo el equipo de trabajo (tanto el equipo técnico como de negocios), los resultados obtenidos usualmente comprometen a todo los participantes y, según estudios realizados por académicos, son incluso menos optimistas y más precisas que las estimaciones obtenidas a través del uso

de cálculos individuales de las mismas tareas. Es una técnica muy utilizada en proyectos de desarrollo de *software* bajo una metodología ágil.

Puntos de función: permite estimar el tamaño en puntos de función de la solución que va a construirse. Requiere como insumo un prototipo y los casos de uso a nivel de detalle, no las descripciones funcionales, pues cuando estas se utilizan el método es muy impreciso. Con el tamaño en puntos de función, si se dispone de estadísticas de desempeño, es posible encontrar un rango estadístico dentro del cual se comportará el proyecto en términos de esfuerzo, calendario y costo.

Puntos de casos de uso: cuando no se dispone de los casos de uso de manera detallada, algunos practicantes utilizan la técnica de puntos de caso de uso. Este método asigna niveles de esfuerzo a los casos de uso no detallados, dependiendo de la complejidad, y da una idea del tamaño, lo que lleva a dimensionar el esfuerzo y costo. Es supremamente impreciso y solo puede ser utilizado por una compañía que lo haya calibrado dentro de sus grupos desarrollo.

Método Delphi: es un método que con la estimación de tres expertos y la distribución de probabilidad de Pearson llega al resultado. Sólo puede ser utilizado en emprendimientos muy pequeños, no sirve para estimaciones de proyectos complejos.

Método del experto: fundamentado en la opinión de un individuo o grupo de individuos que han realizado tareas similares. Sólo se utiliza para tareas pequeñas.

Con los métodos de estimación aquí descritos se puede llegar a comprender el esfuerzo requerido para las actividades asociadas directamente con opciones o funcionalidades del sistema, es decir, desarrollo, revisiones, pruebas, y retrabajo. Se pasa a los puntos de función o puntos de historia de usuario y de allí se calcula el esfuerzo, basándose en las

líneas de desempeño de la compañía: velocidad en escritura de código, *loc* revisadas por hora y *loc* por hora en pruebas, entre otros.

Ahora, para calcular el esfuerzo de las labores como la administración del proyecto, la administración de la configuración, el entrenamiento, labores que no están asociadas con una funcionalidad sino con el proyecto o *sprint* general, es necesario contar con líneas base de distribución de esfuerzos, por etapas de proyectos similares.

Estas estimaciones se llevan a cabo usando simuladores de procesos, en los que se registran los datos, bien sea de puntos de historia, puntos de función, esfuerzo estimado por Delphi y las líneas base de la compañía (de calidad, productividad o distribución de esfuerzo), y el simulador hace el cálculo automático del esfuerzo en las demás actividades. Los simuladores pueden ser adquiridos a través de empresas especializadas en estimaciones como son Leda (www.leda-mc.com) y Fatto (www.fattocs.com)

3.5.2 Técnicas Top – Down

La más conocida y utilizada es Cocomo II o Cocomo avanzado. Poco se emplea para la estimación de esfuerzos de proyectos nuevos, puesto que ambas técnicas requieren como insumo, el tamaño en líneas de código para determinar el esfuerzo del proyecto o viceversa.

También es muy usada para conocer, en ciertas plataformas, cuántas líneas de código producirán un esfuerzo determinado o cuántas horas requieren unas líneas de código conocidas.

3.5.3 Comparación de las metodologías de estimación de esfuerzo usadas con mayor frecuencia

En la siguiente tabla se muestran las metodologías de estimación de esfuerzo más usadas, en las que si identifican el nivel de complejidad, ventajas y desventajas

Método	Complejidad	Ventajas	Desventajas
Puntos de función	Alta	Estandarizada y transparente. Independiente de la tecnología	Sin datos históricos, es difícil mejorar las habilidades de estimación No diferencian entre lenguajes, diseños o estilos
COCOMO II	Promedio a alta	Estándar Basada en modelos matemáticos	Depende de datos históricos. Puede no ser válido para todos los entornos de desarrollo
Estimación por expertos	Baja a promedio	Basada en la opinión de un grupo de expertos	Empírica Costosa, debido al número de reuniones. Aplicable sólo a proyectos similares
Basada en analogía	Baja a promedio	Basada en experiencias reales anteriores. Bastante preciso en proyectos similares	Depende del conocimiento y documentación de proyectos anteriores
Basada en porcentajes	Baja	Práctica para proyectos pequeños, con pequeños módulos o sub sistemas	Empírica e inexacta

Tabla 18. Comparación de las metodologías de estimación de esfuerzo. Fuente: Corporación Colombia Digital.

4 REQUERIMIENTOS

4.1 PROBLEMAS FRECUENTES

- Hay una retroalimentación poco adecuada por parte del usuario final sobre los requisitos funcionales del desarrollo.
- Se hacen cambios en los requerimientos levantados inicialmente, causados por variaciones dentro de los objetivos o prioridades de las entidades. Este tipo de modificaciones se producen por rotación en el equipo de liderazgo de la entidad o por situaciones en las que el contexto de la entidad cambia.
- Alteraciones en los requerimientos ocasionados por las largas esperas en el proceso de contratación del diseño y la arquitectura. Estos cambios se producen por desactualización en el tiempo de los requerimientos inicialmente levantados.
- Levantamiento incompleto de los requerimientos debido a la falta de experiencia en el tema de la persona responsable de esta actividad.
- Baja comprensión de los objetivos de la entidad y de cómo el proyecto de desarrollo de *software* contribuye a alcanzar dichos objetivos.
- Dificultad en la comunicación durante el levantamiento de los requerimientos. Los problemas más frecuentes de este tipo son: recolección informal de la información, funcionalidades implícitas que no son detalladas, supuestos no discutidos en el proceso, documentación pobre y un proceso de gestión de cambios informal.
- Dificultad para plantear los lineamientos básicos para realizar un estudio de viabilidad de los requerimientos identificados.
- Problemas para plantear los lineamientos básicos para detallar requerimientos que estén alineados con la necesidad del negocio.
- Oportunidades de mejora en la interacción entre las áreas de TI y las áreas funcionales, con el fin de hacer un adecuado levantamiento de requerimientos.
- Problemas al plantear los lineamientos básicos para evaluar la factibilidad y priorizar los requerimientos identificados.
- Necesidad de contar con personal experto en levantamiento y definición de requerimientos.

4.2 ASPECTOS GENERALES

Un requerimiento es cualquier solicitud o necesidad a partir de la cual se derive la elección de un diseño. Son las especificaciones de lo que debe ser diseñado y la descripción de cómo debe comportarse (Wiegers & Beatty, 2013). Los problemas en la fase de levantamiento de requerimientos causan entre el 40% y 50% de los defectos encontrados en el producto de *software*, por lo tanto es una de las fases más críticas dentro del ciclo de vida de desarrollo (Wiegers & Beatty, 2013). El retrabajo consume entre el 30% y el 50% de los costos y los errores asociados con el desarrollo, mientras que la gestión de los requerimientos produce entre el 70% y el 85% del retrabajo (Wiegers & Beatty, 2013).

Mejor práctica: *En la fase de requerimientos es importante tener en cuenta que es menos costoso iterar en la fase de requerimientos que en la fase de codificación. Por lo tanto, se sugiere dedicar recursos y tiempo suficientes a esta etapa, para asegurar la calidad de los entregables que se generan en este punto.*

Mejor práctica: *Se recomienda no asumir que las personas interesadas en el proyecto conocen el significado de lo que es un requerimiento. Se sugiere destinar tiempo dentro del proyecto para dar a conocer los conceptos clave, de tal forma que todo el equipo involucrado pueda hablar el mismo idioma.*

Los requerimientos de producto deben ser clasificados según el tipo de información como se muestra en la siguiente tabla.

Tipo de requisito / requerimiento	Descripción
Requerimiento del negocio	Hace parte de los objetivos de alto nivel de la entidad y está directamente relacionado con los servicios y productos que genera. Este tipo de requisitos explica la razón por la cual la entidad está implementando el sistema.
Regla de negocio	Guía, política, norma, estándar o regulación que enmarque o limite aspectos del negocio.

Limitación	Cualquier limitación en las opciones disponibles para el desarrollo del *software*.
Requisito externo de interface	Descripción de la conexión entre el *software* diseñado y cualquier otro sistema con el que deba interactuar.
Característica	Uno o varios elementos del sistema que generan valor para el usuario y que son descritos a partir de requisitos funcionales.
Requisito funcional	Descripción del comportamiento del sistema bajo condiciones específicas de operación.
Requisito no funcional	Descripción de una característica o propiedad con la que debe contar el sistema.
Atributo de calidad	Descripción de las características de servicio o desempeño del *software*. Incluye además la descripción de las características deseadas en términos de seguridad, disponibilidad y portabilidad.
Requisito del sistema	Descripción de un requisito de alto nivel de un producto que está conformado por varios subsistemas.
Requisito del usuario	Descripción de una meta o tarea específica que el usuario debe estar en capacidad de realizar con el desarrollo. Detallan lo que el usuario podrá hacer con el sistema.

Tabla 19. Clasificación de los requerimientos de producto según el tipo de información. Fuente: Wiegers y Beatty, 2013.

La generación de los requisitos de producto y su interrelación están descritas en la siguiente gráfica, en la que se resaltan en recuadros de color azul los elementos que se identifican en el momento de hacer el levantamiento de requisitos. Los recuadros de color naranja corresponden con los entregables que la fase de requerimientos. Las flechas muestran cuáles elementos son los insumos necesarios para crear dichos entregables.

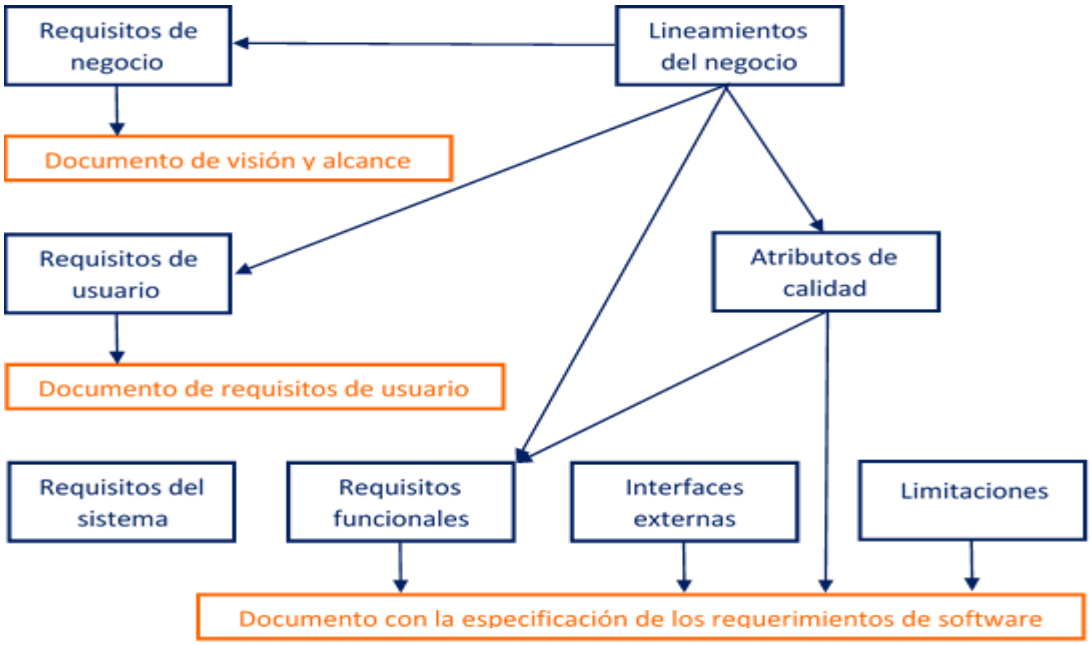


Figura 54. Relación entre los diferentes tipos de requisitos y entregables asociados. Fuente: Wiegers y Beatty, 2013.

Es esencial tener en cuenta que los requisitos de un proyecto de desarrollo de *software* son diferentes a los requisitos de producto descritos anteriormente. Los requisitos de proyecto se describen en la tabla que se presenta a continuación:

Tipo de requisito/requerimiento	**Descripción**
Requisitos sobre recursos físicos	Descripción de todos los recursos de *hardware* que se necesitan para el desarrollo. Incluye además laboratorios y herramientas de pruebas, espacio físico para el equipo de desarrollo, herramientas de colaboración, etc.
Requisitos de entrenamiento del personal	Descripción de la formación académica requerida, certificaciones y experiencia de cada uno de los integrantes del proyecto de *software*.
Requisitos de documentación	Descripción de los documentos requeridos para la ejecución del proyecto: manuales, tutoriales, material de entrenamiento, documentación de procesos, organigrama, etc.
Requisitos de cambio en la infraestructura	Descripción de todos los cambios en infraestructura requeridos para que el desarrollo opere de forma óptima y garantice, adicionalmente, su interoperabilidad con los sistemas relevantes.

Requisitos de migración	Descripción de los requisitos de conversión y migración de datos cuando se da un cambio de sistema. Asimismo, descripción de los lineamientos de seguridad por seguir para la migración.
Requisitos sobre certificaciones del producto	Descripción de las certificaciones de producto requeridas y las directrices de gobierno que debe cumplir el producto.
Requisitos de *software*, licencias y *hardware*	Descripción de los componentes suministrados por terceras partes.
Requisitos exigidos en otras fases	Requisitos que debe cumplir el proyecto en fases como pruebas, codificación y mantenimiento.
Requisitos para gestionar derechos de autor	Requisitos que debe cumplir el proyecto para facilitar la adecuada gestión de los derechos de autor.

Tabla 20. Clasificación de los requerimientos del proyecto. Fuente: Wiegers y Beatty, 2013.

4.3 DETALLES DE LA FASE DE REQUERIMIENTOS

La fase de requerimientos se compone de dos subfases fundamentales: desarrollo y gestión de los requerimientos, como se muestra en la siguiente figura. La primera subfase, de desarrollo, está compuesta por cuatro pasos: levantamiento, análisis, especificación y validación.

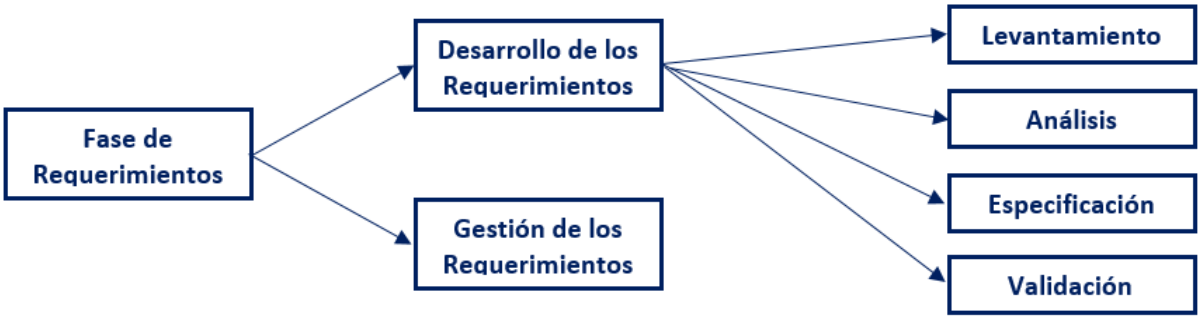


Figura 55. Subfases y pasos en la fase de requerimientos. Fuente: Wiegers y Beatty, 2013.

Es importante resaltar que los cuatro pasos en el desarrollo de los requerimientos no se ejecutan de forma lineal y en una sola oportunidad, debido a las complejas interrelaciones que se identificarán entre los requerimientos. Además, entre la subfase de desarrollo y la subfase de gestión existe un paso intermedio, que es un momento crítico para establecer el punto de partida del proyecto de desarrollo. Este inicio se

conoce como línea base de requerimientos para una versión o iteración del producto, tal y como muestra la siguiente figura:

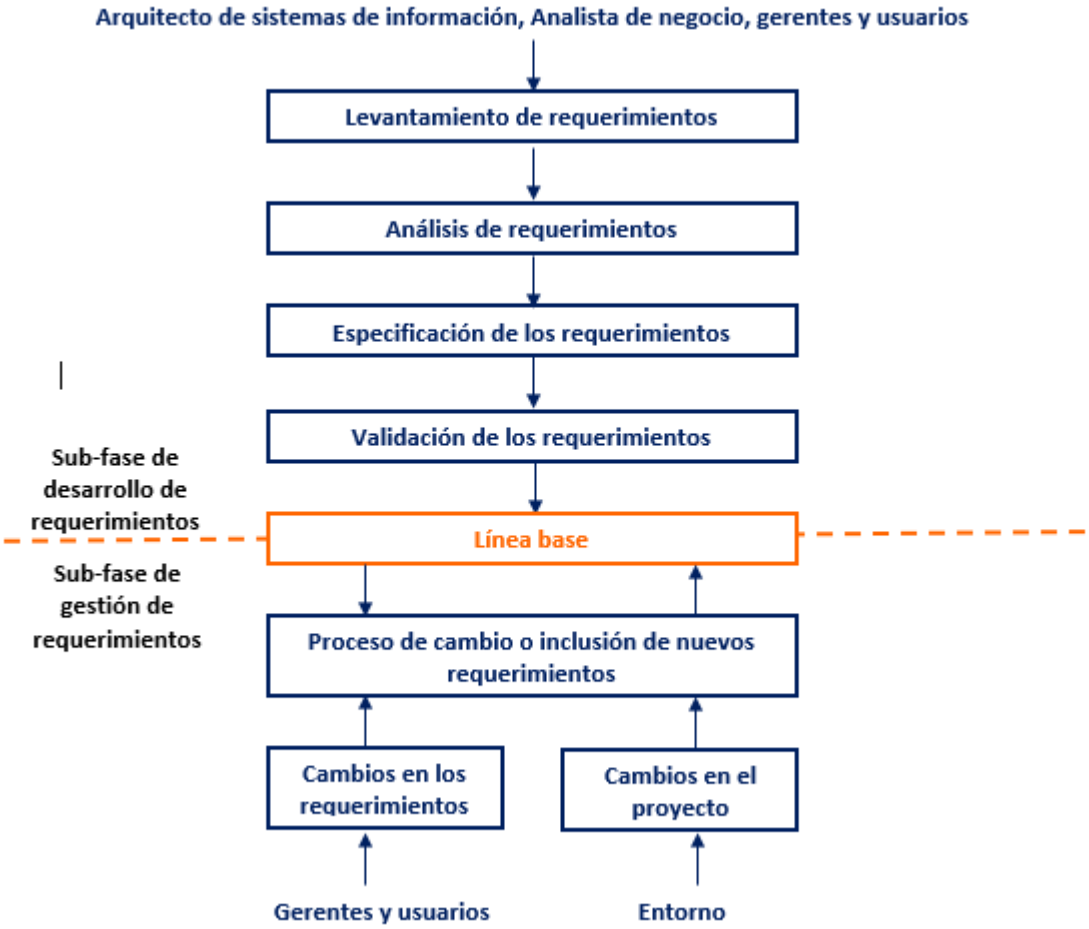


Figura 56. Descripción del proceso en la fase de requerimientos. Fuente: Wiegers y Beatty, 2013.

Profundizando en la subfase de desarrollo, se encuentra que los pasos por desarrollar en esta etapa no se ejecutan linealmente en una sola oportunidad; en la práctica estas actividades están entrelazadas, se ejecutan simultáneamente y requieren de varias iteraciones, como se observa en la siguiente figura:

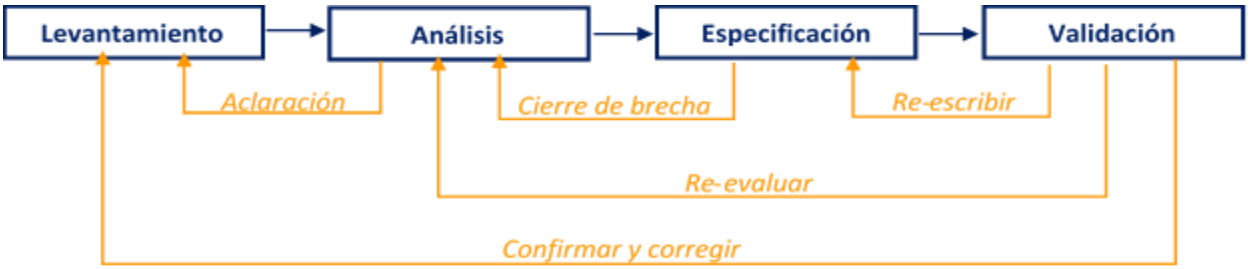


Figura 57. La subfase de desarrollo de requerimientos como proceso iterativo. Fuente: Wiegers y Beatty, 2013.

Asimismo, la subfase de desarrollo contiene varios elementos que se deben ejecutar en los cuatro pasos mencionados anteriormente. La siguiente tabla muestra el detalle de cada uno de estos elementos:

Pasos de la subfase de desarrollo	Elementos para ejecutar en cada paso
Levantamiento	Definición de los requisitos del negocio
	Identificación de las clases de usuario
	Identificación de los representantes de las clases de usuario
	Identificación de las personas o áreas a cargo de las decisiones sobre requerimientos
	Planeación del levantamiento
	Identificación de los requerimientos de usuario
Análisis	Priorización de los requerimientos
	Profundización en los requerimientos de usuario
Especificación	Modelamiento de los requerimientos
	Especificación de los requerimientos no funcionales
	Revisión los requerimientos
	Desarrollo de prototipos
	Desarrollo o evaluación de arquitectura
	Asignación de los requerimientos a componentes
	Desarrollo de pruebas para los requisitos
Validación	Validación de requerimientos de usuario, funcionales, no funcionales, análisis de modelos y prototipos

Tabla 21. Elementos para ejecutar en cada paso de la subfase de desarrollo. Fuente: Wiegers y Beatty, 2013.

Por otro lado la subfase de gestión de requerimientos contiene varios elementos que se deben ejecutar en los pasos mencionados anteriormente. La tabla siguiente muestra el detalle de cada uno de estos elementos:

Pasos de la subfase de gestión de requerimientos	Elementos para ejecutar en cada paso
Proceso de cambio o inclusión de nuevos requerimientos	Desarrollo de una herramienta de control de versiones en la línea base
	Desarrollo de una herramienta de control de cambios
	Desarrollo de una herramienta que permita dar seguimiento al estado de los requerimientos
	Desarrollo de una herramienta que permita hacer trazabilidad de los requerimientos a otros componentes del sistema

Tabla 22. Elementos para ejecutar en cada paso de la subfase de gestión de requerimientos. Fuente: Wiegers y Beatty, 2013.

4.4 IDENTIFICACIÓN Y CLASIFICACIÓN DE USUARIOS

El objetivo principal de este paso es conocer la voz del usuario, eso se logra ejecutando las siguientes tres actividades:

- Identificar las diferentes clases de usuario del producto que será desarrollado.
- Seleccionar y trabajar con un grupo que represente cada clase de usuario identificado y a los interesados que sean relevantes durante la fase de requerimientos.
- Acordar de antemano quién o quiénes serán las personas a cargo de las decisiones en las fases de desarrollo y gestión de los requerimientos.

Para identificar y clasificar los usuarios, se debe tener en cuenta criterios como:

- Los privilegios de acceso o permisos de seguridad permitidos

- Las tareas que desempeñan
- Las características que usan
- La frecuencia con la que usarán el desarrollo
- Su experiencia en el uso de tecnología
- Las plataformas que serán usadas
- El tipo de relación que tendrán con el sistema desarrollado: directa o indirecta
- Su ubicación geográfica
- El área de la entidad a la que pertenecen

Mejor práctica: *Se sugiere no ignorar las necesidades de los usuarios indirectos, ya que en la mayoría de casos este grupo tiene perfiles menos operativos y, por lo tanto, mayor poder dentro de la entidad.*

Mejor practica: *Se sugiere tener en cuenta que es posible identificar usuarios que no son humanos sino sistemas que dependen del producto desarrollado. Los requerimientos de estos usuarios son igualmente importantes, por lo que este tipo de usuarios debe identificarse y clasificarse. Por otro lado, es posible que bajo esta clasificación se detecten usuarios no deseados, con los que se deben establecer los requerimientos que el desarrollo no debe atender.*

Mejor practica: *Para identificar los usuarios, se sugiere agendar una reunión con el patrocinador del proyecto, con el fin de desarrollar una lluvia de ideas para definir tantas clases de usuarios como se pueda. Posteriormente, los perfiles identificados deben ser condesados aproximadamente en 15 clases de usuarios, que deben ser agrupados teniendo en cuenta necesidades comunes.*

Las clases de usuario pueden ser identificadas estudiando el organigrama de la entidad y encontrando los siguientes grupos:

- Áreas que participan en el proceso de negocio
- Áreas que son afectadas por el proceso de negocio
- Interesados externos que son afectados por el proceso de negocio

Elemento transversal – documentación: *La documentación o entregable que debe producir la identificación de los usuarios consiste en una matriz como la que se muestra a continuación:*

Nombre de la clase de usuario	Número de usuarios	Descripción	Representantes de la clase de usuarios	Poder	Interés
		Descripción demográfica del usuario.	Nombres.	Bajo.	Bajo.
		Comportamientos.	Datos de contacto.	Alto.	Alto.
		Preferencias.			
		Molestias.			

Tabla 23. Matriz de identificación y clasificación de usuarios. Fuente: Corporación Colombia Digital – CCD.

Mejor práctica: *Si los interesados no se involucran adecuadamente durante la fase de desarrollo y gestión de requerimientos, se generan diferencias entre sus expectativas interesado y lo que el equipo del proyecto esté ejecutando. La siguiente figura ilustra este fenómeno y la problemática que se deriva si no se gestiona de forma oportuna.*

Para resolver este problema se sugiere establecer puntos de contacto con los interesados en los momentos clave del proyecto, mediante herramientas como entrevistas, reuniones de revisión de requerimientos, presentaciones de los avances, presentaciones de prototipos, etc.

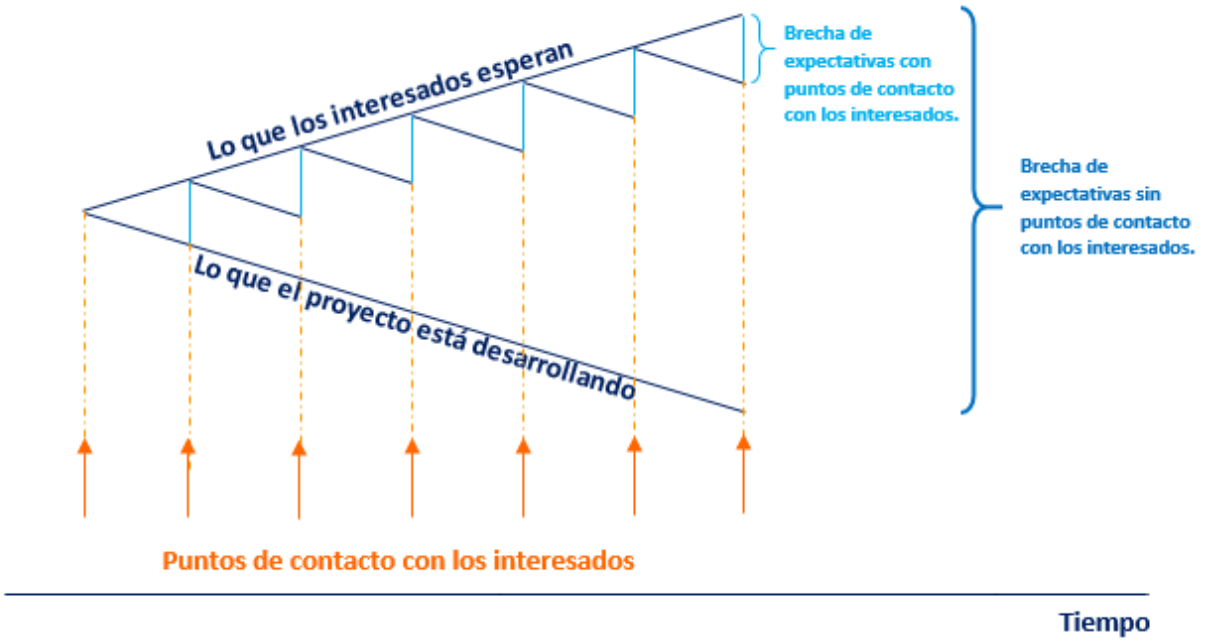


Figura 58. Relación entre los puntos de contacto con los interesados y la brecha de expectativas. Fuente: Sharp, Galal & Finkelstein, 1999.

Elemento transversal – Riesgos: *Interesados inadvertidos. Este riesgo se materializa cuando en la fase de gestión de requerimientos se detecta un nuevo usuario o interesado que no fue tenido en cuenta durante la fase de levantamiento, por lo tanto el producto desarrollado no cumple con la funcionalidad esperada.*

4.5 LEVANTAMIENTO DE REQUERIMIENTOS

Las siguientes son las cuatro actividades claves que se sugiere incluir para cubrir el proceso de levantamiento de requisitos:

- Identificar los posibles tipos de usuarios del producto y el universo de interesados que están de alguna forma relacionados con el proyecto.
- Entender los objetivos y las tareas de los usuarios y como estos se encuentran alineados con la estrategia de la entidad.

- Entender el ambiente bajo el cual el nuevo desarrollo será utilizado.
- Trabajar con cada uno de los tipos de usuarios identificados para entender cuál es la funcionalidad esperada y las expectativas de calidad sobre el producto.

El proceso de levantamiento de requerimientos se desarrolla a través de dos tipos de actividades: el primer tipo implica interacción con los interesados identificados y el segundo lo desarrolla el analista de negocio por su cuenta para descubrir aquellas relaciones entre la información que suministran los interesados, detectar incongruencias o encontrar información adicional.

Mejor práctica: *Se sugiere tener muy presente el alcance y visión de proyectos durante este paso, con el fin de validar la pertinencia de cada requerimiento encontrado.*

Mejor práctica: *Se debe recordar que otros sistemas pueden ser usuarios del desarrollo, por lo tanto se deben listar los eventos externos que el sistema puede experimentar, que pueden ser de tres tipos: señales de control o datos enviados por otros sistemas, eventos temporales que se activan con determinada periodicidad y eventos de negocio que se producen cuando ocurren situaciones que afectan de manera directa o indirecta la operación normal del negocio.*

Las herramientas usadas en el levantamiento de requerimientos son:

Entrevistas: es una forma directa de obtener información, se pregunta al usuario cuáles son sus necesidades puntuales, bien sea de forma individual o en grupos de no más de tres personas. En proyectos que utilizan metodología ágil son una forma rápida de obtener información. Las siguientes son las consideraciones más importantes para desarrollar de forma eficaz una entrevista:

- Genere empatía: para iniciar la entrevista preséntese, revise la agenda, recuerde los objetivos de la sesión y atienda las preguntas iniciales o preocupaciones de los participantes.
- No salga del alcance del proyecto: mantenga el foco en el proceso de levantamiento, dentro de los lineamientos del proyecto. En caso que los cometarios se dispersen, tome el control de la conversación, agradezca los aportes y enfoque el grupo nuevamente recordando el alcance y el objetivo del proyecto y la entrevista.
- Prepárese con anticipación: escriba las preguntas de la entrevista y los modelos que se utilizarán durante la sesión, esto le permitirá guiar la conversación y crear un punto de partida sobre el cual pueda comenzar a construir.
- Participe: el rol de quien hace el levantamiento de los requerimientos no se limita a transcribir lo que escucha. Sugiera ideas y haga observaciones desde el punto de vista del desarrollador. Si observa opciones de mejora, valide las alternativas con el grupo entrevistado.
- Escuche activamente: el lenguaje corporal revela el interés que usted tiene sobre el tema que está siendo discutido. Es importante inclinar levemente el cuerpo hacia el interlocutor, dar retroalimentación sobre el tema discutido, parafrasear para garantizar que se ha comprendido una idea y hacer preguntas adicionales cuando hay dudas (Wiegers & Beatty, 2013).

Mejor practica: *Las entrevistas son una herramienta útil para levantar requerimientos, en particular cuando la persona entrevistada dispone de poco tiempo. Las entrevistas son particularmente útiles como un paso de preparación para los talleres.*

Talleres: es una reunión de un grupo multidisciplinario que cuenta con perfiles escogidos cuidadosamente para lograr colaboración en la identificación y refinamiento de los requerimientos. Por lo general el grupo multidisciplinario está compuesto por usuarios, gerentes de negocio, desarrolladores e integrantes del equipo de pruebas. Un taller no debe contar con más de cinco personas para lograr que sea una actividad realmente efectiva.

El equipo definido para trabajar el taller debe contar con un líder que sea responsable de planear el taller, seleccionar los participantes y moderar el desarrollo para cumplir con el propósito final. Para evitar que esta actividad se convierta en una pérdida de tiempo y esfuerzo, contemple las siguientes pautas:

- Establezca reglas de trabajo: comprometa al equipo para comenzar y terminar a tiempo, respetar los tiempos designados para descansos, limite en lo posible el uso de dispositivos electrónicos para tareas ajenas al taller, mantenga solo una conversación al tiempo, incentive la participación de las personas introvertidas y recuérdele al grupo mantener el foco de las críticas sobre las ideas y no sobre las personas.
- Distribuya responsabilidades entre el equipo: tome de notas, monitoree del tiempo, mantenga el foco en el alcance y verifique el cumplimiento de las reglas de trabajo.
- Defina tiempos para la discusión de temas específicos: con frecuencia el tiempo asignado para cubrir cierta cantidad de temas no alcanza porque la discusión sobre los primeros temas se extiende más de lo presupuestado. Recuerde que es necesario usar eficientemente el tiempo y para que al final de la reunión no queden temas importantes sin ser discutidos.
- Monitoree la participación del equipo: es posible que algunos integrantes participen menos debido a ser introvertidos, en esos casos es importante integrar metodologías de participación que contemplen este tipo de personalidades. Asimismo, se debe estar atento a personas que han participado activamente y que de un momento a otro dejan de hacerlo, ya que esto es un síntoma de inconformidad que debe ser manejado para garantizar el éxito del taller. En general, esté pendiente del lenguaje corporal de los participantes: contacto visual, revisión frecuente del reloj, ansiedad, etc.
- Siga también las consideraciones planteadas para una entrevista.

Mejor práctica: *los talleres facilitan el levantamiento de requerimientos, ya que permiten la colaboración entre diferentes áreas para la elaboración de un mismo requerimiento al brindar puntos de vista complementarios y solucionar cualquier inconsistencia previamente detectada.*

Grupos de enfoque: el propósito de estos grupos es suministrar retroalimentación sobre los requerimientos funcionales y de calidad del producto, a través de la identificación de impresiones, preferencias, necesidades y rasgos de personalidad. Las personas que se deben incluir en un grupo de enfoque son normalmente usuarios con experiencia en versiones anteriores o similares del producto. Tenga en cuenta que los grupos de enfoque no toman decisiones sobre los requerimientos identificados.

Observación: con frecuencia los usuarios en las entrevistas, talleres y grupos de enfoque olvidan mencionar características relevantes o dan información incorrecta. Con el fin de mitigar este riesgo es muy importante destinar tiempo para observar en vivo la ejecución de las tareas. Desafortunadamente, esta actividad consume bastante tiempo y, en el caso de la metodología ágil, esta herramienta debe tener un uso muy limitado. El proceso de observación puede ser silencioso o interactivo, depende de la disponibilidad de tiempo del usuario.

Elemento transversal – seguridad: *El proceso de observación puede apoyarse en herramientas de grabación, si las políticas de seguridad de la entidad así lo permite.*

Cuestionarios: son una herramienta que permite indagar sobre la necesidad de un grupo grande de personas. Una de las principales ventajas de los cuestionarios es que pueden ser aplicados fácilmente en sitios remotos a los cuales el equipo de levantamiento no tiene fácil acceso.

Mejor práctica: *se sugiere utilizar los cuestionarios como una herramienta de apoyo para enfocar apropiadamente los esfuerzos de levantamiento de información.*

Las consideraciones al diseñar un cuestionario son las siguientes:

- En lo posible trate de usar cuestionarios de opción múltiple para facilitar el proceso de evaluación de resultados. No olvide cubrir todas las opciones disponibles para una respuesta de opción múltiple.
- Cuando utilice rangos numéricos verifique que no se sobreponen y que sean consistentes.
- Verifique que en las respuestas de opción múltiple no haya dos opciones o más que sean posibles de forma simultánea.
- Tenga cuidado de no formular preguntas que sugieran o induzcan una respuesta especifica.
- Verifique que haya consistencia a lo largo del cuestionario (uso de los mismos términos, escalas, etc.).
- Tenga en cuenta que las preguntas abiertas pueden dar respuestas valiosas en términos de las expectativas y experiencias del usuario, pero no olvide que estas preguntas son difíciles de evaluar y no permiten hacer un análisis estadístico.
- Verifique el contenido del cuestionario con expertos de desarrollo y del área del negocio.
- Pruebe el cuestionario antes de usarlo. Verifique si la cantidad de tiempo programado es suficiente, si las preguntas son claras y si el cuestionario cumple el objetivo final.
- No haga demasiadas preguntas. Siempre revise cuál es la intención o propósito de cada pregunta y cómo contribuye al proceso de levantamiento de requerimientos.

Análisis de la interface de sistema: esta herramienta permite identificar requerimientos funcionales derivados de la conexión con otros sistemas. Puntualmente, los requisitos pueden ser de intercambio de datos o servicios. Este análisis de sistemas facilita además detectar sinergias entre los sistemas y evitar re trabajos durante el desarrollo.

Análisis de la interface de usuario: se usa cuando existen versiones anteriores o similares del sistema que será desarrollado. El propósito es detectar requisitos funcionales y de usuario en el sistema antiguo. Cuando no existen versiones anteriores o parecidas, este ejercicio se puede desarrollar investigando sistemas similares en el mercado. Es importante validar con el usuario los hallazgos que se hacen con esta herramienta, ya que no necesariamente hacen parte de los requerimientos que se esperan con el nuevo sistema.

Análisis de la documentación: la documentación existente es otra fuente de información sobre requerimientos, puede revelar diferencias entre la forma en que se realizan las tareas y el diseño inicial de las mismas. Entre los documentos usados con más frecuencia se encuentra: especificaciones de requerimientos de otras versiones o desarrollos similares, manuales de procesos de negocios, mapas de procesos, lecciones aprendidas de otros proyectos, el organigrama de la entidad, etc. Idealmente, el análisis de la documentación debe ser una de las primeras herramientas que se utilizan durante el levantamiento, ya que la información obtenida permite sacar mayor provecho de las otras herramientas descritas.

Mejor práctica: *los procesos o procedimientos actuales pueden tener importantes oportunidades de mejora. Durante el levantamiento de requerimientos es crítico tener en cuenta dichas oportunidades de mejora para desarrollar un sistema que genere valor agregado a la entidad.*

Mejor práctica: *en los casos en que sea posible, reutilice requerimientos existentes, esto permitirá emplear con mayor eficiencia el tiempo y profundizar en temas de análisis y validación de requerimientos.*

Tenga en cuenta que durante el proceso, se pude obtener una descripción detallada de la tarea que el usuario final desea realizar, pero no de todos los requisitos funcionales que el desarrollador debe implementar para cumplir con esa tarea.

Para mejorar la probabilidad de éxito del levantamiento, se puede usar como guía la siguiente tabla que ilustra la clasificación de los interesados relevantes. Es importante anotar que este modelo es solo una guía que puede cambiar de acuerdo con las condiciones particulares de cada proyecto, por lo cual su implementación debe ser evaluada por el lector:

Tipo de requisito	Interesados relevantes
Requisitos de negocio	Gerentes, líderes y directivas
Requisitos de usuario	Analistas de negocio, usuarios y gerentes de producto
Requisitos funcionales	Analistas de negocio y gerentes de producto

Tabla 24. Interesados relevantes en el levantamiento de requisitos. Fuente: Wiegers & Beatty, 2013.

El proceso de levantamiento de requerimientos debe ser planeado cuidadosamente a través de la definición de los siguientes puntos:

- Objetivos del levantamiento de requerimientos
- Estrategia y herramientas que se usarán
- Cronograma de trabajo
- Estimación de recursos necesarios
- Listado de documentos requeridos
- Listado de accesos a sistemas requeridos
- Riegos derivados del proceso

Las actividades que se ejecutan cada vez que se utiliza una de las herramientas de levantamiento se resumen en la siguiente figura:

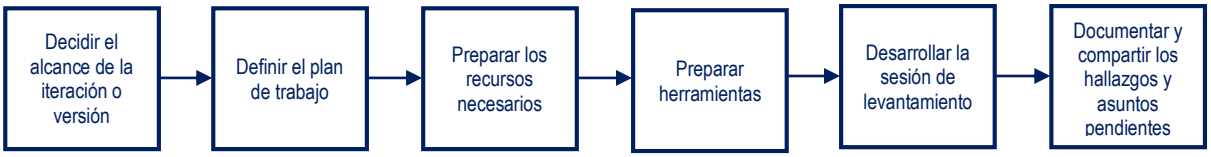


Figura 59. Actividades de preparación para un proceso de levantamiento de requerimientos. Fuente: Wiegers & Beatty, 2013.

Mejor práctica: *antes de comenzar las sesiones de levantamiento de información se sugiere contextualizar a los participantes para que conozcan el proceso y su importancia.*

Mejor práctica: *durante el desarrollo de las sesiones de levantamiento de requerimientos, se sugiere asignar a una persona que no esté participando activamente, para que tome notas de buena calidad.*

Mejor práctica: *durante el levantamiento de ideas, se sugiere usar elementos versátiles que promuevan el flujo de propuestas. Por ejemplo, pueden usarse post-it para plasmar cada necesidad y expectativa. Posteriormente, esos papeles pueden ser agrupados y pegados en las paredes para fomentar la colaboración.*

Mejor práctica: *una vez la sesión de levantamiento ha concluido, trate de organizar y documentar los hallazgos tan pronto como pueda, para garantizar que la información esté fresca en su memoria.*

Mejor práctica: *no olvide dar seguimiento a los puntos que quedaron abiertos: información adicional requerida, inconsistencias, dependencias, etc. Implemente algún instrumento que le permita organizar y seguir cada punto según su criticidad.*

Dependiendo del tipo de sistema que se desea desarrollar, se sugiere el uso las herramientas planteadas en este documento, tal y como se observa en la siguiente tabla:

	Entrevistas	Talleres	Grupos de enfoque	Observación	Cuestionarios	Análisis de la interface de sistema	Análisis de la interface de usuario	Análisis de documentación
Software de uso masivo	X		X		X			
Software de uso interno en la entidad	X	X	X	X		X		X
Software que sustituye un sistema existente	X	X		X		X	X	X
Software que amplía el alcance de un sistema existente	X	X				X	X	X
Nueva aplicación	X	X				X		
Personalización de un *software*	X	X		X		X		X
Sistemas embebidos	X	X				X		X
Interesados diseminados geográficamente	X	X			X			

Tabla 25. Herramientas de levantamiento sugeridas según el tipo de desarrollo. Fuente: Wiegers & Beatty, 2013.

El proceso de levantamiento de requerimientos nunca termina por completo. A medida que se avanza y profundiza en el proyecto, se encuentran nuevos requerimientos; sin embargo, es posible saber que no se requieren más iteraciones o versiones cuando:

- A los usuarios no se les ocurren más casos de uso o historias de uso.
- Los usuarios proponen nuevos escenarios, pero estos no llevan a identificar nuevos requerimientos.
- Los usuarios repiten temas ya cubiertos en sesiones anteriores.
- Los usuarios o interesados sugieren requerimientos de usuario o funcionales que están fuera del alcance.
- Los usuarios o interesados sugieren requerimientos de usuario o funcionales de baja prioridad.

Mejor práctica: *no olvide que existen requerimientos que los usuarios definen como obvios o implícitos, que pueden estar quedando por fuera de la identificación. Para mitigar este riesgo siempre formule preguntas como: ¿cuáles son los supuestos de este escenario? o descomponga requerimientos complejos en requerimientos simples para profundizar en su significado.*

Mejor práctica: *asegurar la calidad de los requisitos es una tarea casi imprescindible para culminar un proyecto de desarrollo de software con éxito. Existen varias técnicas ampliamente aceptadas y documentadas para mejorar la calidad de los requisitos, como pueden ser revisiones formales, inspecciones, listas de verificación, ontologías, auditorias, matrices de rastreabilidad, métricas de calidad, etc.*

Elemento transversal – Riesgos: *El usuario final se involucra mínimamente en el desarrollo de los requerimientos. Las causas para la baja participación de los usuarios finales pueden ser:*

- *El analista de negocio o el desarrollador puede pensar que comprende bien lo que el usuario final desea. Este es un error que se presenta con alguna frecuencia y se previene siguiendo los pasos que se sugieren en la subfase de desarrollo de requerimientos. En particular, el paso de validación permite identificar problemas de esta naturaleza.*
- *Dificultad para acceder a los usuarios finales. Puntualmente esta fue una de las situaciones que se observó con mayor frecuencia en el interior de las entidades. La carga laboral que significa el día a día y los problemas de comunicación entre áreas afecta de forma importante la participación de los usuarios finales en el desarrollo de requerimientos.*

Elemento transversal – Riesgos: *Acumulación progresiva de requisitos del usuario. Durante el desarrollo de los requerimientos es frecuente que se incluyan algunos requisitos que exceden el alcance inicial. Las causas frecuentes de este tipo de situaciones incluyen supuestos optimistas de duración de las actividades o proveedores inexpertos que tratan de*

generar mayor valor para el cliente agregando características que no han sido solicitadas. Ambas conductas generan riesgos para el proyecto de desarrollo de software porque comprometen su éxito.

Mejor práctica: *contratar un tercero especializado para realizar el proceso de levantamiento de requerimientos, en el caso que las entidades no cuenten con los recursos necesarios para hacerlo en su interior.*

4.6 ANÁLISIS DE REQUERIMIENTOS

El análisis de los requerimientos significa lograr un nivel más profundo de comprensión de cada requerimiento identificado y garantizar que todos los interesados tengan ese mismo nivel de entendimiento. Las siguientes son las actividades clave que se sugiere incluir para dicho análisis:

- La información recopilada debe ser clasificada para diferenciar claramente los objetivos de la tarea, los requisitos funcionales, las expectativas de calidad, las reglas del negocio, las soluciones sugeridas, etc.
- Se debe verificar que el nivel de detalle de los requerimientos sea adecuado para que se entienda fácilmente y no haya lugar a múltiples interpretaciones.
- El próximo paso consiste en identificar requerimientos que se deriven a partir de otros requerimientos y que no hayan sido identificados durante la fase de levantamiento.
- Priorizar los requerimientos siguiendo criterios definidos por el proyecto de desarrollo.
- Por último se deben identificar incongruencias y verificar que cada uno de los requerimientos se ajuste al alcance inicialmente definido.

Mejor práctica: *para requerimientos muy complejos o de muy alto nivel, se sugiere comenzar el proceso de análisis descomponiendo el requerimiento de tal forma que se logre un nivel de detalle suficiente para facilitar su comprensión. Adicionalmente, se puede*

representar el requerimiento de forma gráfica para dar mayor claridad o para encontrar elementos importantes que inicialmente no hayan sido tenidos en cuenta.

El proceso de análisis comienza modelando el ambiente bajo el cual debe operar el sistema. Esta primera actividad se puede lograr utilizando alguna de las siguientes herramientas:

Diagrama de contexto: permite definir los límites del sistema, las interfaces con las que debe interactuar, las interconexiones y dependencias con otros sistemas, el tipo de datos que se intercambian, los procesos de control y el flujo de información del contexto. La siguiente gráfica muestra un ejemplo muy básico de un diagrama de contexto:

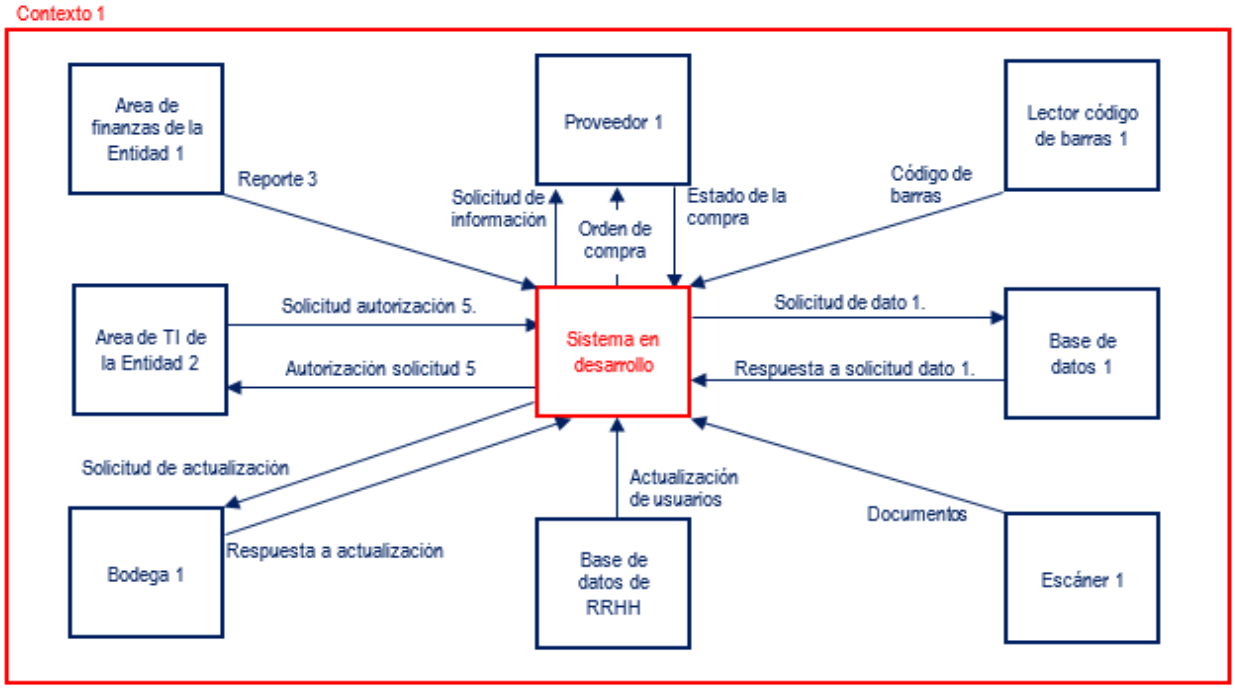


Figura 60. Ejemplo básico de un diagrama de contexto. Fuente: Wiegers & Beatty, 2013.

Mapa de ecosistema: Este es otro tipo de diagrama sobre el cual se puede apoyar el análisis. El mapa del ecosistema ilustra todos los sistemas que interactúan con el sistema en desarrollo, la naturaleza de dichas interacciones y el alcance del proyecto de desarrollo frente a los otros sistemas. La siguiente gráfica muestra un ejemplo básico de un mapa de ecosistema:

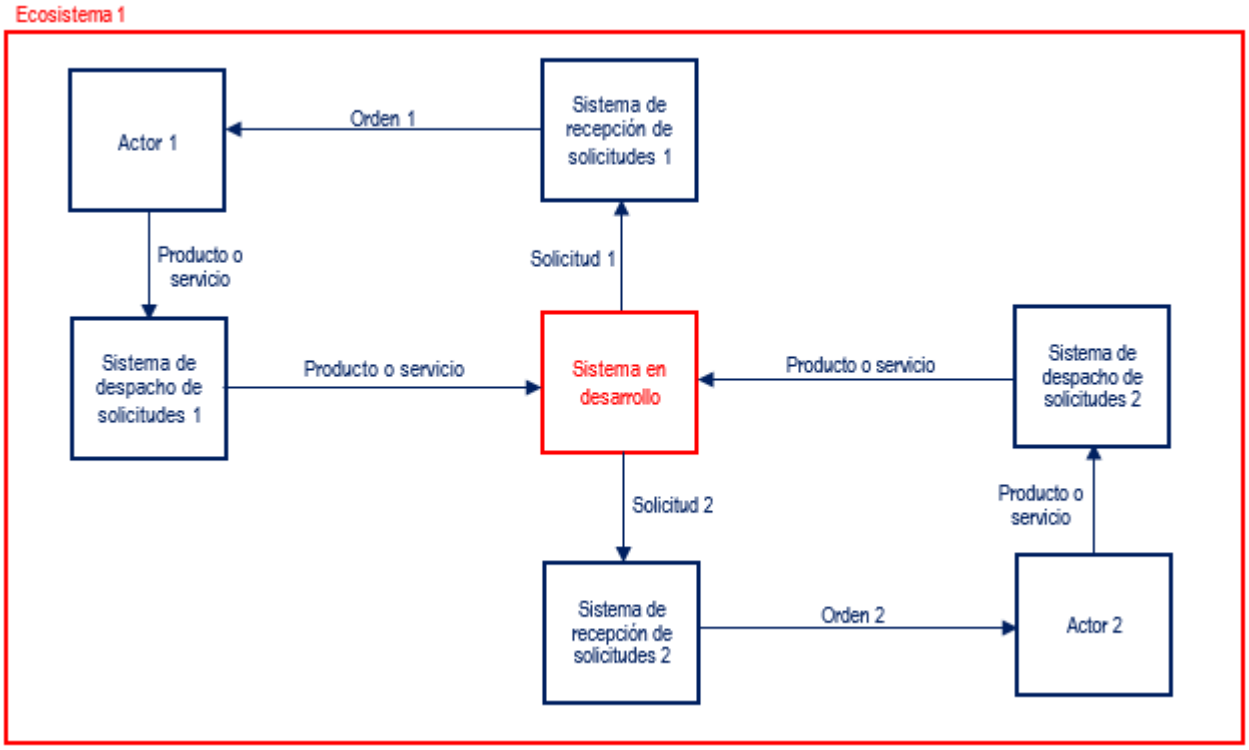


Figura 61. Ejemplo básico de un mapa de ecosistema. Fuente: Wiegers & Beatty, 2013.

Diagrama de causa y efecto: Esta herramienta profundiza sobre un evento o tarea a través de la identificación lógica y organizada de los prerrequisitos que permiten que ese evento o tarea se lleve a cabo. Asimismo, este tipo diagrama organiza las causas de forma jerárquica y las agrupa de acuerdo con su naturaleza. El siguiente es un esquema genérico de un diagrama de causa y efecto (Heizer & Render, 2009)**:**

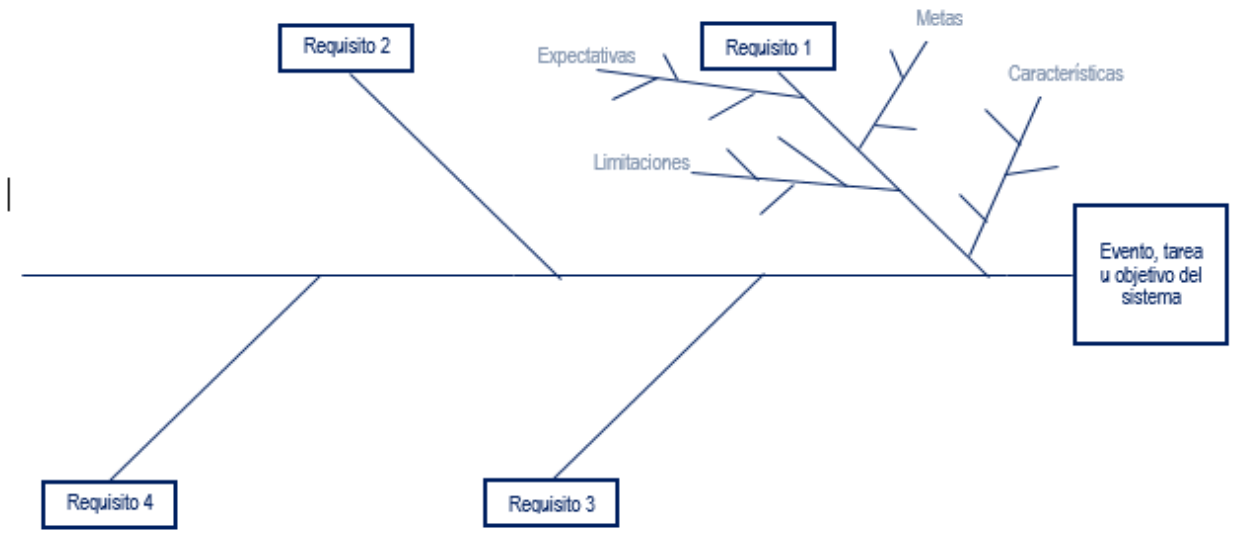


Figura 62. Ejemplo básico diagrama causa y efecto. Fuente: Wiegers & Beatty, 2013.

En el diagrama anterior es importante tener en cuenta que las ramas o agrupaciones sobre cada requisito que se muestran en la figura anterior son solo un ejemplo. En la realidad las agrupaciones deben ser definidas de acuerdo con las características del proyecto y de los requerimientos levantados.

Diagramas de flujo: se utilizan para describir un sistema, proceso o tarea, interconectando cada paso de acuerdo con la secuencia de actividades identificada y estableciendo claramente los puntos de decisión, entradas y salidas de cada paso. Son una herramienta útil para hallarle sentido a un proceso y explicarlo con detalle (Heizer & Render, 2009).

Listas de eventos: contienen los eventos externos que pueden disparar determinada conducta en el sistema. Esta herramienta permite visualizar todos los eventos posibles y validar si hace falta alguno. Asimismo, facilita establecer si se han identificado los requerimientos necesarios para cada tipo de evento (Wiegers & Beatty, 2013).

Una vez se ha modelado el ambiente en el cual debe operar el sistema, se puede proceder a crear una interface o prototipo para los casos en los que no se tiene seguridad sobre la

comprensión y análisis de un requerimiento. La idea del prototipo es lograr que los conceptos y posibilidades sean tangibles para todos los interesados. Los prototipos de *software* no necesariamente son funcionales y muchas veces son modelos estáticos como secuencias de pantallazos o simulaciones. El prototipo debe tener en cuenta los siguientes puntos:

- Debe aclarar, completar o validar un requerimiento.
- Debe permitir explorar diferentes alternativas de diseño.
- Debe generar un resultado que pueda ser reutilizable en la fase de desarrollo.

Existen tres tipos de atributos en un prototipo:

- Prototipo maqueta: se concentra en la experiencia de usuario, la verificación del concepto o la validación del acercamiento técnico.
- Prototipo desechable: es un prototipo sobre el cual se itera. Una vez finalizada cada iteración se rescata la retroalimentación, se desecha y se comienza un nuevo prototipo.
- Prototipo de papel: se utiliza inicialmente para aterrizar las ideas, esbozándolas en gráficos generados por los interesados.
- Prototipo electrónico: es un *software* que opera parte de la solución.

Es importante resaltar que los prototipos no solo se usan para aclarar requerimientos. Es posible observar en proyectos de desarrollo de *software* que las fases de definición de arquitectura, diseño y codificación también usan prototipos.

En el momento en que los requerimientos están claros, se procede a analizar su viabilidad. En este paso, la idea es identificar los riegos sobre cada requerimiento, el costo de implementación y su impacto en el desempeño del sistema por desarrollar. Requiere más investigación.

Una vez se ha establecido la viabilidad de los requerimientos, se deben priorizar para asegurar que se implementen primero los que mayor valor generen. La priorización se puede dar una vez se tenga claridad en los siguientes siete aspectos:

- Las necesidades del cliente
- La importancia relativa de cada requisito para los clientes
- El momento en que el cliente debe contar con cada característica
- La jerarquía cronológica de los requerimientos; es decir, cuales son predecesores de otros requerimientos
- Las relaciones de interdependencia entre los requerimientos
- Los requerimientos que deben ser implementados de forma simultánea
- El costo de satisfacer cada requerimiento

De acuerdo con esta información, el analista de negocio puede plantear un primer borrador con los requerimientos priorizados. Después debe preguntar sobre cada requerimiento, para evaluar su primer modelo de priorización:

- ¿Existe otra forma de satisfacer la necesidad que este requerimiento plantea?
- ¿Cuáles son las consecuencias de omitir o relegar este requerimiento?
- ¿Cuál sería el impacto en el objetivo de negocio del proyecto de demorar varios meses la implementación de este requerimiento?
- ¿Por qué el cliente estaría molesto si este requerimiento se desarrolla más tarde?
- ¿Es este requerimiento tan importante como los que quedaron calificados con este mismo nivel de criticidad?

La priorización de los requerimientos se puede hacer teniendo en cuenta dos variables: importancia y urgencia. La siguiente matriz muestra los niveles con los que debe ser calificado cada requerimiento:

	Importante	No tan importante
Urgente	Prioridad alta	No implementar
No tan urgente	Prioridad media	Prioridad baja

Tabla 26. Matriz de priorización de requerimientos de acuerdo a urgencia e importancia Fuente: Wiegers & Beatty, 2013.

Otro método para priorizar los requerimientos cuando las partes no se pueden poner de acuerdo fácilmente consiste en utilizar una matriz de evaluación como la que se muestra a continuación:

Pesos relativos	**P1=2**	**P2=1**			**P3=1**		**P4=0.5**		
Requerimiento	**Beneficio relativo**	**Penalidad relativa**	**Valor total**	**% de valor**	**Costo relativo**	**% de costo**	**Riesgo relativo**	**% de Riesgo**	**Priori-dad**
En esta columna liste los requerimientos que desea priorizar	Evaluación en una escala de 1 a 9 por parte de los clientes del beneficio que ofrece el requerimiento	Evaluación en una escala de 1 a 9 por parte de los clientes de impacto que tiene no implementar el requerimiento	Se calcula con la siguiente formula: $\boldsymbol{Valor\ tot} = (\boldsymbol{P1} \times \boldsymbol{beneficio\ relativo}) + (\boldsymbol{P2} \times \boldsymbol{penalidad\ relativa})$	Se calcula con la siguiente formula:: $\%\ \boldsymbol{valor} = \boldsymbol{valor\ total}/_{\boldsymbol{T3}}$	Evaluación en una escala de 1 a 9 por parte de los desarrolladores del grado de dificultad, tiempo y costo monetario que significa implementar este requerimiento	Se calcula con la siguiente formula:: $\%\ \boldsymbol{valor} = \boldsymbol{costo\ relativo}/_{\boldsymbol{T4}}$	Evaluación en una escala de 1 a 9 por parte de los desarrolladores de los riesgos técnicos derivados del requerimiento que llegarían a impedir que el requerimiento se implementara correctamente en el primer intento	Se calcula con la siguiente formula:: $\%\ \boldsymbol{valor} = \boldsymbol{Riesgo\ relativo}/_{\boldsymbol{T5}}$	Se calcula con la siguiente formula: $\boldsymbol{Prioridad} = \frac{\%\ \boldsymbol{de\ valor}}{\%\ \boldsymbol{de\ costo} + \%\ \boldsymbol{de\ riesgo}}$
1									
2									
Totales	T1	T2	T3	100%	T4	100%	T5	100%	

Tabla 27. Matriz de priorización de requerimientos de acuerdo con pesos. Fuente: Wiegers & Beatty, 2013.

Después de priorizar los requerimientos, se crea un diccionario de datos en el que se consignan los elementos de datos, estructuras y cualquier otro término o parámetro que permita aclarar los requerimientos descritos. El propósito de este diccionario es facilitar la comunicación entre todas las partes y la información que puede incluir se lista a continuación:

- Tipos de datos
- Valores permitidos
- Criterios de validación de datos
- Complementos al glosario del proyecto
- Acrónimos
- Términos técnicos o de negocio
- Abreviaciones

Durante el proceso de levantamiento, los requerimientos han sido agrupados según criterios de negocio, funcionalidad o de usuarios. Adicionalmente, al utilizar las herramientas de análisis se han empleado otros criterios de proyecto o de la naturaleza misma de los requerimientos para agruparlos. Como último paso del análisis, se sugiere hacer un proceso final de clasificación como el que se muestra en la tabla a continuación:

<table>
<tr><td colspan="5">Requerimientos del sistema</td></tr>
<tr><td colspan="5">Descomposición y derivación</td></tr>
<tr><td colspan="2">Requerimientos de software</td><td>Requerimientos manuales</td><td colspan="2">Requerimientos de Hardware</td></tr>
<tr><td colspan="2">Distribución</td><td>Distribución</td><td colspan="2">Distribución</td></tr>
<tr><td>Componente A</td><td>Componente B</td><td>Personas</td><td>Componente C</td><td>Componente D</td></tr>
</table>

Tabla 28. Matriz de clasificación de los requerimientos de un sistema y asignación de recursos. Fuente: Wiegers & Beatty, 2013.

4.7 ESPECIFICACIÓN DE LOS REQUERIMIENTOS

En el proceso de especificación de los requerimientos previamente levantados y analizados, se sugiere incluir una sola actividad clave que consiste en traducir la información recolectada en diagramas e información escrita que sea de fácil comprensión, revisión y uso para las audiencias a las que los documentos van dirigidos. El objetivo fundamental de la especificación de los requerimientos es documentar requerimientos de diferente naturaleza, de tal forma que sean consistentes, accesibles y entendibles para cualquier público.

La documentación de los requerimientos enfrenta los siguientes desafíos:

- Incluir los atributos descriptivos junto con los requerimientos.
- Planear la documentación desde un inicio, de tal forma que la gestión de cambios sea sencilla.
- Facilitar el seguimiento de las versiones, historias o transformaciones que ha tenido un requerimiento.
- Controlar las porciones de un requerimiento que están siendo incluidas en una iteración, mientras que el resto del contenido queda pendiente de ser implementado.
- Garantizar la trazabilidad.

Los requerimientos pueden se representados de varias formas, las más comunes son las siguientes tres:

Lenguaje natural: Los requerimientos usualmente se pueden representar en forma de casos de uso o de historias de usuario. Los casos de uso describen una secuencia de interacciones entre el sistema y un actor externo y tienen como resultado una salida con

valor agregado. Los casos de uso se escriben iniciando con un verbo y luego con un objeto, por ejemplo: revisar el estado de una orden.

En contraste, las historias de usuario describen las características deseadas de forma breve y concreta, desde la perspectiva del usuario. A continuación se muestra el formato que suele tener una historia de usuario:

Como __ (descripción del tipo de usuario)___, deseo __ (descripción características objetivo)___ con el fin de ___(descripción de la razón por la que se desea dicha característica)___.

La diferencia entre los dos tipos de representación es muy sutil inicialmente, pero se evidencia en los pasos que se deben seguir bajo cada modelo. La siguiente figura muestra las diferencias para cada tipo de representación:

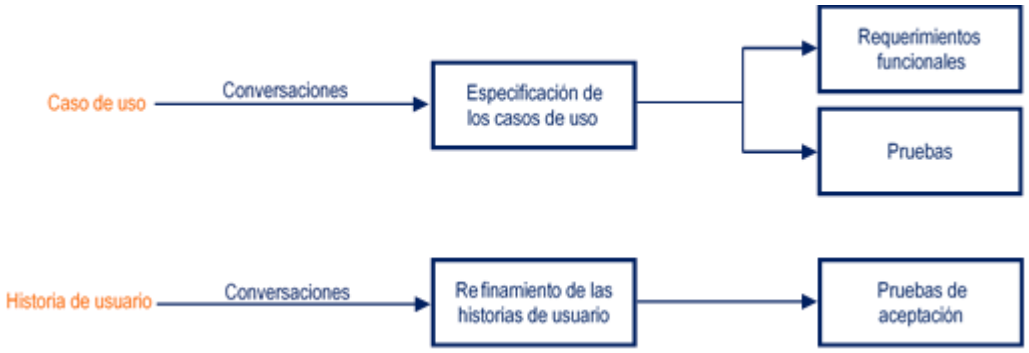


Figura 63. Pasos a seguir según el modelo de representación seleccionado. Fuente: Wiegers & Beatty, 2013.

Los casos de uso intentan responder tres preguntas:

- ¿Quién o qué es notificado cuando el evento ocurre en el sistema?
- ¿Quién o qué provee información o servicios al sistema?
- ¿Quién o qué ayuda a que el sistema responda y complete una tarea?

Los elementos esenciales de un caso de uso son los siguientes:

- Un identificador único para garantizar la trazabilidad
- Una descripción del objetivo del usuario
- Una breve descripción del objetivo del caso
- Una condición que dispara el inicio del caso
- Las condiciones que se deben dar para que el caso suceda
- La secuencia de interacción entre el usuario y el sistema.

La identificación de los casos de uso a partir de los requerimientos levantados se hace de la siguiente forma:

- Se identifican los actores
- Se identifican los procesos
- Se crea un escenario específico para ilustrar cada proceso de negocio
- Se limitan los actores para ese escenario específico
- Se analiza qué factores convierten las entradas en salidas para ese escenario específico
- Se identifican los factores externos a los que debe responder el sistema
- Se diagrama el flujo y se identifican los objetivos que persiguen los actores externos

Modelos visuales: como los que se explicaron en la sección de análisis, pero ajustados con los hallazgos hechos durante el análisis de los requerimientos.
Definición formal: a través del uso de lenguaje matemático preciso.

Sin importar cuál opción de representación se seleccione, se deben tener en cuenta los siguientes puntos para garantizar que en efecto se cuenta con requerimientos que pueden ser comprendidos por cualquier tipo de audiencia:

- Utilizar un formato preestablecido para organizar toda la información que sea necesaria.

Mejor práctica: *se sugiere tener en cuenta el siguiente esquema para organizar la información y los documentos que se han generado durante la fase de definición del alcance y requerimientos. Es muy importante determinar la estructura teniendo en cuenta las necesidades propias del proyecto y que el esquema sea personalizado de acuerdo con estas necesidades.*

1	**Introducción:** Descripción de todo el documento y el modo de usarlo			
	1.1	**Propósito:** Descripción de las audiencias a las que está dirigido el documento y qué sistema, iteración o versión cubren los requerimientos descritos		
	1.2	**Convenciones del documento:** Explicación de la metodología de etiquetado y las convenciones usadas en el texto para identificar cada elemento		
	1.3	**Alcance del proyecto:** Resumen breve de la descripción del sistema, iteración o versión y su objetivo		
	1.4	**Referencias:** Listado de los documentos usados como fuente		
2	**Descripción general:** Descripción del producto, ambiente en que será usado, posibles usuarios, limitaciones, supuestos y dependencias			
	2.1	**Perspectiva del producto:** Descripción del contexto del producto y su origen. Especifica si es un desarrollo nuevo, una versión actualizada, una iteración, etc.		
	2.2	**Clases de usuario y características:** Resumen de los hallazgos hechos durante la identificación y clasificación de usuarios		
	2.3	**Ambiente de operación:** Descripción del ambiente en que va a operar el sistema incluyendo: la plataforma de *hardware*, sistemas operativos, ubicación geográfica, servidores, bases de datos, sitios de internet, etc.		
	2.4	**Limitaciones de diseño e implementación:** descripción de las condiciones que limitan las opciones que pueden usar los desarrolladores en el proyecto		
	2.5	**Supuestos y dependencias:** Descripción de cualquier condición que se supone como cierta y de los componentes que están fuera de control y que generan dependencias para el desarrollo		
3	**Características del sistema:** Descripción de la forma en que serán presentados los requerimientos funcionales: por sistema, área funcional, proceso, caso de uso, clase de usuario, etc.			
	3.1	**Característica del sistema X**: Especificación del sistema, área funcional, proceso, caso de uso, etc. que será utilizado		
		3.1.1	**Descripción:** Descripción del sistema, el área funcional, proceso, caso de uso, etc.	
		3.1.2	**Requerimientos funcionales:** Descripción de los requisitos funcionales asociados a la característica identificada previamente Nota: La numeración en esa sección debe estar alineada con la metodología seleccionada.	

4	**Requerimientos de datos:** Descripción de los datos que serán consumidos como entradas, los datos que serán procesados por el sistema y los datos que conformarán las salidas	
	4.1	**Modelo de datos lógico:** Descripción gráfica de objetos de datos y su relación con los sistemas
	4.2	**Diccionario de dato**: Definición de la composición de la estructura de datos y su significado, incluye además los tipos de datos, longitud de los datos, formato de los datos y valores permitidos
	4.3	**Reportes:** Si el sistema, versión o iteración va a generar reportes en esta sección, se deben listar dichos reportes junto con las características que deben tener
	4.4	**Adquisición de datos, integridad, retención y disposición:** En los casos en que sea relevante, se debe describir cómo se adquieren y mantienen los datos. Adicionalmente incorpore a esta sección cualquier requisito de integridad, protección de la información, almacenamiento, respaldo y disposición.
5	**Requisitos de interface externos:** Descripción de los procesos de comunicación entre el sistema y otros elementos de *hardware* y *software*.	
	5.1	**Interface de usuario:** Descripción de las características que cada usuario requiere. Tenga en cuenta elementos como: tipo de fuente, tamaño de la pantalla, botones, imágenes, gama de colores, secuencias, derechos de autor, hipervínculos de navegación, ventanas emergentes, tipo de lenguaje y requisitos de personas en situación de discapacidad.
	5.2	**Interface de *software*:** Descripción de las conexiones entre el sistema, versión o iteración y otros componentes de *software*. Los otros componentes de *software* pueden ser bases de datos, sistemas operativos, herramientas, librerías, sitios web, etc.
	5.3	**Interface de hardware:** Descripción de las conexiones entre el sistema, versión o iteración y otros componentes de *hardware*. La descripción debe incluir tipos de dispositivos, interacciones de información y control, protocolos de comunicación, entradas, salidas, etc.
	5.4	**Interface de comunicaciones:** Describe los requisitos para cualquier función de comunicación que será usada por el sistema, versión o iteración. La descripción debe incluir cualquier aspecto de seguridad, encriptación, sincronización, etc. que se deba tener en cuenta.
6	**Atributos de calidad:** Esta sección contiene los requisitos no funcionales o limitaciones del sistema, versión o iteración. Los atributos definidos en esta sección deben ser específicos, cuantificables y verificables.	
	6.1	**Usabilidad:** Descripción de las condiciones que se requieren en términos de facilidad de uso, facilidad de aprendizaje, prevención de errores, eficiencia en las interacciones y accesibilidad.
	6.2	**Desempeño:** Descripción de las condiciones operativas de desempeño bajo los escenarios probables.
	6.3	**Seguridad:** Descripción de las condiciones de seguridad, privacidad, prevención de perdida de datos o mitigación de daño que se deban tener en cuenta.
7	**Otros requisitos:** Descripción de cualquier otro requisito que considere relevante para el sistema, versión o iteración.	

Tabla 29. Formato modelo para la documentación de requerimientos. Fuente: Wiegers & Beatty, 2013.

- Utilizar nombres y definir estilos para las secciones y subsecciones de tal forma que la estructura sea consistente.

- Utilizar herramientas para resaltar como negrilla, itálicas, subrayado, colores, etc., de forma consistente en toda la documentación.
- Utilizar tablas de contenido para que sea fácil ubicar la información necesaria.
- En lo posible, utilizar hipervínculos para que el lector pueda acceder a la información relevante rápidamente.
- Usar representaciones gráficas de la información para facilitar el proceso de comprensión.

A continuación se presentan las consideraciones básicas para tener en cuenta en el proceso de documentación de los requerimientos:

Etiquetado de los requerimientos: cada requerimiento debe contar con una referencia de identificación única que facilite el proceso de cambio, seguimiento a la historia de modificación y trazabilidad.

Numero de secuencia: para proyectos con baja complejidad, se sugiere utilizar un esquema de numeración secuencial precedido por dos letras que identifiquen el tipo de requerimiento. La siguiente tabla muestra un esquema básico de numeración:

Tipo de requerimiento	Esquema de numeración								
Requerimientos funcionales	RF-001	RF-002	RF-003						RF-00N
Requerimientos de negocio	RN-001	RN-002	RN-003						RN-00N
Requerimientos de usuario	RU-001								

Tabla 30. Numeración de requerimientos secuencial. Fuente: Corporación Colmbia Digital – CCD.

Numeración jerárquica: incluye información sobre la agrupación de los requerimientos y cómo están anidados. Se puede tener que un requerimiento puede estar numerado como RF-003 y a su vez se encuentra un requerimiento identificado como RF-003.2, en este caso

se concluye que el requisito funcional RF-003.2 es un hijo del requerimiento funcional RF-003 y que probablemente contiene un nivel más profundo o detallado.

Jerarquía de texto para etiquetas: esta técnica tiene en cuenta la clasificación temática que pueden tener los requerimientos, para que sea más fácil ubicarlo. Si por ejemplo se tiene un requerimiento que hace referencia a la información específica que debe tener un formato en un proceso determinado, el nombre de dicho requerimiento podría estar dado de la siguiente manera: Nombre_del_proceso.Nombre_del_formato.Campos_clave. Este esquema también permite anidar temas, de tal forma que los nombres reflejen la jerarquía, y permite definir nombres genéricos que agrupen o filtren por temas.

Consideraciones para información faltante: con frecuencia se encuentran casos en los que la información de un requisito no está completa. Es importante que en esos casos no se utilicen supuestos para completar, se sugiere llenar estos campos con un comentario que indique que la información está pendiente o no está disponible.

Elemento transversal – Riesgos: *Requerimientos ambiguos. Esto sucede cuando un requerimiento puede tener varias interpretaciones, lo que genera expectativas diferentes para cada uno de los interesados y compromete el éxito del proyecto.*

4.8 VALIDACIÓN DE REQUERIMIENTOS

En el último paso del proceso de desarrollo de requerimientos se debe validar que se cuenta con la información correcta para que los desarrolladores construyan una solución que cumpla los objetivos de la entidad. Esto se logra mediante la ejecución de las siguientes dos actividades claves:

- Revisar los documentos para corregir cualquier problema.

- Desarrollar criterios y pruebas de aceptación a partir de los requerimientos desarrollados, de tal forma que sea claro cuando se han cumplido los objetivos del producto y del negocio.

Mejor practica: *es posible que durante la validación de los requerimientos se encuentren contradicciones entre las solicitudes de diferentes clases de usuario. Como criterio de decisión no olvide tener en cuenta la importancia o criticidad que tiene cada clase de usuario identificado para el producto final*

4.9 CREACIÓN DE LA LÍNEA BASE

Elemento transversal – Riesgos: *Planeación poco adecuada. Cuando el desarrollo de requerimientos genera una línea base vaga y con poco nivel de detalle, se hacen estimaciones optimistas en términos de costos, esfuerzo y duración que pueden llevar al fracaso del proyecto.*

Después de terminar la validación de los requerimientos, es necesario hacer varios acuerdos para definir la línea base, como:

Acuerdo	Descripción
Aceptación de los requerimientos por parte de la entidad	La entidad debe acordar que los requerimientos que se construyeron en conjunto reflejan sus necesidades en ese momento de tiempo, versión o iteración.
Aceptación de los requerimientos por parte del desarrollador	El grupo o empresa desarrolladora debe aceptar que los requerimientos que se construyeron en conjunto fueron comprendidos en su totalidad y son realizables.
Aceptación de la viabilidad de las pruebas por parte de equipo de pruebas	El grupo o empresa a cargo de las pruebas debe aceptar que se pueden efectuar pruebas sobre los requerimientos para verificar su cumplimiento.
Aceptación de los requerimientos por parte del equipo de liderazgo	El equipo de liderazgo acuerda que los requerimientos que se construyeron en conjunto reflejan la necesidad de negocio, cumplen los objetivos planteados y se alinean con la estrategia de la entidad.

Tabla 31. Acuerdos mínimos para establecer la línea base. Fuente: Corporación Colombia Digital – CCD.

Mejor práctica: *el texto sugerido para incluir dentro de los acuerdos mínimos de la línea base debe siempre tener un lenguaje positivo y constructivo que contemple la naturaleza cambiante de los requerimientos. Se sugiere incluir en el documento por firmar, un texto como el del siguiente ejemplo:*

"Estoy de acuerdo con el grupo de requerimientos que a continuación se listan y describen, ya que representan nuestro (este pronombre personal hace referencia al grupo de interés que representa) mejor entendimiento de esta fase del proyecto (versión o iteración). Adicionalmente, confirmo que la solución que se describe a partir de estos requerimientos cumple con nuestras necesidades actuales. Estoy de acuerdo con formular cambios futuros sobre la línea base de requerimientos hoy definida, a través del procedimiento de gestión de cambios que se ha establecido para este proyecto. Reconozco que esos cambios pueden significar ajustes en términos de costos, alcance y tiempo".

En los casos en que no es posible llegar a un acuerdo sobre la línea base, se sugiere proceder con precaución, incorporar esta situación dentro de los riesgos identificados y documentar el impacto que puede tener el hecho de contar con requerimientos incorrectos o incompletos.

Elemento transversal – Documentación: *Los entregables que deben estar disponibles después de concluir la fase de desarrollo de requerimientos son:*

- *Requerimientos de negocio*
- *Requerimientos de usuario*
- *Requerimientos funcionales*
- *Requerimientos no funcionales*
- *Diccionario de datos*

- *Modelos de análisis*

Una vez los entregables anteriormente mencionados son revisados y aprobados, se constituye la línea base de los requerimientos. Es importante resaltar que al definir esta línea base, se deben tener en cuenta los acuerdos hechos previamente en la fase de definición del alcance y la visión del proyecto, como:

- Acuerdos sobre entregables durante el ciclo de vida del proyecto y sobre cada versión o iteración
- Limitaciones detectadas
- Cronograma de trabajo de todo el proyecto
- Horarios de trabajo
- Presupuesto
- Elementos contractuales

La línea base en proyectos que usan metodología ágil se debe gestionar de forma diferente y los requerimientos se documentan en forma de historias de usuario. La línea base se define seleccionando las historias de usuario que serán usadas en dicha iteración.

4.10 GESTIÓN DE REQUERIMIENTOS

Para mejorar la probabilidad de éxito de la subfase de gestión de requerimientos, se sugiere ejecutar cada una de las actividades que se describen a continuación:

- Definir una línea base de los requerimientos desarrollados, que sea aceptada por cada uno de los interesados y refleje las elecciones de diseño de una versión o iteración del producto final.
- Evaluar propuestas sobre nuevos requerimientos, teniendo en cuenta criterios como el impacto en términos de tiempo, presupuesto y alcance de los cambios propuestos.
- Mantener actualizado el progreso del cumplimiento de los requerimientos definidos.

- Negociar los requerimientos teniendo en cuenta la priorización que se hizo durante la subfase de desarrollo.
- Dar seguimiento a las relaciones y dependencias que existen entre los requerimientos.
- Garantizar la trazabilidad de los requerimientos hasta su diseño, código fuente y pruebas.
- Prever posibles riesgos asociados con los requerimientos y contar con alternativas de mitigación que minimicen su impacto sobre el proyecto.

La gestión de requerimientos se divide en cuatro actividades principales, como muestra la siguiente tabla:

Gestión de requerimientos			
Control de versiones	**Control de cambios**	**Seguimiento al estado de los requerimientos**	**Trazabilidad de los requerimientos**
Definición del método de identificación de cada versión	Propuestas de cambios	Definición de los posibles estados de un requerimiento	Definición de conexiones entre requerimientos
Seguimiento a las versiones individuales de cada requerimiento	Análisis de impacto	Registro del estado de un requerimiento	Definición de conexiones con elementos de otros sistemas.
Seguimiento a las versiones grupales de requerimientos	Toma de decisiones	Seguimiento individual de los cambios de estado	
	Actualización de requerimientos individuales		
	Actualización de grupos de requerimientos		
	Planes de actualización		
	Medición de la volatilidad de los requerimientos		

Tabla 32. Componentes de la gestión de requerimientos. Fuente: Wiegers & Beatty, 2013.

Con el fin de garantizar que haya empatía y una relación productiva entre el equipo a cargo del desarrollo y los interesados, se sugiere hacer una reunión de inicio de la subfase de desarrollo de requerimientos, en la que se adquieran formalmente los siguientes compromisos:

Compromisos por parte de los interesados en el desarrollo de software (Por ejemplo: usuarios finales, usuarios indirectos, gerentes de negocio, líderes de área, líderes de entidad, etc.)
Educar al equipo desarrollador sobre la naturaleza del negocio, lenguaje y procesos.
Nota: Para cumplir con este compromiso se sugiere agendar formalmente sesiones de entrenamiento y suministrar un glosario con los términos de negocio usados por la Entidad.
Dedicar el tiempo necesario a suministrar y a aclarar requerimientos.
Ser específico y preciso al suministrar información sobre los requerimientos.
Tomar decisiones dentro de los tiempos acordados.
Escuchar la opinión del desarrollador en cuento a costos y factibilidad de desarrollo de los requerimientos.
Trabajar de la mano con el equipo desarrollador para definir prioridades realistas.
Revisar los requerimientos y evaluar prototipos.
Establecer los criterios de aceptación.
Comunicar oportunamente cualquier cambio en los requerimientos.
Respetar el tiempo que tarda el desarrollo de un requerimiento.
Respetar el proceso definido para gestionar los cambios o actualizaciones de los requerimientos.
• Compromisos por parte del equipo desarrollador de software.
• Entender el negocio, su lenguaje y los procesos.
• Entender los objetivos de la Entidad y sus prioridades.
• Documentar los requerimientos de forma apropiada.
• Explicar los procesos de gestión de requerimientos y los entregables.
• Gestionar oportunamente las solicitudes de cambios en los requerimientos e informar las consecuencias para el proyecto.
• Suministrar alternativas para solucionar la necesidad de la Entidad.
• Identificar oportunidades de mejora que permita optimizar el proceso de desarrollo en términos de costos y tiempo.
• Identificar oportunidades de re-uso para acelerar el proceso de desarrollo.
• Proveer un sistema que cumpla con las expectativas de funcionalidad y calidad.

Tabla 33. Compromisos de la reunión de inicio. Fuente: Wiegers & Beatty, 2013.

La resistencia de alguno de los interesados a suministrar información, dar una aprobación o gestionar algún cambio, puede ser causada por: miedo, inseguridad, falta de información o intereses personales. Para manejar adecuadamente este tipo de situaciones es muy importante conocer la causa de resistencia, para luego determinar si la estrategia de acercamiento será a través de la creación de confianza en el proceso, la aclaración de malos entendidos o la educación.

Con respecto a los acuerdos que logran con los diferentes interesados al momento de definir la línea base, se sugiere no utilizar dichos acuerdos como una herramienta coercitiva o de protección frente a los eventuales problemas que puedan presentarse. El propósito de la línea base siempre debe ir orientado a ser un entendimiento común entre las partes del proyecto.

Elemento transversal – Talento TI: *el personal responsable de las actividades en la definición de los requerimientos y sus roles es el siguiente:*

Analista de requerimientos

La responsabilidad de los analistas es elaborar un catálogo detallado de requisitos que permita describir con precisión el sistema de información, para lo cual mantendrán entrevistas y sesiones de trabajo con los responsables de la organización y los usuarios, actuando de interlocutor entre estos y el equipo de proyecto, en lo que a requerimientos se refiere. Este catálogo permite elaborar los distintos modelos que sirven de base para el diseño, con lo que se obtienen los modelos de datos y de procesos, en el caso del análisis estructurado, y los modelos de clases e interacción de objetos, en el análisis orientado a objetos. Los analistas también realizan la especificación de las interfaces entre el sistema y el usuario.

Arquitecto de software

Durante la fase de requerimientos, el arquitecto de software se involucra con los requerimientos que influyen en la arquitectura ("drivers"), particularmente respecto con los atributos de calidad del sistema. El arquitecto se debe preocupar por que se identifiquen atributos de calidad pertinentes para el sistema, alineados con los objetivos de negocio, y por que las métricas asociadas estén justificadas. En el caso que la entidad solicite atributos de calidad con métricas muy demandantes (por ejemplo una disponibilidad del 99,99%), debe ser capaz de entender la justificación de esas métricas y debe poder negociar con la

entidad para establecer métricas adecuadas. Nuevamente, el arquitecto debe emplear aquí una combinación de habilidades "duras" y "suaves", con el fin de lograr una identificación adecuada de los requerimientos que influirán sobre el diseño arquitectónico.

5 DISEÑO Y ARQUITECTURA

5.1 PROBLEMAS FRECUENTES.

- En varias entidades se observó la necesidad de contar con una arquitectura empresarial y de *software* apropiadamente estructurados.
- Se detectó la necesidad de plantear los criterios básicos para seleccionar la metodología de desarrollo que mejor se adapte a los requerimientos identificados.
- Se evidenció que se deben robustecer los procesos de gestión de cambios durante todo el desarrollo de sistemas de información.
- Se encontraron algunas deficiencias en términos de infraestructura tecnológica que dificultan la óptima operación de los desarrollos.
- Se identificó la necesidad de plantear los criterios básicos para definir los atributos de calidad del proyecto.
- Se observó con frecuencia que la información utilizada está desactualizada o no tiene el nivel de detalle requerido.
- Se detectó la necesidad de contar con personal experto en arquitectura de *software*.

5.2 ASPECTOS GENERALES.

5.2.1 Diseño de *software*

Ésta es la actividad del ciclo de vida del *software* en la cual se analizan los requisitos para producir una descripción de la estructura interna del *software*, que sirva de base para su construcción. La salida o resultado es un conjunto de modelos y artefactos que registran las principales decisiones adoptadas.

Un diseño de *software* describe la arquitectura del *software* (cómo está descompuesto y organizado por componentes), las interfaces y los componentes a un nivel de detalle que facilite su construcción.

Una vez que se han establecido los requisitos del *software*, el diseño es la primera de tres actividades técnicas: diseño, codificación y prueba. Cada actividad transforma la información de forma que al final se obtiene un *software* validado.

El diseño es técnicamente la parte central de la ingeniería del *software*. Durante éste se desarrollan, revisan y documentan los refinamientos progresivos de las estructuras de datos, de la estructura del programa y de los detalles procedimentales. Como resultado del diseño se tienen las representaciones, cuya calidad puede ser evaluada.

Mejor práctica: *Las fases de diseño, codificación y prueba absorben el 75% o más del costo de la ingeniería del software, excluyendo el mantenimiento. Es este momento cuando se toman las decisiones que afectarán finalmente al éxito de la implementación del programa y, también, a la facilidad de mantenimiento que tendrá el software. Por tanto el diseño es un paso fundamental de la fase de desarrollo.*

Elemento transversal – Riesgos: *Sin diseño, el equipo se arriesga a construir un sistema inestable, que falle cuando se realicen pequeños cambios, que sea difícil de probar y cuya calidad no pueda ser evaluada hasta el final.*

5.2.2 Actividades del diseño de *software*

El diseño de software generalmente se realiza mediante dos procesos:

- El diseño arquitectónico (también referido como diseño de alto nivel, de nivel superior o preliminar), el cual describe cómo el *software* está organizado por componentes.
- El diseño detallado describe el comportamiento deseado de estos componentes.

La salida de estos dos procesos es un conjunto de modelos y artefactos que registra las grandes decisiones que se han tomado, junto con una explicación de los fundamentos.

Diseño Arquitectónico

Describe la estructura y organización de alto nivel, es decir, los subsistemas o componentes y sus relaciones. Es el primer paso en el diseño de un sistema, previo al diseño detallado, y su resultado se conoce como arquitectura del *software*.

Este proceso representa el enlace entre la especificación de requisitos y el diseño. Puede llevarse a cabo en paralelo con actividades de especificación de requisitos e implica un esfuerzo creativo, de forma que las actividades por realizar pueden cambiar según la naturaleza del sistema que se desarrollará.

Durante el proceso se debe dar respuesta a los siguientes interrogantes:

- ¿Existe una arquitectura genérica que pueda ser usada?
- ¿Cómo será distribuido el sistema?
- ¿Qué estilos arquitectónicos son apropiados?
- ¿Qué aproximación se utilizará para estructurar el sistema?
- ¿Cómo se descompondrá el sistema en módulos?
- ¿Qué estrategia de control se utilizará?
- ¿Cómo se evaluará el diseño arquitectural resultante?
- ¿Cómo se documentará la arquitectura?

Diseño Detallado

Describe cada componente y su comportamiento específico, de forma que se puede proceder con la construcción. Se ocupa del refinamiento de la representación arquitectónica, que lleva a una estructura de datos detallada y de las representaciones algorítmicas del *software*.

5.2.3 Principios del diseño de *software*

Los principios del diseño de *software* son:

Abstracción: es una visión de un objeto que se centra en la información relevante para un propósito particular y para prescindir del resto de la información.

Acoplamiento y cohesión: el acoplamiento se define como una medida de la interdependencia entre módulos en un programa, mientras que la cohesión se entiende como una medida de la fuerza de la asociación de los elementos dentro de un módulo.

Descomposición: consiste en dividir un *software* en varias unidades más pequeñas, usualmente con el fin de situar diferentes funcionalidades o responsabilidades en diversos componentes.

Encapsulamiento: se refiere a agrupar y empaquetar los elementos y detalles internos de una abstracción, para hacerlos inaccesibles desde fuera.

Separación de interfaz e implementación: define un componente especificando una interfaz pública conocida por otros componentes o clientes y separada de los detalles sobre cómo dicho componente está realizado (implementado).

Suficiencia y completitud: significa asegurar que un componente de *software* captura todas las características importantes de una abstracción y nada más.

Descomposición: hay dos tipos de descomposiciones: la primera que se refiere a la estructuración del sistema en subsistemas y la segunda que es la descomposición modular en la que los subsistemas se dividen en módulos.

Los aspectos principales que se enfrentan en el diseño de *software* son:

Concurrencia: se ocupa de descomponer el *software* en procesos, tareas y discusiones, y de abordar problemas con la eficiencia, atomicidad, sincronización y programación.
Control y manejo de eventos: este problema de diseño se ocupa de organizar los datos y flujos de control, así como del manejo de eventos reactivos y temporales a través de diversos mecanismos como la invocación implícita y devolución de llamadas (*call-backs*).
Perseverancia en los datos: se trata de cómo manejar datos de larga vida.
Distribución de componentes: este problema tiene que ver con la forma de distribuir el *software* en el *hardware* (incluyendo equipos informáticos y *hardware* de red) y cómo los componentes se comunican
Error y control de excepciones y tolerancia a fallos: se refiere a cómo prevenir y tolerar los errores de proceso y cómo hacer frente a condiciones excepcionales.
Interacción y presentación: este problema de diseño se ocupa de cómo estructurar y organizar la interacción con los usuarios, así como la presentación de la información.
Seguridad: tiene que ver con la forma de prevenir la divulgación no autorizada, creación, cambio, eliminación o la denegación del acceso a la información y otras fuentes.

5.2.4 Estructura del *software* y arquitectura

Una arquitectura de *software* es "el conjunto de las estructuras necesarias para razonar acerca del sistema, comprenden elementos de *software*, las relaciones entre ellos y las propiedades de ambos", (Swebok v3.0, 2014).

A mediados de la década de 1990, la arquitectura de *software* comenzó a surgir como una amplia disciplina que involucró el estudio de estructuras y arquitecturas en un campo más específico. Esto dio lugar a una serie de conceptos sobre el diseño de *software*, con diferentes niveles de abstracción. Algunos de estos conceptos pueden ser útiles durante el

diseño arquitectónico (por ejemplo los estilos arquitectónicos), así como en el diseño detallado (por ejemplo los patrones de diseño). Estos conceptos de diseño también pueden utilizarse para diseñar familias de programas, conocidas como líneas de productos.

Estilos arquitectónicos

Un estilo arquitectónico es "una especialización de elementos y tipos de relación, junto con un conjunto de restricciones sobre la forma en que se pueden utilizar", (Swebok v3.0, 2014).

Los beneficios de utilizar estilos son:

- Poder seleccionar una solución entendible y probada para ciertos problemas, definiendo los principios organizativos del sistema.
- Las personas entienden más fácilmente las características importantes de la arquitectura.

Algunos estilos son:

- Las estructuras generales (capas, tubos, filtros y pizarra)
- Los sistemas distribuidos (*client server*, tres niveles y agente)
- Los sistemas interactivos (Modelo-vista-controlador, presentación-abstracción-control)
- Los sistemas adaptables (*microkernel* y reflexión)
- Otros (lotes, intérpretes y proceso de control, basado en normas)

Patrones de diseño

Mientras los estilos arquitectónicos pueden ser vistos como patrones que describen la organización de alto nivel de *software*, otros patrones de diseño pueden utilizarse para describir los detalles en un nivel inferior.

Estos patrones de diseño son los siguientes:

- Patrones creacionales (constructor, fábrica y prototipo)
- Patrones estructurales (adaptador, puente, compuesto, decorador, fachada, peso mosca y apoderado)
- Patrones de comportamiento (comandos, intérprete, repetidor, mediador, recuerdo, observador, estado, estrategia, plantilla y visitante)

Familias de programas

Un enfoque para proporcionar la reutilización de diseños de *software* y de componentes es diseñar familias de programas o líneas de productos software. Esto se puede hacer mediante la identificación de los elementos comunes entre los miembros de estas familias y del diseño de componentes reutilizables y personalizables para dar variedad entre los miembros.

Diseño de la interfaz de usuario

El diseño de la interfaz de usuario es una parte esencial de la proceso de diseño de *software*. En el que se detallan los distintos componentes, tanto de hardware como de software, sus características y su disposición, que se utilizarán para interactuar con una serie de usuarios determinados en un medio ambiente determinado

Mejor práctica: *El diseño de la interfaz de usuario debe garantizar que la interacción entre el usuario y la máquina proporciona un control eficaz para la operación.*

Para que el *software* alcance su máximo potencial, la interfaz de usuario debe ser diseñada para que coincida con las habilidades, experiencia y expectativas del público previsto.

Éste es un proceso iterativo. Los prototipos de interfaz se utilizan a menudo para determinar las características, la organización y el aspecto del *software*. Incluye tres actividades centrales:

- Análisis del usuario: en esta fase el diseñador analiza las tareas de los usuarios, el medio ambiente de trabajo, otro *software* y cómo los usuarios interactúan con otras personas.
- Creación de prototipos de *software*: el desarrollo del prototipo ayuda a los usuarios de software a ver la evolución de la interfaz.
- Evaluación de la interfaz: los diseñadores pueden observar experiencias de los usuarios con la interfaz en evolución.

Principios de diseño de la interfaz de usuario

- Facilidad de aprendizaje: el *software* debe ser fácil de aprender, de manera que el usuario pueda empezar a trabajar rápidamente.
- Familiaridad del usuario: la interfaz debe utilizar términos y conceptos extraídos de las experiencias de las personas que utilizarán el *software*.
- Coherencia: la interfaz debe ser coherente de modo que las operaciones comparables se activan de la misma manera.
- Mínima sorpresa: el comportamiento de *software* no debe sorprender a los usuarios.
- Recuperabilidad: la interfaz debe proporcionar mecanismos que permitan a los usuarios recuperar información de errores.
- Guía del usuario: La interfaz debe dar retroalimentación significativa cuando se producen errores y proporcionar ayuda relacionada con el contexto.
- La diversidad de usuario: la interfaz debe proporcionar mecanismos de interacción adecuados para diversos tipos de usuarios y con diferentes capacidades: ciegos, problemas de visión, sordo, daltónico, etc.

Mejor práctica: *El diseño de la interfaz de usuario debe resolver dos cuestiones clave:*

- *¿Cómo el usuario debe interactuar con el software?*
- *¿Cómo la información desde el software debe ser presentada al usuario?*

Modalidades de interacción con el usuario

La interacción del usuario implica dar órdenes y proporcionar datos asociados al *software*. Se pueden clasificar en los siguientes estilos principales:

- Pregunta-respuesta: La interacción es esencialmente restringida a una sola pregunta-respuesta entre el usuario y el *software*. El usuario emite una pregunta y el *software* la responde.
- Manipulación directa: Los usuarios interactúan con objetos en la pantalla. La manipulación a menudo incluye un señalador dispositivo tal como un ratón, *trackball* o un señalador en las pantallas táctiles.
- Selección de menú: El usuario elige un comando entre una lista de menú de comandos.
- Formulario de relleno: El usuario diligencia los campos de un formulario. Para los campos que incluyen menús, debe haber botones que indiquen la acción que deber realizar el usuario.
- Lenguaje de comandos: El usuario emite un comando y proporciona parámetros para dirigir el *software*.

Presentación de la información

Puede ser de naturaleza textual o gráfica, de acuerdo con el estilo de presentación de la información. Los diseñadores pueden utilizar colores para mejorar la interfaz. Existen varias pautas importantes:

- Limite el número de colores utilizados
- Utilice el cambio de color para mostrar el cambio de estado
- Use códigos de colores para apoyar la tarea del usuario
- Use códigos de colores en una manera reflexiva y coherente
- Use colores para facilitar el acceso de las personas con daltonismo o deficiencia de color, por ejemplo utilice el cambio de la saturación y el brillo del color, trate de evitar combinaciones de azules y rojos.

- No dependa sólo del color para transmitir información importante para los usuarios con diferentes discapacidades.

Métricas

Las medidas pueden ser utilizadas para evaluar o para estimar cuantitativamente diversos aspectos de un diseño de *software*, por ejemplo el tamaño, estructura y calidad. La mayoría de las medidas que se han propuesto dependen del enfoque utilizado para la producción del diseño.

Estas medidas se clasifican en dos grandes categorías:

- Basado en las funciones: son medidas obtenidas mediante la descomposición de análisis funcional; generalmente representadas utilizando un diagrama de estructura (a veces llamado diagrama jerárquico), en el que diversas medidas pueden ser calculadas.
- Orientados a objetos: la estructura de diseño típicamente se representa como una clase de diagrama en el que diversas medidas pueden ser calculadas.

Descripciones estructurales

Aspectos para describir y representar los principales componentes y cómo están interconectados:

- Descripción de la arquitectura de idiomas (avd): Textual y formal, se utilizan para describir la arquitectura de *software* en términos de componentes y conectores.
- Clase y objeto: Diagramas utilizados para representar un conjunto de clases, de objetos y sus interrelaciones.
- Diagramas de componentes: Representa un conjunto de componentes y sus interrelaciones.
- Diagramas de implementación: Representa un conjunto de nodos y sus interrelaciones.
- Diagramas de entidad-relación (ERD): Son utilizados para representar los modelos conceptuales de datos almacenados en los repositorios de información.

- Lenguajes de descripción de interfaz (IDLS): Describen la interfaz de los componentes de *software*.
- Gráficos de estructura: Se utilizan para describir la estructura de los programas.

Descripción del comportamiento

Las siguientes notaciones y lenguajes se utilizan para describir el comportamiento dinámico de los sistemas de *software* y sus componentes. Muchas de estas notaciones son útiles sobre todo en el diseño. Las descripciones de comportamiento pueden incluir una justificación de la decisión de diseño de acuerdo con el cumplimiento de los requisitos de seguridad.

- Diagramas de actividad: Se usan para mostrar el flujo de control de una actividad.
- Diagramas de comunicación: Muestran las interacciones que se producen entre un grupo de los objetos.
- Diagramas de flujo de datos (DFD): Se utilizan para mostrar el flujo de datos entre los elementos. Estos flujos pueden ser empleados para el análisis de la seguridad, ya que ofrecen la identificación de posibles caminos para el ataque y la divulgación de información de carácter confidencial.
- Tablas de decisión y diagramas: Representan combinaciones complejas de condiciones y acciones.
- Diagramas de flujo: Usados para representar el flujo de control y las acciones asociadas.
- Diagramas de secuencia: Muestran las interacciones entre un grupo de objetos.
- Idiomas de especificación formal: Lenguajes textuales que utilizan las nociones básicas de matemáticas (por ejemplo la lógica, el conjunto o la secuencia) para definir con rigor y de forma abstracta las interfaces de componentes y el comportamiento del *software*, a menudo en términos de condiciones previas y posteriores.
- Seudocódigo y lenguajes de programas de diseño (PDL): Se usan para describir, generalmente en la etapa de diseño detallado, el comportamiento de un procedimiento o método.

Estrategias y métodos para el diseño de *software*

Existen varias estrategias generales para ayudar a guiar el proceso de diseño. Algunos ejemplos citados con frecuencia incluyen estrategias de refinamiento por pasos, estrategias de arriba hacia abajo versus de abajo hacia arriba, y estrategias que usan la heurística, modelos y patrones de idiomas.

Diseño estructurado

Este es uno de los métodos clásicos de diseño de *software*, se utiliza generalmente después de un análisis estructurado, con lo que se produce (entre otras cosas) los diagramas de flujo de datos y las descripciones asociadas a los procesos.

5.3 GESTION DE LA CALIDAD

A lo largo del proceso, la calidad del diseño se evalúa mediante una serie de revisiones técnicas formales cuyos objetivos son:

- Descubrir los errores en la función, la lógica o la implementación de cualquier representación del *software*.
- Verificar que el *software* alcance sus requisitos.
- Garantizar que el *software* sea representado según los estándares establecidos.
- Conseguir que el *software* sea desarrollado de manera uniforme.
- Hacer que los proyectos sean manejables.

A continuación se lista una serie de criterios para determinar la calidad del *software*:

- Un diseño debe tener una organización jerárquica.
- Un diseño debe ser modular, es decir que el *software* debe estar dividido en elementos que realicen funciones específicas.
- Un diseño debe tener representaciones distintas y separadas de los datos y de los procedimientos.

- Un diseño debe llevar a módulos que exhiban características funcionales independientes.
- Un diseño debe conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el exterior.
- Un diseño debe obtenerse mediante un método que sea reproducible y que esté dirigido a la información obtenida durante el análisis de requerimientos.

Varios atributos contribuyen a la calidad de un diseño de *software* como son mantenibilidad, portabilidad, capacidad de prueba, usabilidad, corrección y robustez. De igual manera, durante el tiempo de ejecución se deben mantener los atributos de rendimiento, seguridad, disponibilidad, funcionalidad, usabilidad, modificabilidad, portabilidad, reutilización, la capacidad de prueba y aquellos otros atributos relacionados con la arquitectura de cualidades intrínsecas como la integridad conceptual, corrección e integridad.

5.3.1 Análisis de calidad y técnicas de evaluación

Diversas herramientas y técnicas pueden ayudar en el análisis y la evaluación de la calidad del diseño de *software*:

- Revisiones de diseño: Son técnicas para determinar la calidad de artefactos de diseño como: comentarios, revisiones de diseño, inspecciones y técnicas basadas en escenarios. También puede evaluar la seguridad.
- Análisis estático: Es una verificación cruzada automatizada.
- Análisis de la vulnerabilidad de diseño: Es un análisis estático de las vulnerabilidades de seguridad.
- Análisis de diseño formal: Puede ser utilizado para detectar especificaciones residuales y errores de diseño, tal vez causados por la imprecisión, ambigüedad o cualquier otro tipo de errores.
- Simulación y prototipos: Es la validación de simulaciones y viabilidad de los prototipos diseñados.

5.4 HERRAMIENTAS

Es útil para las entidades saber que existen herramientas en el mercado, tanto Open Source como Propietarias, entre la cuales están:

Enterprise Architect

Es una herramienta para modelamiento visual y diseño de *software* basada en OMG UML. Con Enterprise Architect es posible modelar sistemas de información y procesos de negocio. Soporta estándares tales como: UML 2.5, BPMN 2.0, BPEL, WSDL, DDS y GML, entre otros.. Puede ser tenido en cuenta para realizar bosquejos de los componentes del sistema de información o pieza de *software* a ser desarrollada, sus interacciones, relaciones, restricciones, condiciones, artefactos.

Archimate

Es un marco de trabajo en una herramienta con la cual se diseñan procesos de negocio, estructuras organizacionales, flujos de información, sistemas de información e infraestructura tecnológica.

Archimate está basado en el estándar de The Open Group IEEE1471 (sobre arquitectura de *software*) y es usado comúnmente para definir arquitecturas empresariales en marcos de trabajo como Togaf y Zachman.

Es recomendable usar para construir diagramas que visualicen y faciliten la representación de elementos del diseño del sistema de información o *software* que se planea desarrollar, tales como flujos de proceso, estado de transacciones, operaciones entre componentes, intercambio de información, relación con la infraestructura.

IBM Rational Software Architect

Es una herramienta de diseño y modelado para la entrega global de *software*. Utiliza Lenguaje Unificado de Modelado (UML) para diseñar servicios web y aplicaciones. Pertenece a la familia de herramientas de desarrollo de IBM, por lo tanto permite integrarse con otros componentes de dicho fabricante.

Es una herramienta útil, al igual que las dos anteriores, para modelar y diseñar componentes, interacciones y flujos de información del sistema a desarrollar.

Kunagi

Es una herramienta web que proporciona un sistema de gestión de proyectos basado en Scrum. Ofrece herramientas colaborativas y otras facilidades como un cuadro de mando del proyecto, panel interactivo para los sprints y soporte para la estimación con Planning Poker.

En el caso de que el proyecto adopte la metodología ágil (*Scrum*), Kunagi es una opción con la cual facilitar actividades de gestión de tareas para seguimiento de *sprints*, planeación y estimación.

ScrumDo

Herramienta web Scrum muy centrada en la simplicidad y en la facilidad de uso. Permite gestionar las listas de tareas, crear y gestionar iteraciones, obtener informes de avance y facilitar la estimación con Planning Poker.

IceScrum

Herramienta Scrum y Kanban (Kanban: método para gestionar el trabajo intelectual, con énfasis en la entrega justo a tiempo, mientras no se sobrecarga a los miembros del equipo). Permite programar y gestionar *sprints* y asociarlos a la historia de los usuarios para facilitar la aplicación de técnicas de estimación.

Pango Scrum

Herramienta web para planificación, gestión y seguimiento de proyectos Scrum, permite definir *sprints*, actividades, tiempos estimados y priorizaciones.

Tanto Pango *Scrum*, como IceScrum, ScrumDo y Kunagi son herramientas que apoyan la gestión de proyectos basados en metodología ágil *Scrum*. Debe seleccionarse la que por facilidad, conveniencia y practicidad se ajusten a las necesidades de la entidad.

Elemento transversal – Talento TI: *El personal responsable de las actividades del diseño, arquitectura y sus roles son los siguientes:*

Arquitecto de software

Tiene la responsabilidad general de conducir las principales decisiones técnicas, expresadas en la arquitectura de software. Por lo general, para esto debe identificar y documentar la arquitectura de los aspectos importantes del sistema, incluidos los requisitos, diseños, implementación y despliegue, es decir, las vistas del sistema.

El arquitecto se encarga también de proporcionar la justificación de estas decisiones, buscar el equilibrio entre los participantes interesados, haciendo disminuir los riesgos técnicos y garantizando que las decisiones sean comunicadas, validadas y adoptadas efectivamente.

El arquitecto de software debe poseer la madurez, visión y experiencia que permiten comprender los problemas de manera rápida y tener un juicio crítico cuando existe información incompleta.

La visión general del arquitecto de software se muestra en la siguiente figura:

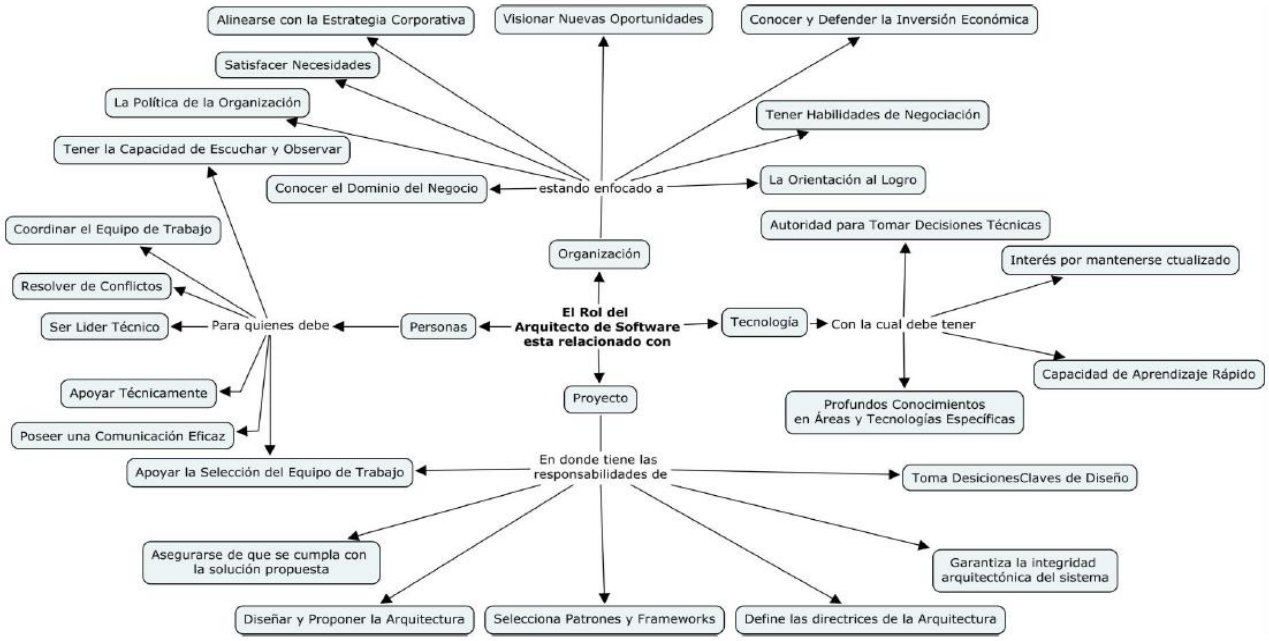


Figura 64. Rol del arquitecto de software. Fuente: Universidad EAFIT, 2008.

6 CODIFICACION

6.1 PROBLEMAS FRECUENTES.

- La alta rotación de personal afecta la calidad y el cronograma de entrega de los productos a las entidades.
- Se observa con alguna frecuencia que a los proyectos, los proveedores asignan recursos humanos con poca experiencia en desarrollo de *software*.
- No se definen las pruebas requeridas para asegurar la calidad de los desarrollos.
- El proceso de pruebas se ejecuta en las etapas finales del proyecto, no se tienen en cuenta en la fase de desarrollo.
- Baja calidad del *software*.

- Los documentos de requerimientos de sistema y de *software* están mal diseñados, lo que conlleva a que los desarrolladores no ejecuten bien su tarea.

6.2 ASPECTOS GENERALES.

La construcción de *software* hace referencia a la creación detallada de *software*, que trabaja a través de una combinación de codificación, verificación, pruebas unitarias, pruebas de integración y depuración.

Una vez que han sido diseñados los algoritmos de una aplicación, se puede iniciar la fase de codificación, etapa en la cual se deben traducir dichos algoritmos a un lenguaje de programación específico; es decir, las acciones definidas en los algoritmos hay que convertirlas en instrucciones.

Para codificar un algoritmo hay que conocer la sintaxis del lenguaje al que se va a traducir. Sin embargo, independientemente del lenguaje de programación en que esté escrito un programa, será su algoritmo el que determine su lógica. Justamente es la lógica de un programa la que establece cuáles son sus acciones y en qué orden se deben ejecutar. Por lo tanto, es conveniente que todo programador aprenda a diseñar algoritmos antes de pasar a la fase de codificación.

Durante la fase de programación, el código puede adoptar varios estados, de acuerdo con la forma de trabajo y del lenguaje elegido, estos pueden ser:

Código fuente: es el código escrito directamente por los programadores en editores de texto, que genera el programa. Contiene el conjunto de instrucciones codificadas en algún lenguaje de alto nivel. Puede estar distribuido en paquetes, procedimientos, bibliotecas de fuentes, etc.

Código objeto: es el código binario o intermedio resultado de procesar con un compilador el código fuente. El código objeto no es inteligible para el ser humano, normalmente está en formato binario, tampoco es directamente ejecutable para la computadora.

Principios fundamentales:

Minimizar la complejidad

- Al escribir el código sencillo y fácil de leer
- Utilizar estándares
- Emplear técnicas de codificación
- Implementar técnicas de aseguramiento de la calidad

Anticipar los cambios

- El *software* se ve afectado por los cambios en su entorno y está destinado a cambiar con el tiempo.

Pensar en la verificación posterior

- Construir el *software* de tal forma que los fallos puedan ser encontrados lo más pronto posible al codificar, hacer pruebas u operar el sistema.
- Técnicas:
  - Seguir estándares de codificación
  - Hacer pruebas unitarias
  - Organizar el código para soportar pruebas automatizadas
  - Restringir el uso de técnicas complejas

Aplicar estándares

- Directamente a la construcción del *software*
- A los formatos de comunicación: documentos y contenidos

- En las versiones de los lenguajes de programación (Java, .net, etc.)
- En las reglas de codificación (nombres de variables, comentarios, etc)
- A las notaciones de diagramas (UML)
- Durante el intercambio entre herramientas (XML)

La gestión de los diversos cambios que se realizan sobre los elementos de un producto de *software* o la configuración del mismo, es denominada control de versiones. Es aconsejable contar con un sistema y organización para el control de versiones (*Version Control System* - VCS). Dichos sistemas permiten administrar las distintas ediciones de cada producto desarrollado.

Este tipo de control se realiza principalmente para controlar el código fuente y dan lugar a los sistemas de control del mismo nombre (*Source Code Management* - SCM).

Existen herramientas tanto libres como propietarias para facilitar la gestión de versiones de código fuente, algunas son: CVS, Subversion, SourceSafe, ClearCase, Darcs, Bazaar, Plastic, SCM, Git, Mercurial y Perforce, entre otros.

Existen términos claves y de uso común en la gestión de control de versiones:

Línea base (Baseline): es una revisión o versión aprobada de un archivo de código fuente, a partir de la cual se deben realizar cambios subsiguientes.

Repositorio: es el lugar en el que se almacenan los datos actualizados e históricos de cambios, comúnmente en un servidor.

Rotular (tag): se le asigna a una versión del módulo en desarrollo en un momento preciso, con un nombre común (“etiqueta” o “rótulo”), para asegurar que posteriormente se encuentre ese desarrollo.

Abrir rama ("ramificar" – "Branch"): un módulo puede ser *branched* o bifurcado en un instante, de forma que desde ese momento en adelante se tengan dos copias que evolucionan de forma independiente siguiendo su propia línea de desarrollo. El módulo tiene entonces dos o más ramas. Algunos de los motivos habituales para la creación de ramas son la creación de nuevas funcionalidades y la corrección de errores.

Publicar ("commit"): un *commit* sucede cuando una copia de los cambios hechos para tener una copia local, es escrita o integrada sobre el repositorio.

Integración o fusión ("merge"): una integración une dos conjuntos de cambios sobre el archivo o un conjunto de archivos en una revisión unificada de los mismos. Se utiliza para unificar versiones de desarrollo de un mismo archivo de tal manera que queden integrados en un solo archivo.

Desplegar ("check out"): un despliegue crea una copia local del trabajo, desde el repositorio. Le permite al desarrollador trabajar de forma descentralizada del repositorio.

Existen ciertas arquitecturas de almacenamiento que utilizan los sistemas de control de versiones. Una de ellas es la centralizada, en la que existe un repositorio con todo el código, tiene un único usuario como responsable, quien debe aprobar las actualizaciones que quieran adelantarse. Esta arquitectura reduce flexibilidad, pero facilita la administración y el control.

La arquitectura distribuida permite a cada usuario contar con su propio repositorio. Los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos.

Versionamiento del *software*

Es un proceso de asignación de un nombre, código o número único a un *software*, para indicar su nivel de desarrollo. Generalmente se asignan dos números, que se van incrementando a medida que el desarrollo de *software* aumenta y se requiera asignación de un nuevo nombre, código o número.

Generalmente se aumenta un número cuando:

- Mayor: el *software* sufre grandes cambios y mejoras.
- Menor: el *software* sufre pequeños cambios o correcciones de errores.

Mejor práctica: *Escriba comentarios sólo para el código que es difícil de entender. Procure no escribir código que sea difícil de entender.*

Elemento transversal – Talento TI: *el personal responsable de las actividades de codificación incluye los siguientes roles:*

Desarrollador de software

Los programadores deben convertir la especificación del sistema en código fuente ejecutable, utilizando uno o más lenguajes de programación, así como herramientas de software de apoyo a la programación.

El éxito del desarrollo de software depende altamente del conocimiento, que no sólo corresponde a habilidades de programación y de administración de proyectos, sino también a una percepción y entendimiento de los últimos desarrollos de la industria.

Arquitecto de software

En esta etapa, desde un punto de vista técnico, el arquitecto debe completar las partes faltantes del diseño de la arquitectura y corregir las decisiones previas que hayan resultado ser equivocadas. Desde un punto de vista no técnico, el esfuerzo aumenta pues el arquitecto debe enfocarse en cuidar que el sistema se desarrolle de acuerdo con la arquitectura que

se definió para el mismo. Aquí el arquitecto juega un papel de mentor y muchas veces debe explicar cuestiones del diseño del sistema al equipo de desarrollo.

El arquitecto puede también realizar actividades de aseguramiento de la calidad, por ejemplo inspecciones de productos de trabajo, ya que su nivel técnico y conocimiento del dominio del problema le dan una ventaja para identificar problemas, que posiblemente podrían no ser identificados por ingenieros con un nivel técnico y menor conocimiento. En el momento de realizar pruebas del sistema, la participación del arquitecto es importante, en particular al realizar pruebas relativas a los atributos de calidad del sistema.

6.3 GESTION DE LA CALIDAD

La importancia de la verificación y validación a la hora de asegurar la calidad del producto durante su ciclo de vida, en especial durante su desarrollo, minimiza el riesgo de tener *software* de mala calidad, por lo que es necesario tener claro el concepto de cada uno de ellos y el nivel de pruebas requeridas para tal fin.

La verificación comprueba que el producto cumple con los requisitos establecidos y ayuda a asegurar que dicho producto se está desarrollando de la forma correcta. La verificación es un proceso incremental porque ocurre a lo largo del desarrollo, empieza con la comprobación de los requisitos, luego revisa la evolución del requerimiento y finaliza con la verificación del producto completo. Sin embargo, la validación cambia el enfoque, evalúa el producto en función de las necesidades de la entidad para asegurar que se satisface la necesidad y uso por el cual se creó.

Las actividades de validación son similares a las de verificación, por ejemplo ambas llevan a cabo pruebas, análisis, inspecciones y normalmente se ejecutan de forma concurrente.

El término “nivel de pruebas” da un indicio del centro o foco de las pruebas y del tipo de problemas que tratan de descubrir. Los niveles más típicos son:

- Pruebas unitarias o de componente
- Pruebas de integración
- Pruebas de sistemas
- Pruebas de aceptación

Cada uno de estos niveles incluye pruebas diseñadas para encontrar problemas específicos en cada etapa de desarrollo. Estos niveles de prueba también pueden aplicarse a desarrollos iterativos y pueden cambiar dependiendo del tipo de sistema. Por ejemplo, si el sistema incluye *software* desarrollado por alguien externo, las pruebas de aceptación serán llevadas a cabo antes de unirlo al sistema.

6.3.1 Pruebas de componentes o unitarias

Los desarrolladores suelen escribir el código en unidades aisladas que se integran en posteriores etapas del desarrollo. Cada una de estas unidades se llama programas, módulos o componentes. Las pruebas unitarias intentan asegurar que cada unidad cumpla las especificaciones antes de integrarla. Además, comprueban que el código puede ser ejecutado sin problemas.

Las pruebas unitarias pueden realizarse de forma aislada del sistema, mediante el uso de trozos de código y *drivers* que reemplazan el *software* omitido o ausente y simulan la interconexión entre los componentes de una forma simple. Pueden incluir pruebas de funcionalidad y sobre características no funcionales, por ejemplo sobre el comportamiento del medio, el rendimiento o la robustez del sistema. Este tipo de pruebas suelen ser realizadas por el programador que escribe el código y que, usualmente, también escribe la especificación del programa, aunque algunas veces las realiza un programador diferente,

dependiendo del nivel de riesgo, para conseguir cierta independencia. Los defectos son corregidos tan pronto como se detectan, sin ningún tipo de registro de incidencias.

Una técnica usada en la metodología de desarrollo *extreme programming* (programación extrema) es preparar y automatizar casos de pruebas antes de comenzar la codificación. Esta técnica se conoce como desarrollo orientado a pruebas (*test-driven development*), es altamente iterativa y está basada en ciclos de desarrollo de casos de pruebas.

Objetivos	•Probar la **funcionalidad de un programa o unidad de código** para encontrar y corregir posibles defectos •Validar que la unidad/componente cumple con las especificaciones •Verificar que los componentes no tienen defectos y que los procedimientos de recuperación de errores y excepciones funcionan
KPI's	•Número de componentes probados •Estado de componentes (correcto/incorrecto)
Alcance	•Identificar y descubrir los componentes por probar •Establecer un plan de pruebas para cada componente •Ejecutar los casos de pruebas •Gestionar los defectos •Documentar los resultados de ejecución
Entregables	•Plan detallado de Pruebas de cada componente y resultados específicos •Casos de prueba actualizados •Informes de defectos hallados
Prerrequisitos	•Disponer y revisar los requisitos y especificaciones técnicas funcionales •Diseñar y codificar los módulos revisados •Plan General de Proyecto documentado, revisado y autorizado

Tabla 34. Proceso de pruebas unitarias. Fuente: Inteco. (2009). Guía de mejores prácticas de calidad de producto.

6.3.2 Pruebas de integración

Una vez que las unidades se han desarrollado, la siguiente fase es unirlas para crear el sistema. A este proceso se le llama integración y con él a partir de un pequeño número de piezas se consigue algo mayor y con más funcionalidad.

El propósito de las pruebas de integración es descubrir los defectos de las interconexiones y de la interacción entre sistemas o componentes integrados. Puede haber más de un nivel de integración y las pruebas pueden realizarse sobre objetos de tamaño variable. Por ejemplo:

- Las pruebas de integración de componentes revisan la interconexión entre los componentes de *software* y son ejecutadas por los desarrolladores después de las pruebas unitarias.
- Las pruebas de integración de sistemas inspeccionan la interconexión entre distintos sistemas y las ejecutan los técnicos de pruebas tras las pruebas individuales de cada sistema.

Antes de realizar cualquier prueba de integración, es necesario establecer una estrategia y decidir cómo unir las distintas partes de un sistema. Hay tres estrategias que se suelen usar comúnmente:

- Integración *Big-bang*: En la que todas las unidades se unen en una sola para formar un sistema completo. Esta técnica tiene la ventaja de que no es necesario simular ninguna parte porque todo está acabado antes de empezar las pruebas de integración. La mayor desventaja es, en general, el tiempo que se consume y la dificultad que supone realizar un seguimiento de las causas de los fallos de esta integración. Cuando se realizan pruebas en este tipo de integración, es realmente difícil conseguir aislar los errores encontrados, ya que no se presta atención a las interfaces entre unidades individuales.

Mejor práctica: *De esta forma, la integración big-bang es una buena idea cuando en la planificación del proyecto se es optimista y se espera que no haya problemas.*

- Integración *top-down*: En la que el sistema se construye por fases, empezando por los componentes que llaman a otros componentes. Este tipo de integración permitirá a los técnicos de pruebas evaluar la conexión entre los componentes iniciando desde arriba. La estructura de control de un programa puede representarse en una gráfica similar a la

que se muestra a continuación, en la que el componente 1 llama al componente 2 y 3, por esa razón el componente 1 aparece por encima de ellos:

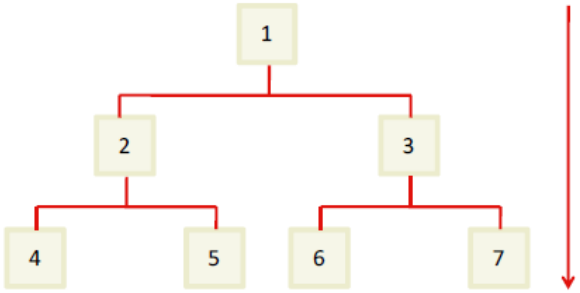


Figura 65. Estructura de control top-down. Fuente: Inteco. (2009). Guía de mejores prácticas de calidad de producto.

Las pruebas de integración *top-down* requieren que las interacciones sean probadas en el momento en que el componente sea construido, es decir, que aquellos componentes más bajos de la jerarquía puede que no hayan sido construidos ni integrados aún.
Si por ejemplo se quiere probar la interacción del componente 1 con el componente 2, puede ser necesario reemplazar el componente 2 por un trozo de código. Un trozo de código es un componente pasivo que es llamado por otro componente y es de gran ayuda en los casos en que se necesite reemplazar componentes aún no integrados.

- Integración *bottom-up* es lo contrario a la integración *top-down*. Los componentes son integrados en el orden 'de abajo hacia arriba' como muestra la siguiente figura:

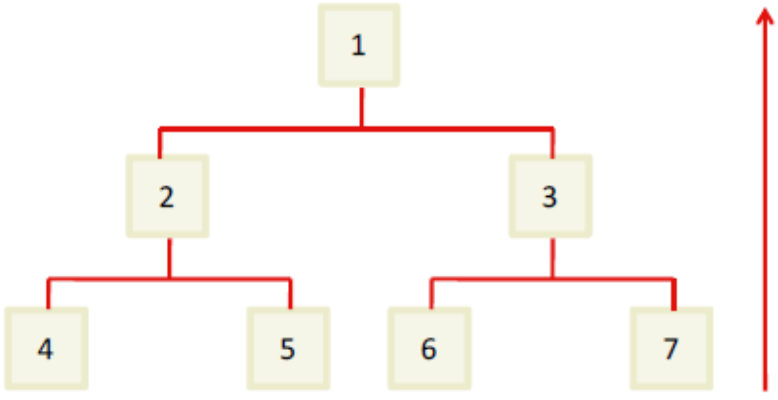


Figura 66. Estructura de control botton-up. Fuente: Inteco. (2009). Guía de mejores prácticas de calidad de producto.

El orden de integración puede ser: 4,2 - 5,2 - 6,3 – 7,3 – 2,1 – 3,1, de tal forma que los componentes del 4 al 7 deberían integrarse antes que los 2 y 3. En este caso, los componentes que deben ser sustituidos por componentes especiales son aquellos que llaman de forma activa a otros y que son conocidos como *drivers*, ya que en el programa controlan a otros componentes. En el ejemplo, los componentes 2 y 3 serán remplazados por *drivers* cuando se necesiten probar 4, 5, 6 o 7. Los *drivers* son generalmente más complejos que los trozos de código.

Mejor práctica: *La secuencia y número de pasos requeridos en la integración dependen de la ubicación de las interconexiones de alto riesgo dentro de la arquitectura. La mejor opción es empezar la integración con aquellas conexiones que se espera causen mayores problemas; de esta forma, se evitan encontrar los mayores defectos al final de la etapa de integración de pruebas. Con el objetivo de reducir este riesgo es aconsejable no usar la integración Big-bang.*

Mejor práctica: *Se recomienda que las personas que lleven a cabo las pruebas entiendan la arquitectura y realicen una buena planificación, ya que si las pruebas de integración se planifican antes de que los componentes y sistemas sean construidos, se elegirá el orden requerido para obtener las pruebas más eficientes.*

Tabla 35. Proceso de pruebas de integración. Fuente: Inteco. (2009). Guía de mejores prácticas de calidad de producto.

6.3.3 Pruebas de sistemas

Una vez se compruebe que los componentes trabajan correctamente a nivel unitario, el siguiente paso es considerar la funcionalidad desde otra perspectiva.

Las pruebas de sistemas se ocupan del comportamiento del sistema o del producto como un todo; incluyen pruebas basadas en riesgos y en la especificación de los requisitos, procesos de negocios, casos de uso u otras descripciones de alto nivel, interacciones con el sistema operativo y recursos del sistema. Estas pruebas suelen realizarse al final de las pruebas de desarrollo y verifican que el sistema entregado cumpla las especificaciones. Normalmente son llevadas a cabo por un equipo independiente de técnicos especializados en pruebas; al finalizar las pruebas, este grupo entrega los informes al director del proyecto.

Mejor práctica: *Si la entidad no cuenta con los recursos suficientes o la infraestructura necesaria, se recomienda que estas pruebas sean llevadas a cabo por un equipo externo o por analistas de negocio.*

El comportamiento del sistema está documentado en la especificación funcional, que debería contener definiciones tanto de los requisitos funcionales como de los no funcionales.

Un requisito funcional es aquel que especifica una función que debe realizar el sistema o el componente del sistema. Este tipo de requisitos puede ser específico para un determinado sistema y da detalles sobre lo que hará la aplicación.

Las pruebas de sistemas no funcionales tratan aspectos importantes, pero no directamente relacionados con las funciones que debería tener el sistema; tienden a ser requisitos genéricos que pueden aplicarse a diferentes sistemas.

Las pruebas de sistema deberían investigar tanto los requisitos de sistemas funcionales como de los no funcionales. Las pruebas de los requisitos funcionales usan técnicas basadas en especificación (o de caja negra) sobre el sistema probado. Las técnicas basadas en estructura suelen usarse para valorar la minuciosidad de los elementos de pruebas.

Las pruebas de sistemas requieren un entorno controlado en lo que se refiere al manejo de versiones del *software* o pruebas de datos, entre otras cosas. El entorno de pruebas debería corresponder al objetivo final o al entorno de producción tanto como sea posible, para minimizar el riesgo de encontrar fallos específicos del entorno.

Objetivos	•Validar que los procesos de negocio han sido implementados y cumplen las especificaciones •Validar compatibilidad e integración de aplicaciones y sistemas que forman la plataforma integrada (solución final) •Validar las interfaces entre aplicaciones y sistemas, intercambio de datos entre componentes, recuperación tras errores, etc.
KPI's	•Número de pruebas de integración realizadas •Estado de las pruebas de integración
Alcance	•Identificar y realizar pruebas entre todos los elementos que forman el sistema final •Actualizar los casos de prueba en función de resultados •Gestionar los defectos •Documentar los resultados de ejecución
Entregables	•Plan de pruebas y resultado detallado •Actualizar los casos de prueba •Diligenciar los formularios de gestión de incidencias y control de cambios • Realizar los informes de defectos hallados
Prerrequisitos	•Plan de pruebas de integración, incluyendo escenarios, casos y datos de prueba preparado y aprobado. •Pruebas unitarias completadas y aceptadas •Requisitos y especificaciones de negocio y de sistema disponibles

Tabla 36. Proceso de pruebas de sistema. Fuente: Inteco. (2009). Guía de mejores prácticas de calidad de producto.

6.3.4 Pruebas de aceptación

Después de que el proveedor realice sus pruebas de sistemas y la mayoría de los ajustes, el sistema será entregado al usuario o a la entidad para que dé su aprobación. Es entonces cuando se realizan las pruebas de aceptación, cuyo objetivo es dar confianza al usuario final de que el sistema funcionará de acuerdo con sus expectativas, de las partes del sistema y sus características no funcionales.

Las pruebas de aceptación son a menudo responsabilidad del usuario o del cliente, aunque cualquier persona involucrada en el negocio puede realizarlas. La ejecución de estas pruebas requiere un entorno que represente al entorno de producción.

Dentro de las pruebas de aceptación para un sistema basado en el negocio, se pueden distinguir dos tipos principales de pruebas que son preparadas y ejecutadas normalmente por separado:

- Pruebas de aceptación de usuario: son llevadas a cabo por un usuario que valida que el sistema cumple con sus necesidades de negocio.
- Pruebas de aceptación operacional: validan que los procesos y procedimientos son adecuados para el uso y mantenimiento del sistema. Suelen incluir: facilidades para hacer copias de seguridad (*back-up*), procedimientos de recuperación ante desastres, formación de los usuarios finales, procedimientos de mantenimiento y procedimientos de seguridad.

En la mayoría de las entidades, las áreas de TI realizan las pruebas de aceptación operacionales antes de que el sistema sea liberado.

Objetivos	•Verificar que la plataforma cumple con los requisitos generales desde el punto de vista del usuario final •Confirmar si la plataforma está disponible para llevarse a producción
KPI's	•% Casos de pruebas de aceptación de usuarios probados •Estado de cada uno de los casos de pruebas de aceptación de usuario probados •Criterios de puesta en producción del sistema
Alcance	•Identificar y realizar pruebas funcionales desde el punto de vista del usuario •Actualizar los casos de prueba en función de resultados •Gestionar los defectos •Documentar los resultados de ejecución
Entregables	•Plan de Pruebas y resultado detallado •Actualizar los casos de prueba •Diligenciar los formularios de gestión de incidencias y control de cambios • Realizar los informes de defectos hallados
Pre requisitos	•Plan de pruebas de sistema completado y aprobado. •Disponer del Plan detallado de pruebas de aceptación de usuario •Disponer y revisar los requisitos y especificaciones técnicas y funcionales.

Tabla 37. Proceso de pruebas de aceptación. Fuente: Inteco. (2009). Guía de mejores prácticas de calidad de producto.

Elemento transversal – documentación: Plan de pruebas

El propósito de este plan es proyectar, estructurar y documentar las pruebas de aceptación, así como la estrategia para ejecutarlas.

Las pruebas de aceptación se realizan sobre los requerimientos funcionales y a los no funcionales, revisando la facilidad de uso, recuperación y eficiencia, entre otros. El plan que a continuación se detalla pretende dar una visión general sobre las actividades por realizar y los aspectos para tener en cuenta.

El documento debe contar mínimo con lo siguiente:

PLAN DE PRUEBAS DE ACEPTACIÓN			
Requerimientos de pruebas	Introducción		Indique qué se realizará con las pruebas
	Filosofía de pruebas	Generalidades	Documente los requerimientos durante la fase de pruebas del sistema e integración
		Áreas funcionales	Describa las áreas funcionales generales que deberán ser probadas como parte de la fase de pruebas del sistema
		Categorías de resultado de pruebas	Describa las categorías que pueden ser asignadas y los resultados en uncaso de prueba 1. Éxito: El resultado de la prueba es conforme con el resultado esperado. 2. Aceptable: El resultado de la prueba indica que el sistema difiere de la especificación aceptada, no son necesarios cambios en la aplicación, pero se requiere un cambio en la especificación funcional 3. Tolerable: El resultado de la prueba es incorrecto, la aplicación en prueba trabaja y podría ser aceptada, pero la falla deberá ser rectificada en el periodo acordado. 4. Intolerable: El resultado de la prueba es incorrecto y la falla debe ser corregida antes de concluir la fase de prueba. 5. Error: El resultado de la prueba es correcto, pero el resultado esperado de acuerdo a los scripts de prueba son incorrectos

Requerimientos de pruebas	Entorno de prueba	Generalidades	Indique una breve descripción del entorno de prueba
		Hardware	Indique los recusos de *hardware* necesarios
		Software	Indique los recusos de *software* necesarios
	Roles y responsabilidades del equipo de pruebas		
	Identificación de la prueba	Scripts de prueba	describa los pasos y los resultados esperados de cada prueba individual. En particular un *script* contiene la siguiente información: • Identificador de la prueba • Descripción del objetivo de la prueba • Descripción del estado de la aplicación antes de la prueba o precondiciones de la misma • Pasos precisos y no ambiguos para ejecutar la prueba • Descripción de los resultados esperados
		Reporte de resultados	Los resultados de la prueba son registrados en un formulario que contiene: • Nombre y versión de la aplicación en prueba • Fase de prueba • Fecha de prueba • Identificador único de prueba • Hora de ejecución de cada caso de prueba • Resultado observado durante la prueba • Categoría de resultado de prueba • Descripción del error • Firma del ejecutor y del observador de la prueba

Requerimientos de pruebas	Identificación de la prueba	Criterios de aceptación	Esta sección documenta la frecuencia de las categorías de los resultados de prueba que son consideradas para aceptar la aplicación y pasar con éxito la fase de prueba. Identificamos los siguientes criterios, que deben ser evaluados progresivamente: . • Requerimientos de prueba: ¿Todos los requerimientos del sistema han sido probados? • Pruebas cubiertas: ¿Todas las partes del *software* han sido probadas, incluyendo manejo de errores? • Medida de casos de prueba: ¿Cuántos casos de prueba han sido planeados, diseñados, implementados, ejecutados y pasaron con éxito o falla? • Defectos detectados en casos de prueba: Es importante tener un ratio de los defectos encontrados en los casos de prueba y de los defectos corregidos y mantenidos.

Requerimientos de pruebas	Identificación de la prueba	Errores de prueba	Esta sección especifica los procesos para alcanzar la corrección de los errores observados y registrados durante la prueba. Para cada error observado que requiera corrección de la aplicación o de la especificación de funcionalidades, el líder del equipo de prueba, el líder de desarrollo y sus respectivos equipos deben estar de acuerdo en: • El ámbito de trabajo adicional y las escalas de tiempo para la corrección • El caso de prueba requerido para ser ejecutado después de la corrección • Dada una falla, el principal responsable de realizar la corrección es el que se encargó de desarrollar dicho componente • Establecer prioridades de acuerdo con una serie de fallas
		Documentación de la prueba	Esta sección describe los documentos que deben ser generados durante la actividad de prueba. Estos documentos son los siguientes: • *Scripts* y casos de prueba • Resultados de pruebas siguiendo el formato especificado • Reporte consolidado de pruebas por módulo • Certificado de prueba para formalizar el hecho de que la aplicación en prueba ha pasado la prueba con éxito
Estrategia de pruebas	Indique la estrategia a seguir		
Casos de prueba	Liste los casos de prueba y los resultados esperados		

Tabla 38. Plan de pruebas de aceptación. Fuente: Corporación Colombia Digital.

Elemento transversal – Talento TI: *El personal responsable de las actividades de pruebas es el siguiente:*

Gerente de proyecto

Es el responsable, junto con el analista de pruebas, de generar el plan de pruebas y de coordinar, evaluar y registrar las pruebas de aceptación.

Analista de pruebas

A su cargo está todo el proceso de la planeación, ejecución, seguimiento y control del desarrollo de pruebas del sistema, debe enfocarse principalmente en:

- *Identificar y definir las pruebas necesarias.*
- *Facilitar los recursos necesarios para el desarrollo de las pruebas.*
- *Hacer el seguimiento detallado de las pruebas.*
- *Revisar los informes de pruebas entregados por el equipo.*

- *Recopilar y gestionar los datos de prueba en cada ciclo.*
- *Evaluar la calidad global como resultado de las actividades de prueba.*

6.3.5 Ambientes de pruebas y producción

El desarrollo de *software* hoy en día está caracterizado por múltiples equipos de proyectos que trabajan de forma simultánea, bajo cronogramas cada vez más exigentes y desarrollando sistemas que interoperan con otras aplicaciones y plataformas. Bajo un escenario como éste, la gestión de los ambientes (entornos) de pruebas integrales y de producción adquiere gran importancia para asegurar que el *software* sea puesto en producción con los niveles de calidad necesarios. Los ambientes de pruebas y producción deben ser diferentes, como lo muestra la siguiente figura:

Figura 67. Separación ambientes de pruebas y producción. Fuente: Corporación Colombia Digital.

6.3.5.1 Ambiente de pruebas

El objetivo del ambiente de pruebas es proveer la infraestructura de *hardware* y *software*, necesaria para ejecutar las pruebas de integración, sistema y funcionales que determinan el comportamiento de la aplicación en escenarios reales. Es recomendable que este ambiente sea muy similar al entorno de producción, para que permita obtener resultados más cercanos a la realidad.

Requisitos

- Contar con un servidor compartido por los equipos de pruebas que correspondan.

- Ser lo más idéntico posible a los ambientes de producción.
- No debe ser utilizado para actividades de desarrollo o producción.
- Los desarrolladores no deben tener privilegios de algún tipo de modificación en el ambiente de pruebas, esto para evitar cambios no informados en la configuración.
- Una versión se instala en producción sólo después de que ha sido revisada en ambiente de pruebas.

Restricciones

- Los desarrolladores no deben poseer privilegios de acceso de modificación de cualquier tipo en el ambiente de pruebas, esto para evitar cambios no informados en la configuración.
- Los desarrolladores no poseen herramientas de *software* o permisos de acceso especiales para ejecutar desarrollos de *software*, en su lugar, tiene una configuración similar o igual a la de producción.
- Sólo el administrador se encarga de instalar, actualizar o desplegar en el ambiente de pruebas, nunca el desarrollador directamente.
- Requiere estrictos controles de cambio que mantengan rastro de las modificaciones en la configuración, para asegurar resultados controlados y que el ambiente sea similar al de producción. Cualquier ciclo de pruebas que esté en proceso puede quedar invalidado por cambios en la configuración y, por ende, tiene que repetirse.
- Una versión se instala en producción sólo después que ha sido instalada y probada en ambiente de pruebas integrales.

Procedimiento de uso

- Definir las especificaciones de ambientes de prueba que necesita el equipo.
- Antes de iniciar cualquier ciclo de pruebas, debe verificarse que el ambiente ha sido configurado adecuadamente.

- No todas los test se ejecutan en el ambiente de pruebas integrales de *software*, por ejemplo, las pruebas de componente de *software*, con las que se revisa el código, se realizan en el ambiente de desarrollo.
- Si durante la ejecución de las pruebas integrales se realizan cambios de configuración, será necesario identificar las pruebas afectadas y ejecutar nuevamente dichos casos.
- Para proyectos de mantenimiento o migración de plataformas tecnológicas (por ejemplo un *upgrade* de base de datos), se deben incluir pruebas operacionales, para lo cual es necesario configurar y probar un ambiente similar al de producción.
- Ajustes al cronograma dada la disponibilidad de ambientes de pruebas.
- Los reportes de incidencias deben referirse al ambiente en el que se está probando y a cualquier aspecto específico de configuración.

6.3.5.2 Ambiente de producción

Corresponde al ambiente en el que finalmente se radicará la aplicación final o el programa desarrollado. La actividad más crítica es el paso a producción del *software*.

Después de haber realizado las pruebas de implantación y aceptación del sistema, es necesario que se disponga el entorno de producción perfectamente instalado en cuanto a *hardware* y *software* de base, componentes del nuevo sistema, procedimientos y manuales, ambientes totalmente aislados a los de desarrollo y pruebas.

Requisitos

- Contar con la aprobación del gerente del proyecto para el paso a producción.
- Crear ventanas de tiempo para el paso a producción, de tal modo que si se detecta algún problema, se disponga de un lapso para corregirlo antes de decidir una “vuelta atrás”.
- Nunca se debe pasar a un ambiente de producción sin haber finalizado la etapa de pruebas.
- Contar con la infraestructura tecnológica adecuada para realizar los despliegues.

- Contar con el personal idóneo para realizar los despliegues.

Otro factor importante para considerar en el despliegue de *software* seguro es el control del código fuente. Se suele liberar versiones de *software* sólo para reparar rápidamente incidencias en producción o sencillamente para agregar características, sin tomar en cuenta que el código liberado está en etapa de desarrollo o contiene secciones aún no probadas, que se piensan liberar en un futuro. En este sentido, es muy útil contar con herramientas de control de código fuente como el *team foundation server*, que nos permite crear ramas (*branch*) sobre nuestro código, y separar el código evolutivo del código que se desarrolla en forma de parches.

7 PRUEBAS

7.1 PROBLEMAS FRECUENTES.

- No existe una planificación adecuada en el desarrollo de las pruebas a los sistemas de información implantados en las entidades.
- Los ambientes de pruebas de *software* son la base principal para que los productos desarrollados sean entregados con calidad a las entidades, en este sentido se ha detectado una deficiencia en que no existen ambientes adecuados para el desarrollo de esta actividad, generando problemas posteriores en el uso de los sistemas de información.
- Carencia de procesos adecuados para realizar las pruebas necesarias al *software* desarrollado, así como los protocolo para realizar los planes y ejecución de pruebas que aseguren el cumplimiento de los criterios de aceptación establecidos y certifiquen los pasos a producción.
- Las entidades no cuentan con una estandarización de tipo y cantidad de pruebas a realizar al *software* en las diferentes fases del ciclo de vida del desarrollo.
- Las entidades no cuentan con personal idóneo y que desconocen el proceso de pruebas de *software*, lo que implica aceptación de productos sin la calidad requerida o con fallas que luego deben ser tramitados por controles de cambios generando sobrecostos a la entidad.
- Las entidades no dedican el tiempo necesario a los procesos de pruebas.

- La mayoría de errores detectados durante las pruebas corresponde a falencias en la etapa de definición de requerimientos, por lo que se hace necesario realizar pruebas durante todo el ciclo de vida de desarrollo de sistemas de información.
- El proceso de pruebas se ejecuta en las etapas finales del proyecto.
- No se cuenta con herramientas automatizadas para la ejecución de pruebas.

7.2 ASPECTOS GENERALES.

Hoy en día, debido al aumento del tamaño y la complejidad del *software*, el proceso de prueba se ha convertido en una tarea vital dentro del desarrollo de cualquier sistema de información.

Las pruebas constituyen un proceso con el objetivo principal de encontrar defectos en el software. Una prueba tiene éxito si descubre un defecto y fracasa si hay defectos pero no es capaz de descubrirlos. Es imposible realizar pruebas exhaustivas y garantizar la ausencia de defectos. Las pruebas sólo pueden demostrar la presencia de errores, pero no su ausencia.

Las pruebas deben estar presentes a lo largo de todo el ciclo de vida de desarrollo. La mayor parte de defectos se concentran en las fases tempranas del desarrollo y el costo de corrección aumenta a medida que el defecto continúa sin ser detectado. De esta forma, si las pruebas se realizan en todas las fases del ciclo de vida, se conseguirá un ahorro considerable a la hora de detectar y corregir errores en la misma fase en la que se produjeron.

Un aspecto fundamental del proceso de prueba es evaluar el cumplimiento de la especificación funcional del sistema o requisitos por parte del sistema construido. Para garantizar el nivel de calidad del sistema construido es necesario verificar la correcta y

completa implantación de los requisitos establecidos en las etapas iniciales del desarrollo. Una herramienta adecuada para efectuar esta validación son las pruebas del sistema.

Mejor práctica: *Es importante planificar y diseñar bien las pruebas para buscar el máximo número de errores posibles o, al menos, los más importantes. Para no depender en exclusiva de los conocimientos y experiencia de los ingenieros de prueba, es necesario un proceso metodológico que ofrezca una pauta clara para seleccionar el conjunto de pruebas que abarque un mayor rango de posibles errores. O bien obtener un conjunto de pruebas centrado en localizar errores con unas características concretas, cómo, por ejemplo, el nivel de rendimiento del sistema ante cargas extremas de trabajo.*

Mejor práctica: *Los costos de corregir errores en la fase de prueba son mucho mayores que los costos de corregir dichos errores en fases tempranas, como la fase de requisitos o de análisis. Adelantar la fase de pruebas a la fase de requisitos o de análisis permite reducir el costo de corrección de errores. Esto debe complementarse con estrategias que permitan, además, disminuir el número de errores potenciales, adelantando la generación de pruebas se evita dejar todo el proceso de prueba para final de la construcción, momento donde el tiempo y los recursos disponibles suelen ser insuficientes para realizar este proceso adecuadamente*

Tipos de pruebas necesarias: Existen diferentes tipos de pruebas que se pueden realizar, en la siguiente tabla se listan las más adecuadas. Es importante anotar que este tipo de pruebas es solo una guía que puede cambiar de acuerdo a las condiciones particulares de cada proyecto y por lo tanto la aplicabilidad de este modelo debe ser evaluada por el lector

Tipo de pruebas	Fase de realización	Descripción
Pruebas Unitarias	Durante la construcción del sistema	Prueban el diseño y el comportamiento de cada uno de los componentes del sistema una vez construidos
Pruebas de Integración	Durante la construcción del sistema	Comprueban la correcta unión de los componentes del sistema entre sí a través de sus interfaces, y si cumplen con la funcionalidad establecida
Pruebas de Sistema	Después de la construcción del sistema	Prueban a fondo el sistema, comprobando su funcionalidad e integridad globalmente, en un entorno lo más parecido posible al entorno final de producción
Pruebas de Implantación	Durante la implantación en el entorno de producción	Comprueba el correcto funcionamiento del sistema dentro del entorno real de producción
Pruebas de Aceptación	Después de la implantación en el entorno de producción	Verifican que el sistema cumple con todos los requisitos indicados y permite que los usuarios del sistema den el visto bueno definitivo
Pruebas de Regresión	Después de realizar modificaciones del sistema	El objetivo es comprobar que los cambios sobre un componente del sistema, no generan errores adicionales en otros componentes no modificados
Pruebas de Rendimiento	Durante la construcción, pruebas y liberación del producto	El objetivo es probar para determinar el rendimiento de un producto software
Pruebas de Seguridad	Durante la construcción, pruebas y liberación del producto	El objetivo es probar para determinar la seguridad de un producto software

Tabla 39. Descripción de los tipos de prueba del software. Fuente: Corporación Colombia Digital.

Mejor práctica: *Para que el proceso de prueba del sistema sea eficaz debe estar integrado dentro del propio proceso de desarrollo. Como cualquier otra fase de dicho proceso, el proceso de prueba debe realizarse de manera sistemática, minimizando el factor experiencia o intuición. Esto se puede conseguir a través de metodologías que guíen el proceso de desarrollo de pruebas de sistema.*

Para que la entidad no tenga problemas con los desarrollos implantados es necesario garantizar la calidad a través de pruebas propuestas en este documento. Dependiendo del procedimiento para realizar las pruebas, estas pueden ser:

Pruebas automáticas: son aquellas realizadas por un programa o herramienta que prueba el sistema sin necesidad de la interacción de una persona. La herramienta suministra una serie de valores de prueba, o acciones de prueba al sistema y verifica los resultados devueltos por el sistema con los resultados esperados. Al final del proceso de prueba se emite un informe con los resultados de las mismas.

Pruebas manuales: son aquellas pruebas realizadas por una o más personas que interactúan directamente con el sistema. Estas personas verifican si los resultados

obtenidos son válidos o no. Cuando se desea repetir el proceso es necesario que la persona o grupo repita las interacciones y vuelvan a verificar todos los resultados obtenidos.

Pruebas funcionales: Se denominan pruebas funcionales a las pruebas de *software* que tienen por objetivo probar que los sistemas desarrollados cumplen con las funciones específicas para los que han sido creados. Es común que este tipo de pruebas sean desarrolladas por analistas de pruebas con apoyo de algunos usuarios finales.

La función de un sistema es lo que hace dicho sistema, y normalmente es descrita en una especificación de requisitos, una especificación funcional o en casos de uso. Las pruebas funcionales son pruebas basadas en el análisis de la especificación funcional de un componente o de un sistema.

Las pruebas funcionales, a menudo, se llaman pruebas de caja negra porque no enfocan su atención en la manera de generar las respuestas del sistema. El enfoque de este tipo de pruebas se basa en el análisis de los datos de entrada y de salida, sin preocuparse del funcionamiento interno del sistema.

Las pruebas funcionales, en la mayoría de los casos, son realizadas manualmente por el analista de pruebas. También es posible automatizar este tipo de pruebas utilizando herramientas que permiten generar scripts conforme se hagan interacciones con el aplicativo a probar. La automatización de pruebas puede resultar compleja y sólo sería recomendable en algunos casos.

Al realizar pruebas funcionales lo que se pretende es ponerse en el lugar del usuario usando el sistema como él lo usaría, sin embargo, el analista de pruebas debe ir más allá que cualquier usuario y probar funciones que al usuario no se le ocurrirían. Generalmente, se requiere apoyo de los usuarios finales ya que ellos pueden aportar mucho en el desarrollo de casos de prueba complejos, enfocados básicamente al negocio y posibles

particularidades que no se hayan contemplado adecuadamente en el diseño funcional. Los analistas de pruebas han de tener en cuenta que los usuarios realizan las pruebas con la idea de que todo debería funcionar, a diferencia de ellos que tiene la función de encontrar defectos.

El proceso de pruebas para determinar la funcionalidad de un producto *software* se centra en la conformidad, interoperabilidad, seguridad y en la exactitud. Las pruebas de seguridad por ejemplo, se llevan a cabo para identificar y resolver vulnerabilidades de seguridad antes del despliegue o para identificar de forma periódica y resolver cuestiones de seguridad dentro del sistema.

Pruebas no funcionales (o pruebas de las características del producto de software): El objetivo es probar la calidad de las características de un *software*, o lo que es lo mismo, los atributos no funcionales del sistema, atributos no funcionales como: rendimiento, facilidad de uso, confiabilidad, portabilidad, mantenibilidad, entre otras. Las pruebas no funcionales, al igual que las funcionales, se pueden realizar a cualquier nivel de pruebas.

Las pruebas no funcionales incluyen pruebas de rendimiento, de usabilidad, mantenibilidad, fiabilidad y portabilidad. Entre los tipos de pruebas no funcionales que se han mencionado, unas de las más representativas son las pruebas de rendimiento que se encargan de determinar los límites operativos reales y simulan el uso real de la aplicación. Estas pruebas se pueden dividir en los siguientes tipos:

- Pruebas de carga: estas pruebas se encargan de determinar el comportamiento del sistema bajo diferentes cargas de trabajo.
- Pruebas de estrés: determinan el punto de ruptura donde el sistema revela el nivel de servicio máximo que puede conseguir.
- Pruebas de escalabilidad: evalúan los defectos de añadir *hardware y/o software* adicional para distribuir el trabajo entre los componentes del sistema.

Pruebas estructurales: Las pruebas estructurales son conocidas como pruebas de caja blanca o caja de cristal ya que se interesan en lo que pasa dentro del sistema. Son una aproximación al diseño de casos de prueba, donde las pruebas se derivan a partir del conocimiento de la estructura e implementación del *software*. Es decir, el diseño de los casos de prueba se enfoca en la estructura del componente o sistema. Estas pruebas tratan de analizar la estructura interna del componente/sistema y crear un modelo de pseudo-código a partir del código real.

Pruebas relacionadas con cambios

Pruebas de confirmación: Cuando una prueba falla, se averigua el defecto del *software* (la causa del fallo), se informa del defecto encontrado y se espera una nueva versión del *software* con el defecto arreglado. En este caso será necesario ejecutar de nuevo la prueba para confirmar que el defecto ha sido realmente solucionado. A esta prueba se llama prueba de confirmación.

Al realizar una prueba de confirmación es importante asegurarse que es ejecutada exactamente bajo las mismas condiciones y de la misma manera que la primera vez que se llevó a cabo, usando los mismos valores de entrada y el mismo entorno. Si todo sale bien se puede afirmar que el software es correcto para esas condiciones específicas, pero este arreglo puede haber introducido algún defecto distinto en otra parte del *software*. La manera de detectar estos efectos inesperados son las pruebas de regresión.

Pruebas de regresión: Al igual que las pruebas de confirmación, las pruebas de regresión incluyen casos de prueba que se ejecutaron con anterioridad. La diferencia es que, en las pruebas de regresión, los casos de prueba pasaron, probablemente, de forma satisfactoria la última vez que fueron ejecutadas.

El propósito de las pruebas de regresión es verificar que las modificaciones en el *software* o en su entorno no han causado ningún efecto adverso de forma no intencionada y que el sistema todavía cumple con sus requisitos.

Mejor práctica: *Incorporar el analista de pruebas en el equipo de desarrollo de software, quien es el responsable de todo el proceso de la planeación, ejecución, seguimiento y control del desarrollo de pruebas del sistema.*

Mejor práctica: *Las pruebas se deben realizar desde el inicio del ciclo de vida de desarrollo de sistemas de información, para minimizar el riesgo de errores y garantizar el nivel de calidad del sistema construido*

Mejor práctica: *Contratar un tercero especializado en el proceso de pruebas, en el caso que las entidades no cuenten con los recursos necesarios para hacerlo al interior.*

Mejor práctica: *Establecer la cantidad y tipo de pruebas exigidas y necesarias dentro del ambiente de producción.*

Pruebas de Seguridad: La seguridad de la información y por ende los sistemas que la contienen, administran y operan buscan protección contra una gama de amenazas en procura de minimizar los daños, ampliar las oportunidades de negocio, retorno de la inversión y asegurar la continuidad del negocio.

Las pruebas de seguridad de los sistemas de información son un mecanismo de control preventivo para mitigar riesgos operacionales y gestionar su tratamiento y plan de respuesta (Riesgo operacional, el Comité Basilea II lo define como: “el riesgo de pérdidas debido a la inadecuación o a fallas en los procesos, personal y sistemas internos o por causa de eventos externos”).

El proyecto abierto de seguridad de aplicaciones Web (OWASP, Open Web Application Security Project), realizó un estudio sobre los 10 riesgos más críticos en Aplicaciones web, con el objetivo de generar conciencia acerca de la seguridad en sistemas de información.

A continuación se presenta el listado de riesgos más críticos OWASP TOP 10, del año 2013, en sistemas de información, priorizados con base al impacto que genera para el negocio y la factibilidad de ser vulnerados.

Riesgo
A1 – Inyección Las fallas de inyección, tales como SQL, OS, y LDAP, ocurren cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al intérprete en ejecutar comandos no intencionados o acceder datos no autorizados.
A2 – Pérdida de autenticación y gestión de sesiones Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son frecuentemente implementadas incorrectamente, permitiendo a los atacantes comprometer contraseñas, claves, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios.
A3 – Secuencia de comandos en sitios cruzados (XSS) Las fallas XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada. XSS permite a los atacantes ejecutar secuencia de comandos en el navegador de la víctima los cuales pueden secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso.

A4 – Referencia directa insegura a objetos

Una referencia directa a objetos ocurre cuando un desarrollador expone una referencia a un objeto de implementación interno, tal como un archivo, carpeta, o base de datos. Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder datos no autorizados.

A5 – Configuración de seguridad incorrecta

Una buena seguridad requiere tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos y plataforma. Todas estas configuraciones deben ser definidas, implementadas y mantenidas ya que por lo general no son seguras por defecto. Esto incluye mantener todo el software actualizado, incluidas las librerías de código utilizadas por la aplicación.

A6 – Explosión de datos sensibles

Muchas aplicaciones web no protegen adecuadamente datos sensibles tales como números de tarjetas de crédito o credenciales de autenticación. Los atacantes pueden robar o modificar tales datos para llevar a cabo fraudes, robos de identidad u otros delitos. Los datos sensibles requieren métodos de protección adicionales tales como el cifrado de datos, así como también de precauciones especiales en un intercambio de datos con el navegador.

A7 – Ausencia de control de acceso a las funciones

La mayoría de aplicaciones web verifican los derechos de acceso a nivel de función antes de hacer visible en la misma interfaz de usuario. A pesar de esto, las aplicaciones necesitan verificar el control de acceso en el servidor cuando se accede a cada función. Si las solicitudes de acceso no se verifican, los atacantes podrán realizar peticiones sin la autorización apropiada.

A8 – Falsificación de peticiones en sitios cruzados (CSRF)

Un ataque CSRF obliga al navegador de una víctima autenticada a enviar una petición http falsa, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Esto permite al atacante forzar al navegador de la víctima a generar peticiones que la aplicación asume como legítimas y que son propias de la víctima.

A9 – Uso de componentes con vulnerabilidades conocidas

Algunos componentes tales como, las librerías, los frameworks, y otros módulos de software casi siempre funcionan con todos los privilegios. Si se ataca un componente vulnerable esto podría facilitar la intrusión en

el servidor, o una pérdida de dados. Las aplicaciones que utilicen componentes con vulnerabilidades conocidas debilitan las defensas del sistema de información y amplían el rango de posibles ataques e impactos.
A10 – Redirecciones y reenvíos no válidos Las aplicaciones web frecuentemente redirigen y reenvían a los usuarios hacia otras páginas o sitios web, y utilizan datos no confiables para determinar la página de destino. Sin una validación apropiada, los atacantes pueden redirigir a las víctimas hacia sitios de phishing o malware, o utilizar reenvíos para acceder a páginas no autorizadas.

Tabla 40. OWASP Top 10 – 2013 de Riesgos de Seguridad en Aplicaciones. Licencia CC (Creative Commons). Fuente: https://www.owasp.org.

El objetivo de las pruebas de seguridad es determinar el nivel de propensión o aversión al riesgo y la manera como desde la codificación se han diseñado e implementado controles eficientes a las vulnerabilidades más frecuentes.

En el marco de las pruebas, las de intrusión constituyen un método de evaluación de seguridad de sistemas de información o de equipos en una red mediante la simulación de un ataque.

El proceso conlleva un análisis activo del sistema, en busca de cualquier debilidad, fallos técnicos o vulnerabilidades.

¿Qué es una vulnerabilidad?

Dado que una aplicación posee un conjunto de activos (recursos de valor como los datos en una base de datos o en el sistema de archivos), una vulnerabilidad es una debilidad en un activo que hace posible a una amenaza. Así que una amenaza es un caso potencial que puede dañar un activo mediante la explotación de una vulnerabilidad. Un test es una acción que tiende a mostrar una vulnerabilidad en la aplicación.

Las pruebas de seguridad se pueden dividir en 2 fases:

Modo Pasivo: En éste modo, la persona a cargo de realización de las pruebas intenta comprender la lógica del sistema, “juega con la aplicación”; puede usarse una utilidad para la recopilación de la información, como un Proxy http, para observas todas las peticiones y respuestas http. Al final de esta fase esta persona debería comprender cuales son todos los puntos de acceso (puertas) de la aplicación (p. e. Cabeceras http, parámetros, cookies).

Modo Activo: OWASP ha generado un listado de 9 subcategorías de pruebas las cuales a su vez exponen escenarios más específicos de pruebas de seguridad a sistemas de información.

Categoría	Nombre de la Prueba
Pruebas de Gestión de la Configuración	Pruebas SSL/TLS (SSL Versión, Algoritmos, longitud de Claves, Validez de Certificado Digital). Prueba de DB Listener. Prueba de Gestión de Configuración de Infraestructura. Prueba de Gestión de Configuración de Aplicación. Prueba del Gestor de Extensión de Ficheros. Antiguo, backup y ficheros no referenciados. Interface de Administración de Aplicación e Infraestructura. Prueba de métodos HTTP y XST.

Pruebas de Autenticación	Transporte de Credenciales sobre canal cifrado. Prueba para Enumeración de usuarios. Prueba de detección de Cuentas de Usuario Adivinables (Diccionario). Prueba de Fuerza Bruta. Prueba para evitar esquemas de autenticación. Prueba de recordatorio de contraseña y restablecimiento. Prueba de Cierre de Sesión y Gestión de Cache de Navegación.
Gestión de Sesiones	Prueba de atributos de Cookies. Prueba de Fijación de Sesión. Prueba de Variables de Sesión Expuestas. Prueba de CSRF.
Pruebas de Autorización	Prueba de Ruta Transversal. Prueba para Evitar Esquema de Autorización. Prueba de escalada de Privilegios.
Pruebas de Lógica de Negocio	Prueba de lógica de negocio
Pruebas de validación de datos	Prueba de XSS Reflejado Prueba de XSS Almacenado Prueba de XSS basado en DOM Prueba de XSS basado en Flash Inyección SQL Inyección LDAP Inyección ORM Inyección XML Inyección SSI

	Inyección XPath Inyección IMAP/SMTP Inyección de Código Inyección de Ordenes del Sistema Operativo Desbordamiento de buffer Prueba de Vulnerabilidad incubada Prueba de HTTP (Splitting/Smuggling)
Pruebas de denegación de servicio	Prueba de Ataques a través de Comodines SQL Bloqueo de Cuentas de Usuarios Pruebas de DoS mediante Desbordamiento de Buffer Asignación de Objeto de Usuario Especificado Entrada de usuario como un contador de bucle Prueba de Escritura en Disco de data provista por Usuario Fallo en Liberar Recursos Almacenamiento de demasiados datos en Sesión
Pruebas de Servicios Web	Recopilación de Información de WS Prueba de WSDL Prueba en la Estructura del XML Prueba del XML a nivel de contenido Prueba de REST/parámetros HTTP GET Adjuntos SOAP maliciosos Prueba de Repetición
Pruebas Ajax	Vulnerabilidades Ajax Pruebas Ajax

Tabla 41. Pruebas de seguridad a sistemas de información. Fuente: https://www.owasp.org.

Las pruebas de seguridad aplicadas deben considerarse en ambientes de pruebas que simulen el comportamiento transaccional y operacional real, de tal forma que la aplicación de las mismas arrojen evidencias e indicadores del nivel de seguridad o vulnerabilidad del sistema de información evaluado.

Mejor práctica: *Documentar de forma exhaustiva los resultados de los diferentes tipos de pruebas realizadas, para que se tomen como base de conocimiento.*

Mejor práctica: *Es necesario garantizar que las pruebas se ejecuten a través del ciclo de vida del desarrollo de software y no dejarlas para el final.*

Las mejores prácticas mencionadas se pueden desarrollar por medio de Agile Testing que es una práctica de pruebas que sigue los principios del desarrollo ágil de *software*, por lo cual se recomienda utilizarlo. A continuación se describe en qué consisten este tipo de pruebas y sus ventajas

Agile Testing

Involucra a todos los miembros de un equipo ágil multifuncional, en el cual el rol del probador es el de un experto multifuncional, garante de que se entregue el valor de negocio deseado por la entidad a un ritmo sostenible y continuo.

Las metodologías ágiles no ven al *software* de prueba como una fase separada, sino como parte integral del desarrollo de *software*.

Agile Testing, incorpora una serie prácticas, como por ejemplo pruebas de “todo el equipo”, prueba independiente (opcional), Integración continua, pruebas guiadas por el desarrollo (Test Driven Development – TDD), Desarrollo guiado por comportamiento (Behaviour Driven Development – BDD), Desarrollo guiado por pruebas de aceptación (Acceptance Test Driven Development – ATDD), entre otros.

Los equipos ágiles utilizan un enfoque de “todo el equipo” enfocado a las pruebas, con la finalidad de integrar la calidad al desarrollo del producto, al contrario de un enfoque de primero fabricar el producto y luego inspeccionar para determinar su nivel de calidad.

Principios del Agile Testing

El Agile Testing engloba los siguientes principios:

Las pruebas no son una fase: Pruebas continuas son la única forma de garantizar avance continuo, por esto, las pruebas se realizan simultáneamente junto con el desarrollo de *software* y demás actividades.

Las pruebas hacen avanzar el proyecto: Proporciona retroalimentación continua, permitiendo corregir el rumbo continuamente durante el desarrollo de *software*.

Todo el equipo realiza pruebas: en Agile Testing, los analistas de negocio y desarrolladores de *software* también ejecutan pruebas, no sólo los probadores como en métodos convencionales.

Reducir el tiempo para recibir retroalimentación: En Agile Testing, los equipos del área de negocio (el cliente) están involucrados en cada iteración, no solo al final durante la fase de aceptación, como resultado, el tiempo de retroalimentación se reduce y el costo de correcciones también es menor.

Código limpio: Los defectos en el código se corrigen en la misma iteración, por lo que se mantiene el código limpio.

Reducir la documentación de pruebas: Los Agile Testers usan listas de chequeo reusables en lugar de documentación extensa, se enfocan en la esencia de la prueba en lugar de detalles. Siguiendo principios ágiles estas listas de chequeo son el inicio de las definiciones de las pruebas y no el final el probador cuenta con libertad para aportar valor.

Guiado por pruebas: En Agile Testing, las pruebas se hacen durante el desarrollo y no después del desarrollo como en métodos convencionales.

Elemento transversal – Riesgos: *Planeación poco adecuada. Cuando no se llevan a cabo las pruebas necesarias que garanticen la calidad del software conlleva a pérdidas de tiempo y sobrecostos al proyecto*

7.3 GESTION DE LA CALIDAD

Los principales objetivos son:

- Lograr que los productos se entreguen con niveles altos de calidad.
- Que la funcionalidad de los productos cumpla con las expectativas de la entidad.
- Verificar que los requerimientos tengan un nivel de cumplimiento con los objetivos definidos por la entidad.
- Validar la funcionalidad de los módulos, sistemas e interfaces definidas dentro del alcance del proyecto

7.3.1 Normas

La Norma ISO/IEC 25040 define el proceso para llevar a cabo la evaluación del producto de *software* y que sirve como guía para que las entidades las adopten. Dicho proceso de evaluación consta de un total de cinco actividades.

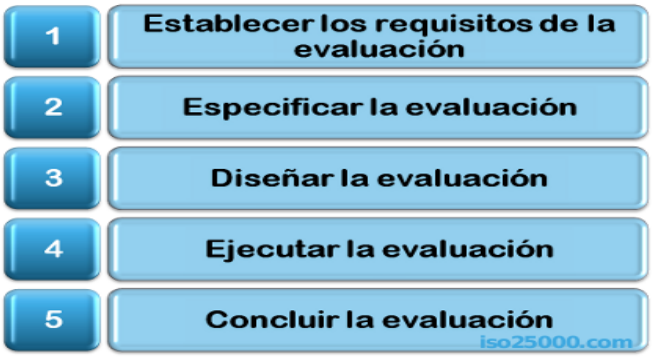


Figura 68. Actividades evaluación del producto de software. Fuente: http://iso25000.com/.

Actividad 1: Establecer los requisitos de la evaluación

El primer paso del proceso de evaluación consiste en establecer los requisitos de la evaluación.

Tarea 1.1: Establecer el propósito de la evaluación

En esta tarea se documenta el propósito por el que la entidad quiere evaluar la calidad de su producto de *software* (asegurar la calidad del producto, decidir si se acepta un producto, determinar la viabilidad del proyecto en desarrollo, comparar la calidad del producto con productos de la competencia, etc.).

Tarea 1.2: Obtener los requisitos de calidad del producto

En esta tarea se identifican las partes interesadas en el producto de *software* (desarrolladores, posibles adquirientes, usuarios, proveedores, etc.) y se especifican los requisitos de calidad del producto utilizando un determinado modelo de calidad.

Tarea 1.3: Identificar las partes del producto que se deben evaluar

Se deben identificar y documentar las partes del producto de *software* incluidas en la evaluación. El tipo de producto a evaluar (especificación de requisitos, diagramas de diseño, documentación de las pruebas, etc.) depende de la fase en el ciclo de vida en que se realiza la evaluación y del propósito de ésta.

Tarea 1.4: Definir el rigor de la evaluación

Se debe definir el rigor de la evaluación en función del propósito y el uso previsto del producto *software*, basándose, por ejemplo, en aspectos como el riesgo para la seguridad, el riesgo económico o el riesgo ambiental. En función del rigor se podrá establecer qué técnicas se aplican y qué resultados se esperan de la evaluación.

Actividad 2: Especificar la evaluación

En esta actividad se especifican los módulos de evaluación (compuestos por las métricas, herramientas y técnicas de medición) y los criterios de decisión que se aplicarán en la evaluación.

Tarea 2.1: Seleccionar los módulos de evaluación

En esta tarea el evaluador selecciona las métricas de calidad, técnicas y herramientas (módulos de evaluación) que cubran todos los requisitos de la evaluación. Dichas métricas deben permitir que, en función de su valor, se puedan realizar comparaciones fiables con criterios que permitan tomar decisiones. Para ello se puede tener en cuenta la Norma ISO/IEC 25020.

Tarea 2.2: Definir los criterios de decisión para las métricas

Se deben definir los criterios de decisión para las métricas seleccionadas. Dichos criterios son umbrales numéricos que se pueden relacionar con los requisitos de calidad y posteriormente con los criterios de evaluación para decidir la calidad del producto. Estos umbrales se pueden establecer a partir de benchmarks, límites de control estadísticos, datos históricos, requisitos del cliente, etc.

Tarea 2.3: Definir los criterios de decisión de la evaluación

Se deben definir criterios para las diferentes características evaluadas a partir de las subcaracterísticas y métricas de calidad. Estos resultados a mayor nivel de abstracción permiten realizar la valoración de la calidad del producto *software* de forma general.

Actividad 3: Diseñar la evaluación

En esta actividad se define el plan con las actividades de evaluación que se deben realizar.

Tarea 3.1: Planificar las actividades de la evaluación

Se deben planificar las actividades de la evaluación teniendo en cuenta la disponibilidad de los recursos, tanto humanos como materiales, que puedan ser necesarios. En la planificación se debe tener en cuenta el presupuesto, los métodos de evaluación y estándares adaptados, las herramientas de evaluación, etc.

El plan de evaluación se revisará y actualizará proporcionando información adicional según sea necesario durante el proceso de evaluación.

Actividad 4: Ejecutar la evaluación

En esta actividad se ejecutan las actividades de evaluación obteniendo las métricas de calidad y aplicando los criterios de evaluación.

Tarea 4.1: Realizar las mediciones

Se deben realizar las mediciones sobre el producto de *software* y sus componentes para obtener los valores de las métricas seleccionadas e indicadas en el plan de evaluación. Todos los resultados obtenidos deberán ser debidamente registrados.

Tarea 4.2: Aplicar los criterios de decisión para las métricas

Se aplican los criterios de decisión para las métricas seleccionadas sobre los valores obtenidos en la medición del producto.

Tarea 4.3: Aplicar los criterios de decisión de la evaluación

En esta última tarea se deben aplicar los criterios de decisión a nivel de características y subcaracterísticas de calidad, produciendo como resultado la valoración del grado en que el producto de *software* cumple los requisitos de calidad establecidos.

Actividad 5: Concluir la evaluación

En esta actividad se concluye la evaluación de la calidad del producto de *software*, realizando el informe de resultados que se entregará a la entidad y revisando con ésta los resultados obtenidos.

Tarea 5.1: Revisar los resultados de la evaluación

Mediante esta tarea, el evaluador y el cliente de la evaluación (en caso de existir) realizan una revisión conjunta de los resultados obtenidos, con el objetivo de realizar una mejor interpretación de la evaluación y una mejor detección de errores.

Tarea 5.2: Crear el informe de evaluación

Una vez revisados los resultados, se elabora el informe de evaluación, con los requisitos de la evaluación, los resultados, las limitaciones y restricciones, el personal evaluador, etc.

Tarea 5.3: Revisar la calidad de la evaluación y obtener retroalimentación de la entidad.

El evaluador revisará los resultados de la evaluación y la validez del proceso de evaluación, de los indicadores y de las métricas aplicadas. La retroalimentación de la revisión debe servir para mejorar el proceso de evaluación de la organización y las técnicas de evaluación utilizadas.

Tarea 5.4: Tratar los datos de la evaluación

Una vez finalizada la evaluación, el evaluador debe realizar el adecuado tratamiento de los datos y los objetos de la evaluación según lo acordado con el cliente (en caso de ser un tercero), devolviéndolos, archivándolos o eliminándolos según corresponda.

Mejor práctica: *Para garantizar el nivel de calidad del sistema construido es necesario verificar la correcta y completa implantación de los requisitos establecidos en las etapas iniciales del desarrollo.*

7.4 HERRAMIENTAS

El uso de herramientas para probar *software* es indispensable durante el ciclo de vida de desarrollo, ya que ahorran tiempo y costo para las entidades. En el mercado existen diversas herramientas para este fin, a continuación se listan algunas de ellas. Aunque se debe precisar que las herramientas que automatizan las pruebas no hacen más seguro o confiable al sistema de información, ayudan a optimizar el proceso y a la oportunidad en la detección y corrección de fallos, por tal motivo, su importancia radica en la intención de la Entidad y los responsables del diseño, construcción y puesta en producción del sistema de información de asegurar la calidad del producto de *software*.

Selenium

Es un framework para pruebas de aplicaciones Web, descargable de forma gratuita desde su sitio web. Proporciona una herramienta de grabación y playback, que permite desarrollar pruebas sin necesidad de aprender un lenguaje de codificación. Incluye características como grabación, playback, selección de campos, auto completar formularios, pruebas de recorrido (Walkthrough), debug, puntos de control, scripts basadas en el lenguaje de programación ruby y otros formatos.

Selenium cobra importancia en la etapa de pruebas, para construir *scripts* para ejecución de pruebas funcionales automatizadas.

HP Quicktest Professional (QTP)

Proporciona la capacidad de automatizar pruebas funcionales y pruebas de regresión para *software* y ambientes de prueba.

Proporciona la capacidad de definir Scripts de prueba y posee una interfaz gráfica que le permiten al usuario emular la funcionalidad que desea probar, incluyendo el uso de interfaces de usuario de las aplicaciones a probar. Incluye características como: Vista de experto, pruebas de procesos de negocio, grabado de pantalla (para captura de las evidencias de prueba), entre otras posibilidades.

Visual Studio Test Proffessional

Conjunto de herramientas de pruebas integradas desarrolladas por Microsoft, que proporcionan soporte a todo el ciclo de planificación, ejecución y registro de pruebas, con facilidades de colaboración entre analistas de prueba (probadores) y desarrolladores en la herramienta.

Proporciona capacidad de realizar pruebas manuales, reutilización de pruebas manuales, integración con el "team foundation server"[1], gestión de ciclo de vida de aplicaciones, entre otros.

Rational Functional Tester

Herramienta de automatización de pruebas funcionales y de regresión. Proporciona capacidades de pruebas de interfaz gráfica, pruebas manejadas por datos (Data Driven), pruebas funcionales y pruebas de regresión.

Algunas de sus características son: simplificación de creación y visualización de pruebas, trazabilidad en todo el ciclo de vida, validación de data dinámica (por medio de un wizard), e inclusive capacidad de definir scripts (por medio de lenguajes de Scripting).

[1] Team Foundation Server es la herramienta definitiva para la gestión completa de todos los aspectos de una aplicación de cualquier tamaño

Apache Jmeter

Es una aplicación open source y 100% Java, diseñada para realizar pruebas de carga del comportamiento funcional y medir el rendimiento de un sistema de información. Originalmente fue creada para practicar pruebas a aplicaciones web pero ha sido extendida su funcionalidad a otros tipos de aplicaciones.

Jmeter tiene la posibilidad de utilizar diferentes tipos de protocolos y servidores, por ejemplo: Web – http, https, Soap / Rest, Ftp, Base de datos: JDBC, Directorio Activo: LDAP, MOM vía JMS, Mail: smtp, pop3, imap, TCP, Lenguajes: Java, PHP, ASP.NET

SoapUI

SoapUI es una herramienta para diseño y ejecución de pruebas funcionales y especialmente de SOA y Web Services basados en REST (Representational State Transfer, Transferencia de Estado Representacional – Estilo de Arquitectura de *Software* para sistemas hipermedia que requiera describir la interfaz entre sistemas, por ejemplo web services SOAP), aunque también permite ejecutar pruebas de regresión y de carga automatizadas.

SoapUI se constituye en la herramienta predilecta de los analistas de pruebas para verificar funcionalidad de los web services publicados o expuestos para ser consumidos por un tercero.

Herramientas para seguimiento y gestión de métricas de pruebas:

TestLink

Es un sistema web, de código abierto, cuya función es permitir la gestión de pruebas de *software*. Presenta una estructura de información dividida en 3 unidades:

Proyecto de Pruebas (Test Project)

Plan de Pruebas (Test Plan)
Usuario (User)

Sobre ésta herramienta se definen los escenarios de pruebas y sus resultados, para generar informes de calidad y métricas.

RedMine

No es una herramienta de gestión de pruebas perse, sin embargo, es utilizada comúnmente para planear y hacer seguimiento a la ejecución y resultados de los ciclos de pruebas.

Una característica especial de RedMine es que permite definir y parametrizar el flujo de trabajo para la definición, ejecución y medición de resultados de pruebas, determinando tiempo invertido y métricas de calidad.

Zephyr / Jira

Se trata de un *software* de gestión del ciclo de pruebas de sistemas de información. Dispone de una versión gratuita con un límite máximo de 10 usuarios. Al igual que los anteriores, cuenta con características que posibilitan la creación de proyectos y planes de prueba y generación de informes de calidad.

Mantis

Es quizás, la herramienta de seguimiento de problemas e incidentes de sistemas de información más común dada su naturaleza Open Source. Permite crear varios proyectos simultáneamente, asociarlos y registrar sus correspondientes issues (incidencias), gestionar los niveles y privilegios de acceso, adjuntar documentación de la ejecución de pruebas, hacer seguimiento al inventario de “bugs” (defectos) y generar reportes por Proyecto, estado del incidente, desarrollador, nivel de complejidad, etc.

***Elemento transversal – Talento TI**: El personal responsable de las actividades de pruebas son los siguientes:*

Analista de pruebas

Es el responsable de todo el proceso de la planeación, ejecución, seguimiento y control del desarrollo de pruebas del sistema, enfocándose principalmente en:

- *Identificar y definir las pruebas necesarias.*
- *Facilitar los recursos necesarios para el desarrollo de las pruebas.*
- *Seguimiento detallado de las pruebas.*
- *Revisar los informes de pruebas entregados por el equipo de pruebas del proyecto*
- *Recopilación y gestión de los datos de prueba en cada ciclo de pruebas.*
- *Evaluación de la calidad global como resultado de las actividades de prueba.*

8 PUESTA EN PRODUCCION

8.1 PROBLEMAS FRECUENTES.

- Falta de planeación para el despliegue o puesta en producción del *software*, lo que no permite realizar un seguimiento paso a paso del proceso.
- Se observan casos en los que los problemas de compatibilidad o de idoneidad de las plataformas tecnológicas se detectan hasta la fase de puesta en producción.
- Se observan casos en los que los problemas de conectividad y seguridad se detectan hasta la fase de puesta en producción.
- Las entidades no cuentan con personal experto en las pruebas de aceptación.
- Algunas veces es necesario realizar cambios en el *software* implantado, por problemas de usabilidad o interoperabilidad, entre otros. Esto genera inconvenientes para la entidad, ya que no existe un procedimiento de gestión de cambios establecido.
- Los entregables que genera el proceso tales como documentación, código fuente o evidencia de pruebas de funcionabilidad, no son almacenados en un repositorio que permita su posterior consulta.
- Los entregables que genera el proceso tales como documentación, código fuente o evidencia de pruebas de funcionabilidad, no tienen la información relevante para su posterior consulta.

8.2 ASPECTOS GENERALES.

La gestión de la puesta en producción permite implementar un proceso integrado para lograr una entrega efectiva de un producto o servicio de *software* que satisfaga los requerimientos, tanto del propio negocio de una entidad como del proveedor.

Esta gestión tiene como objetivos entregar, distribuir y hacer un seguimiento de los cambios que se presenten en la puesta en producción. Es conveniente que este proceso esté integrado con la gestión de la configuración y la gestión de cambios.

Antes de la puesta en producción hay que tener en cuenta una serie de acciones y criterios por seguir:

- Hay que establecer políticas de congelación de código
- Todos los requisitos han de estar cerrados
- Se deben realizar evaluaciones de métricas de riesgos
- El nivel de calidad (basado en el último ciclo de pruebas) debe cumplir con los criterios acordados. Hay que asegurar la calidad de las aplicaciones del *software* antes de pasarlas a producción, ya que ese es el momento en el que es más costoso encontrar un defecto.
- Deben existir estrategias para mitigar los riesgos posteriores a la puesta en producción. Para esto, se debe contar con un plan que ayude a abordar los riesgos que puedan surgir en la producción como: mejoras, solicitudes de cambios e incidentes. Es necesaria una integración entre la gestión del cambio, los requisitos que genera ésta y el aseguramiento de la calidad.

8.2.1 Actividades de la puesta en producción

Planificación

En la planificación se establece un marco general en el que se fijan las fechas para las actividades que se realizarán, debe ser consensuada y aprobada por la entidad y el proveedor.

En cualquier caso, la planificación debe contemplar lo siguiente:

- Alcance, contenido, riesgos y responsabilidades
- Recursos necesarios: *software*, *hardware* y recursos humanos

- Equipo de trabajo requerido
- Método de colaboración con todas las partes interesadas
- Cronograma detallado
- Soporte

Elemento transversal – riesgos:

- *Falta de conocimientos técnicos para el manejo y configuración del software y hardware.*
- *Manejo inadecuado de los ambientes de pruebas y producción.*
- *No contar con una línea base respecto al proceso de puesta en producción.*
- *No configurar adecuadamente los ambientes de desarrollo, pruebas y producción.*
- *Poca disponibilidad de los interesados en el proceso de puesta en producción.*

Pruebas o *Testing*

Una vez el *software* es implementado, se envía a control de calidad para las pruebas estándares de aceptación, se revisa para verificar que cumple con los requerimientos y que funciona correctamente. Durante esta fase, se documenta el proceso completo para tener en el futuro una base de conocimientos. Después de la verificación final se deben actualizar los estándares de prueba para adaptarlos al nuevo *software*.

Mejor práctica: *Para garantizar el nivel de calidad del sistema que se desplegará, se deben realizar las pruebas necesarias y documentarlas de forma adecuada.*

Preparación de la entrega del *software*

Cuando se tiene una entrega correcta y probada, se deben tener como mínimo los siguientes documentos:

- Lista de fallas que han sido corregidas
- Nombre de la entrega (versión que se ha desarrollado)
- Especificación del entorno para el cual se ha construido la entrega

- Archivos de configuración
- Informes de las pruebas realizadas

Entrega o puesta en producción

Este procedimiento hace referencia a la instalación del sistema de información para que pueda ser utilizado por el usuario o solicitante de la entidad. Tenga en cuenta lo siguiente:

- Planificar y supervisar el paso a producción del *software* y *hardware*.
- Controlar procedimientos eficientes para la distribución e instalación de los cambios en los sistemas de TI.
- Asegurar que las modificaciones sobre el *software* y *hardware* se registran, son seguras y sólo se instalan versiones correctas, autorizadas y probadas.
- Comunicarse con los responsables de los proyectos y gestionar sus expectativas durante la planificación y desarrollo de los nuevos pasos a producción.
- Implementar nuevas versiones de *hardware* y *software* mediante la gestión de la configuración y de los cambios.

8.2.2 Procesos asociados a la puesta en producción

Gestión de cambios

El principal objetivo de esta gestión es la evaluación y planificación del proceso de cambio para asegurar que se haga de la forma más eficiente, siguiendo los procedimientos establecidos y asegurando en todo momento la calidad y continuidad del servicio de TI:

Las principales razones para la realización de cambios son:

- Solución de errores conocidos
- Desarrollo de nuevos servicios
- Mejora de los servicios existentes
- Imperativo legal

Es responsabilidad del analista de cambios velar porque la solicitud de cambio contenga la información mínima requerida, con coherencia, así como que la documentación anexa sirva para su debido análisis, clasificación, aprobación/rechazo y seguimiento, y facilite el cierre final de la solicitud.

Cuando la entidad requiera realizar cambios en el *software* implantado, se sugiere establecer un procedimiento de control con las siguientes actividades:

- Registrar la solicitud
- Revisar y clasificar el cambio, la documentación de soporte de la solicitud y las aprobaciones
- Evaluar y planear el cambio
- Aprobar/rechazar el cambio
- Implementar el cambio
- Verificar y cerrar el cambio
- Tener una arquitectura tecnológica adecuada para la puesta en producción de los sistemas de información
- Realizar pruebas de carga y estrés que garanticen la capacidad del sistema
- Contar con personal especializado que apruebe los productos entregados
- Realizar despliegues de forma planeada y concertada con el personal de la operación

Gestión de la configuración

Se denomina gestión de la configuración al conjunto de procesos destinados a asegurar la validez de todo producto obtenido durante cualquiera de las etapas del desarrollo de un sistema de información, a través del estricto control de los cambios. Estos dos elementos, control de cambios y control de versiones, facilitan también el mantenimiento de los sistemas al proporcionar una imagen detallada del sistema en cada etapa del desarrollo. La gestión

de la configuración se realiza durante todas las fases del desarrollo tras la puesta en producción, incluyendo el mantenimiento y control de cambios.

La gestión de la configuración se realiza desde que comienza el proyecto hasta que termina. Involucra la recolección y el mantenimiento de toda la información sobre *hardware* y *software*. Forma parte de un proceso más general de gestión de la calidad del *software*, de hecho, la misma persona o grupo responsable de la calidad puede encargarse también de este apartado. La finalidad de todo esto es tener controlados los cambios y tener la información necesaria en el momento del mantenimiento.

Cuando la entidad requiera realizar gestión de la configuración, se sugiere establecer un procedimiento con las siguientes actividades:

- Planificación: en esta fase se determinan los objetivos y estrategias de la gestión de la configuración y activos TI.
- Clasificación y registro: los elementos de la configuración deben ser registrados de acuerdo con el alcance, nivel de profundidad y nomenclatura predefinidas.
- Monitoreo y Control: se hace seguimiento a la base de datos de la gestión de la configuración (CMDB) para asegurar que todos los componentes autorizados estén correctamente registrados y que se conoce su estado actual.
- Realización de auditorías: se asegura que la información registrada en la CMDB coincide con la configuración real de la estructura TI de la organización.
- Elaboración de informes: se evalúa el rendimiento de la gestión de la configuración y los activos de TI, para aportar información de vital importancia a otras áreas de la infraestructura de TI.

8.3 GESTION DE LA CALIDAD

Las pruebas de disponibilidad operativa son la última fase de las pruebas realizadas en el sistema de producción. Se programan y conducen como parte del plan de puesta en producción. Se ejecutan después de que se hayan completado el resto de actividades y antes de decidir si se lleva a producción o no.

Las pruebas de disponibilidad operativa aseguran que el sistema que vamos a entregar tiene cargados los datos de producción apropiados, que está listo para comunicarse con los sistemas externos requeridos y para ejecutar los procesos por lotes, mientras cumple con cualquier requisito de seguridad y de conformidad.

Esta etapa de pruebas es la última comprobación operativa del sistema de producción, para verificar su disponibilidad antes del punto en el que el sistema se ponga en marcha. Esta etapa debería formar parte del plan de implementación del proyecto.

Los elementos clave de las pruebas de disponibilidad operativa son los siguientes:

- Establecimiento de usuarios finales
- Definición de funciones y requisitos
- Asignaciones de seguridad y contraseñas
- Verificación de conectividad completa
- Verificación de datos de producción
- Verificación de funcionalidad del núcleo del negocio
- Verificación de la herramienta de monitorización de la aplicación
- Soporte para el centro de servicios
- Documentación de apoyo

Los requisitos de calidad para la disponibilidad operativa se definen normalmente por el equipo del proyecto o, si la tarea operativa se externaliza, por una combinación con los proveedores y el grupo interno.

Mejor práctica: *Para garantizar el nivel de calidad del sistema que se desplegará, las entidades deben realizar las actividades de forma planeada y concertada con el personal de producción.*

Elemento transversal – documentación: *La documentación o entregables que debe producir la gestión de la puesta en producción se trata de:*

- *Acta de despliegue: deja constancia de que la aplicación desplegada en el ambiente de desarrollo se encuentra funcionando de manera adecuada.*
- *Acta de conformidad de despliegue en producción: documento en el que se consta el correcto funcionamiento del software en el ambiente de producción.*
- *Manual de configuración: documento en el que se plasma la configuración que debe tener el software luego de ser desplegado en los ambientes de producción.*
- *Manual de instalación: registra los pasos para realizar el despliegue del software desarrollado.*

Mejor práctica: *Para garantizar el nivel de calidad del sistema que se desplegará, las entidades deben tener personal experto en los procesos de pruebas de aceptación y de puesta en producción.*

Elemento transversal – Talento TI: *El personal responsable de las actividades de esta fase debe desempeñar los siguientes roles:*

Gestor de la Configuración

Es el responsable de la gestión de la configuración. Entre sus principales funciones están:

- *Ejecutar el proceso de despliegue de las aplicaciones.*

- *Enviar la solicitud de aprobación del proyecto para que pase a producción.*
- *Llenar el acta de aceptación luego de la puesta en producción.*
- *Enviar las solicitudes de servicio a lo largo del proceso de puesta en producción.*

Gestor de Cambios

Es el responsable de la gestión de los cambios. Entre sus principales funciones están:

- *Planear y presidir las reuniones del Comité de cambios (CAB) y el Comité de cambios de emergencia (ECAB).*
- *Autorizar o rechazar el cambio.*
- *Hacer seguimiento a cada cambio registrado.*
- *Resolver los conflictos sobre los cuales no haya acuerdo en el comité.*
- *Revisar y confirmar el resultado de las solicitudes de cambio implementadas.*
- *Revisar cambios pendientes o en trámite.*
- *Participar en la negociación de un nuevo alcance para los cambios no satisfactorios.*

Analista de cambios

Es el encargado analizar las solicitudes de cambio, así como de asegurar la debida documentación que soporta cada una de ellas, para su respectivo seguimiento hasta el cierre. Entre sus principales funciones están:

- *Realizar seguimiento a cada cambio registrado.*
- *Validar que las especificaciones técnicas y funcionales correspondan al alcance del cambio, además de revisar sus respectivas aprobaciones funcionales o técnicas.*
- *Asistir a las reuniones del Comité de cambios (CAB) y del Comité de cambios de emergencia (ECAB).*
- *Realizar el seguimiento al cambio desde el registro de la solicitud, la verificación de la completitud y la coherencia de la misma.*
- *Evaluar el impacto de la solicitud de cambio, validar la priorización y categorización.*

- *Revisar la documentación asociada y registrada en el repositorio o en la herramienta definida por el proceso de cambios.*
- *Hacer seguimiento al cierre oportuno de las solicitudes de cambios.*
- *Generar los informes e indicadores del proceso de control de cambios con sus respectivas estadísticas.*

Líder funcional

Entre sus principales tareas están:

- *Registrar la solicitud de cambio en la herramienta establecida por el proceso.*
- *Avalar el trámite de la solicitud de cambio.*
- *Dar su concepto en el Comité de cambios (CAB) o el Comité de cambios de emergencia (ECAB) para evaluar, aprobar y priorizar las solicitudes que impacten los servicios de TI.*
- *Emitir su concepto para aprobar la implementación en producción de la solicitud de cambio así como recomendar mejoras, durante la reunión del CAB o el ECAB.*
- *Asegurar el monitoreo de la construcción, pruebas e implementación de las solicitudes de cambio.*
- *Garantizar el registro del resultado de las pruebas que soportan la solicitud de cambio.*

Administrador de base de datos o DBA

Entre sus principales funciones están:

- *Debe mantener y operar las bases de datos que conforman el sistema de información.*
- *Ayuda a realizar los despliegues del software.*
- *Verificar o ayudar a verificar la integridad de datos.*
- *Mantener la disponibilidad, esto significa que los usuarios tengan acceso a los datos cuando lo necesiten.*
- *Limitar a los usuarios para que ejecuten únicamente las operaciones permitidas.*
- *Hacer respaldos de la base de datos y almacenarlos de manera que se minimice el riesgo de daño o pérdida de los mismos.*

Arquitecto de software

Al momento de implantar el sistema en el ambiente productivo, muchas veces es necesario realizar ajustes finos sobre el sistema, en particular una vez que el sistema ya está operando en el ambiente de uso definitivo. La participación del arquitecto puede estar enfocada en realizar ajustes finos de la aplicación, con el fin de lograr un funcionamiento óptimo de la misma.

9 USO Y APROPIACION

9.1 PROBLEMAS FRECUENTES.

- Se detectan oportunidades de mejora en términos de la gestión de la resistencia al cambio cuando se publican o entran en producción nuevos sistemas de información.
- Es necesario robustecer el proceso de capacitación que ofrecen los proveedores sobre los sistemas desarrollados.
- Se observa que algunos contratos de desarrollo de *software* no contemplan una etapa de transferencia de conocimiento entre el proveedor y la entidad.
- En las entidades en las que hay un alto nivel de rotación, se evidencia que no hay un proceso de entrenamiento que permita a los nuevos empleados usar apropiadamente los sistemas de información desarrollados.
- Hay una necesidad de contar con personal experto en el área de TI que pueda brindar soporte técnico sobre las aplicaciones adquiridas por la entidad.

9.2 ASPECTOS GENERALES.

9.2.1 Líneas de acción para uso y apropiación

En la etapa de uso y apropiación se busca que el usuario esté en capacidad de desarrollar competencias que sean reproducidas en su entorno laboral y personal, de forma que pueda usarlo para su crecimiento profesional en la entidad.

Las actividades que se desarrollan para este propósito se deben orientar a mostrar que los procesos tecnificados, mediante el desarrollo del sistema de información, mejorarán el servicio a los usuarios y al cliente interno, y que sin dudas apoyan el desarrollo y buen desempeño de los funcionarios que harán uso de él.

Se debe tener claro que “Uso y Apropiación” no es una de las etapas finales del ciclo de desarrollo de sistemas de información, de hecho, durante todo el ciclo se deben contemplar aspectos encaminados a mejorar la usabilidad, de forma que el funcionario se sienta cómodo y disminuya la resistencia al cambio, tal como se muestra en la siguiente figura:

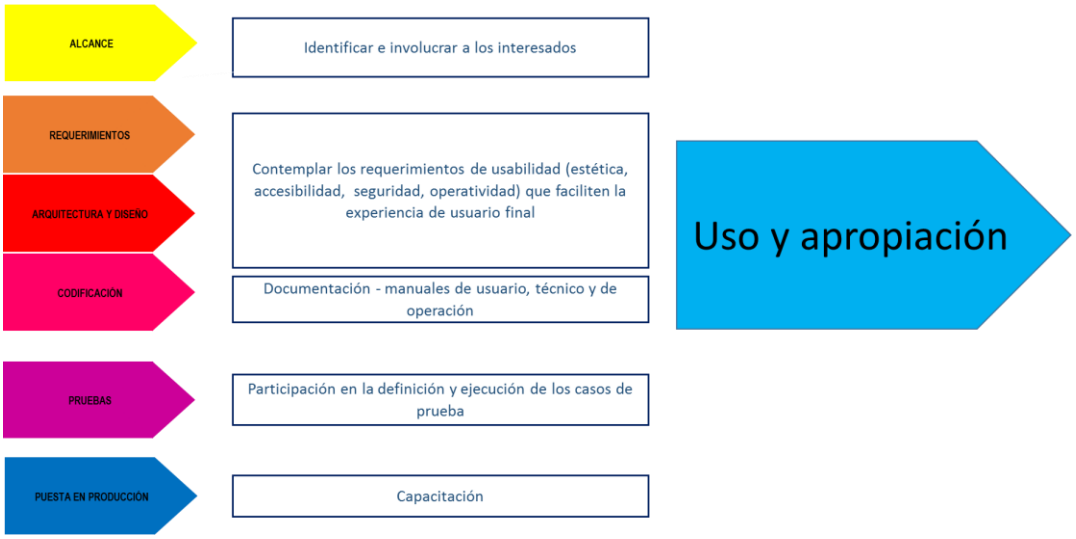


Figura 69. Aspectos sobre uso y apropiación en las etapas del ciclo de desarrollo. Fuente: Corporación Colombia Digital -CCD.

En cada etapa se generan insumos para facilitar el uso y apropiación o para desarrollar las actividades relacionadas. Sin embargo, desde las etapas iniciales se debe identificar e involucrar al personal clave para el desarrollo y uso del sistema, este grupo tendrá diferentes roles y participación para lograr el éxito. Acá se habla no solo de los patrocinadores,

directivas, jefes de área, sino también de los usuarios conocedores del negocio y promotores a futuro de los beneficios que se obtendrán con el desarrollo; y, por supuesto, la contraparte que en este caso es el proveedor.

Durante la definición de requerimientos, la arquitectura, el diseño y la codificación del nuevo sistema, se deben considerar los aspectos que faciliten posteriormente la experiencia del usuario final. Dichos aspectos se refieren principalmente al estilo de presentación, visualización de la información y procesos de navegación entre pantallas.

Mejor práctica: *No se debe olvidar que el sistema de información puede ser usado a través de diferentes medios tecnológicos, como por ejemplo computadores de escritorio o dispositivos móviles (tabletas, smartphones, etc.), de tal manera que el estilo y la usabilidad deben estar particularizados para cada uno.*

Así mismo, un insumo relevante para la etapa de uso y apropiación es la documentación del sistema, por lo tanto como parte de la codificación debe exigirse que los manuales de usuario, técnico y de operación estén debidamente desarrollados y actualizados.

Mejor práctica: *Asegurar que el sistema de información cuente con documentación de usuario, técnica y de operación debidamente actualizada, esto facilita la transferencia de conocimiento hacia los usuarios, hacia la dirección y hacia los servicios de soporte tecnológico.*

La etapa de pruebas puede constituirse como la primera interacción de los usuarios finales con el nuevo sistema, por lo tanto es primordial la participación en la definición de los casos de pruebas y en la ejecución misma, procurando que este momento de verdad se desarrolle de forma satisfactoria. Acá los usuarios tendrán la oportunidad de verificar las bondades del sistema y proponer mejoras que faciliten su uso y adopción.

Mejor práctica: *Puesto que la totalidad de los usuarios finales no podrá participar en las pruebas, se debe seleccionar los usuarios clave, que sirvan de líderes y formadores hacia los demás funcionarios que interactuarán directamente con el nuevo desarrollo y tengan influencia positiva.*

Igualmente como parte de la puesta en producción, será indispensable realizar la capacitación a los diferentes actores que interactuarán con el nuevo sistema, de tal manera que se desarrollen no solo las habilidades técnicas y operativas, sino que sirva para concientizar a los funcionarios sobre la nueva forma de hacer las cosas apoyándose en la tecnología, pero en busca siempre del cumplimiento de la misión estratégica de la entidad.

Con estos insumos generados en cada etapa del ciclo de desarrollo del sistema de información, no se parte de cero, sino que se tiene ya un camino abonado para que el uso y apropiación del nuevo sistema se realice de una forma más fácil y efectiva.

A su vez, con el cambio que se busca en la entidad y como producto propio de las actividades de uso y apropiación, se obtendrá: una matriz de interesados que definirá principalmente el público objetivo y su rol; unos planes de sensibilización, capacitación y transferencia de conocimiento, que como se muestra más adelante, constituyen los ejes fundamentales para este proceso; y la identificación de oportunidades de mejora como insumo para el mantenimiento del nuevo sistema. Vea la siguiente figura:

Figura 70. Principales productos del componente de uso y apropiación. Fuente: Corporación Colombia Digital - CCD.

Como elemento transversal importante se encuentra la medición a través de indicadores, de tal manera que se determine el nivel de satisfacción, adopción e impacto que el uso del nuevo sistema esté generando. El proceso de seguimiento se soportará en la medición de los indicadores que se definan.

Mejor práctica: *Como en todo proceso el seguimiento a los indicadores permitirá que se tenga un mejoramiento continuo, no obstante se debe tener claro que los indicadores que se definan sean medibles y cuantificables, y que se debe establecer el mecanismo periódico para su revisión.*

Elemento transversal – Talento TI: *El liderazgo de la etapa de uso y apropiación está en cabeza del gerente de proyecto, quien deberá asegurar la interacción entre las partes para su ejecución.*

Elemento transversal – Riesgos: Arraigo en prácticas tradicionales. Este riesgo se materializa cuando se encuentran funcionarios con poco interés e interacción con el nuevo sistema que dificultan el proceso de adopción y en ocasiones generan influencia negativa en los demás interesados.

9.2.2 Matriz de interesados

Si bien en la etapa de requerimientos se identifican las clases de usuarios, en este punto se actualiza y complementa con otros posibles interesados que forman parte del público objetivo para la sensibilización. La siguiente tabla presenta el rol de los posibles actores que más comúnmente son identificados.

Rol	Asignación	Responsabilidades / Autoridad
Indique en cada fila, cada uno de los roles de los interesados en el proyecto de desarrollo	Indique la identificación de la persona asignada para el desempeño del rol, es importante conocer nombre, cargo y área en la que labora, y datos de contacto	Enuncie las responsabilidades y la autoridad que tendrá el rol. En caso de requerirse una descripción detallada del rol, haga referencia aquí al respectivo documento de definición detallada del rol
Rol	**Asignación**	**Responsabilidades / Autoridad**
Directivas (patrocinadores)	Presidente, directores, secretario general	Promover y apoyar
Dirección de tecnologías de información	Director de tecnología o quien haga sus veces - CIO	Incentivar y revisar
Coordinadores/Líderes	Personal interno clave para la promoción del nuevo sistema (Ej: gestión humana, comunicaciones)	Promover y adoptar
Auditores	Personal interno o externo de auditoría	Revisar y evaluar
Administradores	Personal interno que interactúa directamente con el nuevo sistema para su administración	Promover y adoptar
Mesa de ayuda	Personal interno o externo encargado de atender las inquietudes y dar soporte de primer nivel	Apoyar
Usuarios	Funcionarios de las áreas organizacionales que se verán impactadas por el uso del nuevo sistema	Promover y adoptar
Proveedor	Tercero que conjuntamente promueve y apoya la planeación y ejecución de los planes de sensibilización y capacitación	Promover y apoyar

Tabla 42. Matriz de interesados. Fuente: Corporación Colombia Digital - CCD.

Como se observa en la columna de responsabilidades, con la participación de estos actores se logra:

- Diseñar y ajustar el plan institucional de comunicaciones para que incorpore acciones de uso y apropiación de nuevos desarrollos.
- Comprometer al gobierno de TI con acciones que soporten los planes de sensibilización y capacitación permanentes para nuevos sistemas.
- Usar adecuadamente los recursos tecnológicos e interactuar entre las áreas de negocio para procurar la migración gradual del uso de los nuevos sistemas.
- Mejorar los canales de acceso y respuesta para dar soporte a los servicios de los nuevos sistemas.

Elemento transversal – Riesgos: *Asignación deficiente o inadecuada de interesados. Este riesgo se materializa cuando no se detecta un rol clave para la etapa de uso y apropiación, o cuando el funcionario asignado no responde a las expectativas de su rol y de sus responsabilidades.*

9.2.3 Ejes fundamentales para el uso y apropiación

El uso y apropiación no puede ser concebido como un componente individual y separado de las políticas y directrices institucionales, pues pese a ser conscientes de los beneficios que conlleva el uso del nuevo sistema, muy seguramente habrá barreras que limiten su adopción, por lo tanto dicho componente debe transcender esos obstáculos para no ser una simple sugerencia o recomendación, sino convertirse en parte de una política institucional encaminada a lograr la nivelación de funcionarios y usuarios por medio de tres ejes fundamentales: 1) sensibilización, 2) capacitación y 3) transferencia de conocimiento, tal como se indica en la siguiente figura:

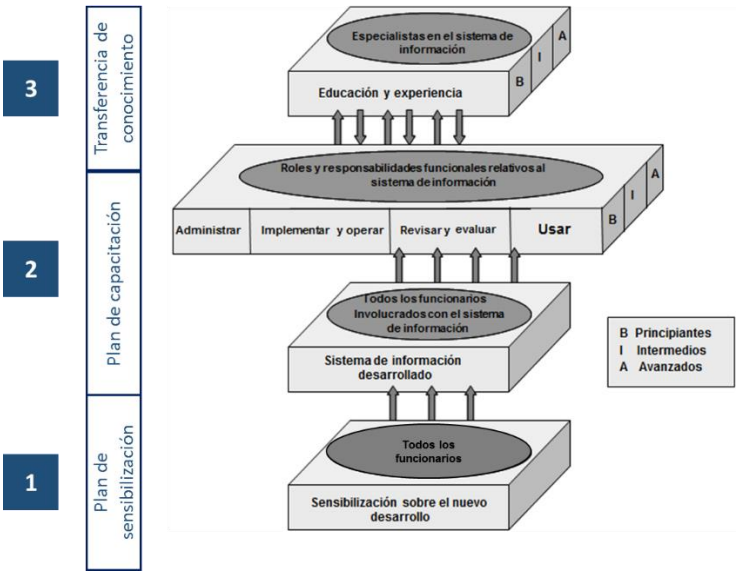


Figura 71. Ejes fundamentales para el uso y apropiación. Fuente: Corporación Colombia Digital - CCD.

Como se observa, el plan de sensibilización es más general, va dirigido a todos los funcionarios de la entidad y su intención principal es dar a conocer el sentido y beneficios del nuevo desarrollo. Entre tanto el plan de capacitación y la transferencia de conocimiento se enfocan en un grupo más específico que incluye a aquellos funcionarios involucrados directamente con el sistema de información y con responsabilidades funcionales relativas a dicho sistema. Los usuarios más experimentados o con mayor nivel de formación serán los que logren la transferencia de conocimiento y serán considerados como especialistas del sistema.

Es importante tener presente que hay diferentes tipos de usuarios, los encargados de administrar, implementar, operar, revisar y evaluar; y los usuarios funcionales propiamente dichos, que a su vez tienen un nivel de entendimiento o experiencia diverso. Puede haber usuarios principiantes, intermedios y avanzados, por ello es importante tener un tipo de contenido adecuado para cada tipo y para cada nivel.

Mejor práctica: *El éxito de la interacción con cada usuario dependerá de la certeza de los contenidos que se manejen según el tipo y nivel; por lo cual los planes de sensibilización y capacitación deben estar acordes con dicho nivel, con su lenguaje, capacidades y expectativas.*

9.2.3.1 Plan de sensibilización.

El plan de sensibilización debe tener el mensaje claro respecto a que el nuevo sistema se entiende como un elemento natural del entorno en el que se desempeña cada funcionario y crea una serie de condiciones que se convierten en la base de la operación institucional para mejorar el servicio. El uso y apropiación va más allá de “aprender y conocer el nuevo sistema”, en realidad genera valor mediante su aporte al logro de las metas estratégicas de la entidad.

En ese orden de ideas, el plan de sensibilización se enfoca en los siguientes elementos:

Nivel	Atributo	Objetivo de aprendizaje	Ejemplos de metodología	Prueba de medición	Marco de tiempo para resultados
Información	¿Qué? ¿Para qué?	Reconocimiento y retención	Medios (videos, carteleras, charlas, etc.)	Encuestas (falso/verdadero, escogencia múltiple)	Corto plazo

Tabla 43. Elementos del plan de sensibilización. Fuente: Corporación Colombia Digital - CCD.

En el plan de sensibilización el nivel es la "información", es decir qué conceptos se van a comunicar para que los usuarios se familiaricen con el nuevo sistema, para ello se utilizarán diferentes medios. El plan de sensibilización requiere permanencia y continuidad, pero sus resultados se ven en el corto plazo; para medir su efectividad comúnmente se utilizan encuestas que verifican que la información está llegando y se está quedando en el público objetivo.

Elemento transversal – Riesgos: *Bajo impacto del plan de sensibilización. Este riesgo se materializa cuando no se cubre un porcentaje significativo de la población objetivo con el plan de sensibilización.*

Mejor práctica: *Para lograr la mayor participación, se deben coordinar las actividades de sensibilización con las áreas de gestión humana y comunicaciones de la entidad, puesto que conocen la disponibilidad de los funcionarios para la asistencia a las sesiones que se programen, lo que evita que se crucen con otras actividades de la entidad; así mismo tienen claras las épocas de vacaciones que comúnmente se solicitan, con lo que se eligen fechas distintas para asegurar una mayor cobertura y eficacia del proceso.*

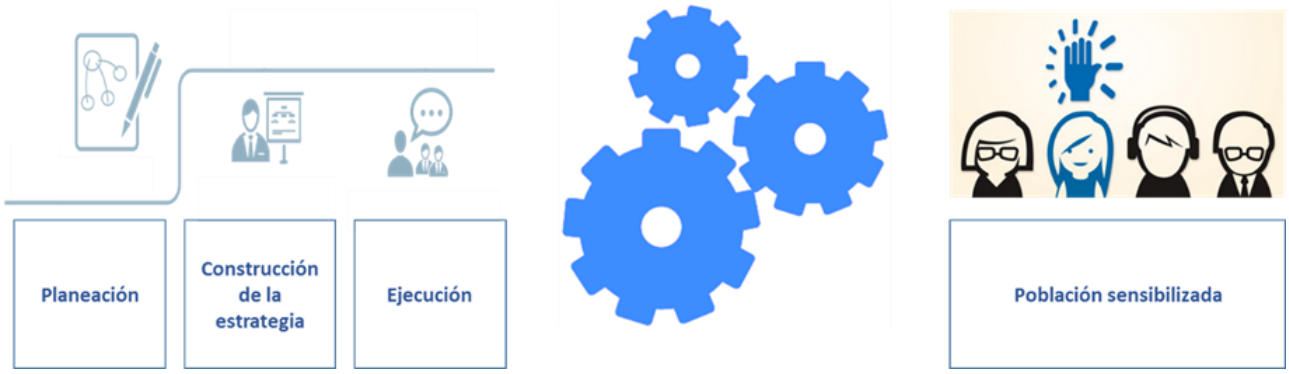


Figura 72. Mecanismo de desarrollo del plan de sensibilización. Fuente: Corporación Colombia Digital - CCD.

La figura anterior indica que una vez considerados estos elementos, el mecanismo para el desarrollo del plan de sensibilización debe seguir unas actividades de:

a) Planeación: el plan se materializa en un documento cuya estructura incluirá un propósito, un alcance, unos beneficios, la estrategia y las recomendaciones

b) Construcción: consiste en el diseño, elaboración y preparación del material que se haya definido en la estrategia

c) Ejecución: es la publicación del material, envío de comunicaciones o realización de las conferencias según la estrategia y acorde con el calendario de ejecución.

a) Planeación:

Consiste en la validación del enfoque, conformación del equipo de trabajo, estudio de los materiales que se utilizarán, costos, identificación de medios de comunicación, definición de los temas que se desarrollarán y preparación del plan de trabajo. Se establece:

- Tema
- Grupo objetivo
- Canal o medio de divulgación
- Periodicidad de publicación
- Textos y material de apoyo para la campaña

Un modelo de la estructura del documento del plan de sensibilización puede ser el que se presenta en la siguiente tabla:

Propósito	Alcance	Beneficios	Estrategia	Recomendaciones	Anexos
Indique la intensión que persigue el plan de forma general	Plantee los objetivos sobre cobertura e impacto	Indique la frecuencia y prioridad con la que se genera la comunicación	Indique el esquema de trabajo, fases y materiales a ser utilizados en las campañas	Indique las mejores prácticas para lograr la efectividad del plan, identifique las restricciones que puedan haber en la entidad y su manejo	Adjunte el material de las campañas o sus diseños y el calendario de ejecución del plan

Tabla 44. Estructura típica de un documento de plan de sensibilización. Fuente: Corporación Colombia Digital - CCD.

La estrategia debe orientarse a:

- Hacer llegar la campaña al público más amplio posible. Resulta conveniente utilizar criterios de multiplicación para aumentar al máximo la difusión del mensaje.
- No asumir una actitud alarmista o excesivamente negativa respecto al tema, en lo posible detallar los problemas o riesgos, con contextos del mundo real.
- Modificar el comportamiento del grupo destinatario en relación con el uso del nuevo sistema.
- El mensaje emitido, los canales utilizados y el emisor deben ser influyentes y creíbles, pues de lo contrario el grupo destinatario podría estar menos dispuesto a escucharlo. Se debe buscar el respaldo de las áreas claves de la entidad para tener mayor poder de convocatoria y atención.

Como aquí la información es el elemento clave, no se puede dejar de lado el contenido, los canales de distribuciones disponibles y potenciales, la periodicidad con que deba publicarse la información y la audiencia a quien va dirigida cada comunicación. Para ello hay que tener en cuenta:

Tipo de comunicación	Medio o canal de distribución	Periodicidad	Responsable de generar	Audiencia (dirigido a)
Indique el tipo de información o comunicación que deba fluir	Indique el medio a través del cual se debe distribuir la comunicación	Indique la frecuencia y prioridad con la que se genera la comunicación	Indique el nombre y rol del responsable de la elaboración y distribución de la comunicación	Indique los nombres y roles de los destinatarios de la comunicación
Objetivo del nuevo desarrollo Plan de implementación Beneficios Aspectos relevantes que deban ser socializados	Intranet Papel tapiz Boletines Videos Conversatorios Redes sociales Material promocional	Calendario de ejecución del plan	Proveedor Area de comunicaciones y/o RH	Público en general Grupos o áreas de interés Directivas

Tabla 45. Modelo de comunicación. Fuente: Corporación Colombia Digital - CCD.

Mejor práctica: *Con el fin de lograr la mayor cobertura posible, se debe considerar el uso de plataformas o escenarios disponibles para llegar a los funcionarios que se encuentren fuera de las sedes o ciudades principales en las que tiene presencia la entidad. La tecnología ofrece espacios de teleconferencia o telepresencia, que pueden ser aprovechados para estos propósitos.*

Elemento transversal – Riesgos: *Recursos limitados para la ejecución de las campañas de sensibilización. Este riesgo se materializa cuando no se han contemplado el presupuesto y personal que destinado para el diseño y elaboración del material y las comunicaciones del plan, o cuando no se ha solicitado explícitamente al proveedor que suministre dicho material.*

b) Construcción:

Esta fase del plan incluye la creación de los materiales para la sensibilización. Para el desarrollo se deben utilizar los medios de comunicación disponibles en la entidad como:

- Carteleras – ubicadas al ingreso, recepción u otros sitios estratégicos.

- Intranet – frases de sensibilización y noticias inmediatas
- Boletines – textos más elaborados y explicativos
- Conferencias – programadas según la disponibilidad de los funcionarios de cada área, es conveniente que se realice en espacios como un auditorio para toda la comunidad empresarial.

c) Ejecución:

Esta fase incluye la publicación de los materiales para la sensibilización, la cual debe realizarse acorde con el calendario de ejecución que se haya construido.

9.2.3.2 Plan de capacitación.

Está orientado a transferir el conocimiento de conceptos y el uso apropiado y efectivo de los nuevos sistemas de información o *software* desarrollado, con el objetivo de potencializar las capacidades, destrezas, habilidades y competencias de los empleados que lo usarán y posibilitar una mejor prestación de los servicios que ofrece la entidad al ciudadano.

Mejor práctica: *El plan debe contar con todos los elementos de planeación, ejecución, medición y gerencia que les permitan a las personas apropiar el nuevo desarrollo y evidenciar los resultados en la prestación de un mejor servicio y en el cumplimiento de la misión institucional.*

La siguiente figura ilustra los elementos que se deben tener en cuenta en la construcción del plan:

Figura 73. Elementos del plan de capacitación. Fuente: Corporación Colombia Digital - CCD.

Propósito.

Describe las razones institucionales, las motivaciones y las transformaciones que la entidad desea lograr con la capacitación. Debe estar orientada a lograr una gestión pública eficiente y eficaz, con la prestación de un servicio confiable, mediante el mejoramiento de competencias laborales.

Debe asegurar que su desarrollo está alineado con las estrategias y políticas de la entidad, basado en los siguientes principios:

- **Complementariedad**: La capacitación se concibe como un proceso complementario de la planeación, por lo cual debe tenerla en cuenta y orientar sus propios objetivos en función de los propósitos institucionales.

- **Integralidad:** La capacitación debe contribuir al desarrollo del potencial de los empleados en su sentir, pensar y actuar, articulando el aprendizaje individual con el aprendizaje en equipo y el organizacional.

- **Objetividad**: La formulación de políticas, de planes y programas de capacitación debe ser la respuesta a un diagnóstico de necesidades de capacitación previamente realizado con procedimientos e instrumentos técnicos propios de las ciencias sociales y las administrativas.

- **Participación**: Todos los procesos que hacen parte de la gestión de la capacitación, tales como detección de necesidades, formulación, ejecución y evaluación de planes y programas, deben contar con la participación activa de los empleados.

- **Prevalencia del interés de la organización**: Las políticas, los planes y los programas responderán fundamentalmente a las necesidades de la organización.

- **Economía**: Siempre se buscará el manejo óptimo de los recursos destinados a la capacitación, mediante acciones que pueden incluir el apoyo interinstitucional.

- **Énfasis en la práctica**: La capacitación se impartirá privilegiando el uso de metodologías que hagan énfasis en la práctica, en el análisis de casos concretos y en la solución de problemas.

Alcance

El alcance describe los objetivos específicos en términos de lo que se quiere lograr con los empleados. Define el tipo de capacitación entre los que pueden ser:

- **Formación:** Su propósito es impartir conocimientos básicos orientados a proporcionar una visión general y amplia con relación al contexto.

- **Actualización:** Se orienta a proporcionar conocimientos y experiencias derivados de recientes avances científico – tecnológicos, en una determinada actividad**.**

- **Especialización:** Se orienta a la profundización y dominio de conocimientos y experiencias o al desarrollo de habilidades, respecto a un área determinada.

- **Perfeccionamiento:** Se propone completar, ampliar o desarrollar el nivel de conocimientos y experiencias, con el fin de potenciar el desempeño de funciones técnicas, profesionales, directivas o de gestión.

- **Complementación:** Su propósito es reforzar la formación, para obtener parte de los conocimientos o habilidades demandados por un rol.

Los tipos de capacitación pueden ser desarrollados a través de las siguientes modalidades:

Modalidad Presencial: aquella cuyas actividades se realizan en un espacio físico donde interactúan los participantes.
Modalidad Virtual: aquella que privilegia los medios electrónicos para la transmisión y asimilación de conocimientos.

Tanto en los tipos como en las modalidades, la capacitación puede darse en los siguientes niveles:

Nivel básico o principiante: Se orienta a personal que se inicia en el desempeño de una ocupación o área específica en la Empresa. Tiene por objeto proporcionar información, conocimientos y habilidades esenciales requeridos para el desempeño en la ocupación.

Nivel intermedio: Se orienta al personal que requiere profundizar conocimientos y experiencias en una ocupación determinada o en un aspecto de ella. Su objeto es ampliar conocimientos y perfeccionar habilidades con relación a las exigencias de especialización y mejor desempeño en la ocupación.

Nivel avanzado: Se orienta a personal que requiere obtener una visión integral y profunda sobre un área de actividad o un campo relacionado con esta. Su objeto es preparar cuadros ocupacionales para el desempeño de tareas de mayor exigencia y responsabilidad dentro de la empresa.

Herramientas y recursos

Comprende la planeación del recurso humano que participará en la capacitación, los participantes, expositores, personal de apoyo y los profesionales involucrados. Incluye definir los requerimientos del lugar donde se desarrollará la capacitación, pues debe contar con un ambiente, mobiliario y recursos tecnológicos adecuados para una ejecución exitosa. Especifica la documentación técnica que se utilizará durante la capacitación, el material que será usado y entregado a los participantes. Se debe planificar el monto de los recursos de inversión que se destinarán.

Lineamientos pedagógicos

La entidad debe establecer las orientaciones conceptuales, pedagógicas y temáticas del plan de capacitación, con el fin de fortalecer las competencias de los participantes y la capacidad técnica de las áreas que aportan a cada uno de los procesos, además de elevar el nivel de compromiso de los empleados respecto con los planes, los programas y los proyectos de la entidad.

Mejor práctica: *Enfocar la formación en la resolución de problemas, como una oportunidad para aprender a través de interrogantes, así como en el trabajo en equipo, con planteamientos de análisis de problemas o retos institucionales para el cumplimiento de las metas propuestas.*

Evaluación

La entidad debe definir y planificar todos los instrumentos necesarios para medir el plan, el impacto de la formación y, sobre todo, los resultados organizacionales. También se deben establecer los mecanismos de retroalimentación para generar posibles mejoras y los criterios de evaluación de conocimiento aprendido, las competencias logradas, la información presentada y la metodología usada.

La generación de reportes o informes debe evidenciar las actividades desarrolladas correctamente y que favorecen la capacitación, las actividades realizadas incorrectamente y que afectan el desarrollo de la capacitación, y todas aquellas lecciones aprendidas que optimicen capacitaciones futuras.

Gerencia del plan

La gerencia se constituye como un elemento que debe estar presente durante todo el ciclo de vida del plan, es una acción que acompaña la planeación, la ejecución y la medición de los resultados. Esta actividad debe contar con todos los instrumentos que la entidad defina para medir, detectar desviaciones y establecer acciones preventivas y correctivas durante la ejecución del plan.

9.2.3.3 Transferencia de conocimiento.

El conocimiento se ha convertido en un recurso que da mayores niveles de agregación de valor a las instituciones y a los procesos, mejora el accionar del Estado y la relación con el ciudadano. Por eso es que la transferencia de conocimiento se constituye como un gran desafío para que las entidades puedan gestionar y optimizar todas las actividades de captura, creación y difusión de nuevos saberes como la aplicación de una nueva tecnología o desarrollo de *software* o una innovación que, transferidas adecuadamente, pueden mejorar el desempeño y las capacidades organizacionales de la entidad, lo que a la larga se traduce en impactos positivos para la sociedad.

Para los procesos de transferencia de conocimiento de nuevos de desarrollo de *software* o de sistemas de información, es importante que la entidad, en coordinación con la empresa o fábrica de desarrollo, tenga en cuenta tres pasos para lograr que un proceso de transferencia de conocimiento sea exitoso. (i) Definir cuál es el tipo de conocimiento que se quiere transferir, por lo que se debe precisar la información y el valor que se le va dar. (ii) De qué manera se va transferir, que se refiere a los medios a través de los cuales se entregará el conocimiento. (iii) Con qué propósitos se quiere transferir el conocimiento, que involucra a quiénes está dirigido y el impacto que se quiere lograr. Asociado a este proceso, tienen que existir unos mecanismos de evaluación y de medición para determinar el impacto y retroalimentación.

La siguiente figura expone los elementos presentes en un proceso de transferencia de conocimiento y la interrelación entre cada uno de ellos:

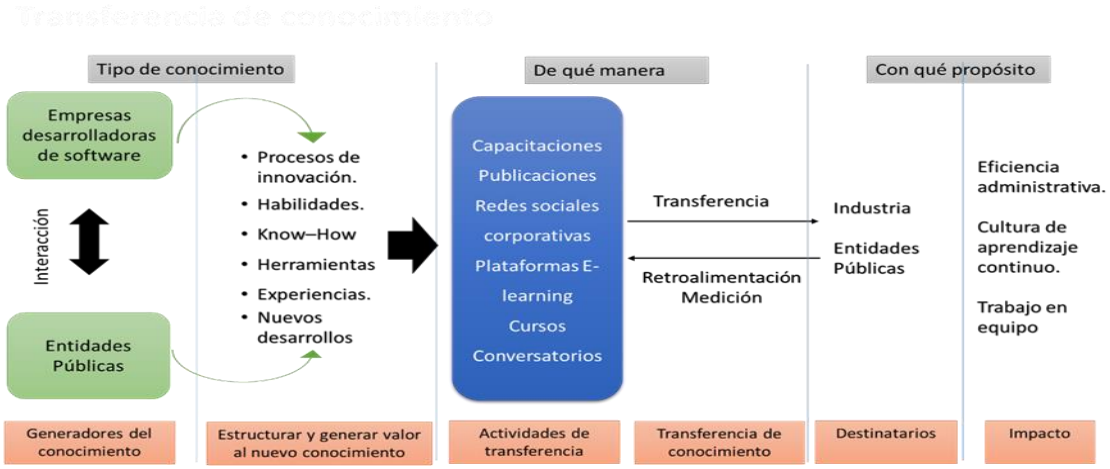


Figura 74. Elementos que componen la transferencia de conocimiento. Fuente: Corporación Colombia Digital - CCD.

9.2.4 Opciones de mejora

Las opciones de mejora se encaminan a desarrollar mecanismos que puedan ser reproducidos en el interior de la entidad y que conduzcan a una mejor adaptación, un mejor

uso y una mejor apropiación del desarrollo de *software* o del sistema de información. Los indicadores y el seguimiento que se le haga a la estrategia de uso y apropiación del nuevo *software* arrojarán los aspectos en los que será importante trabajar.

Las opciones de mejora y su implementación deben estar enfocadas en alguno de los siguientes temas:

- Definición de políticas institucionales que marcarán el rumbo de la estrategia de uso y apropiación
- Redefinición de procesos y procedimientos institucionales de uso y apropiación
- Actualización de los planes de medios, promulgación, sensibilización, capacitación o transferencia de conocimiento
- Caracterización de usuarios para el acceso y la consulta de información a todos los públicos
- Apoyo a los usuarios en la adopción de nuevas herramientas o desarrollos
- Programas de nivelación en el uso y apropiación de los sistemas de información que permitan reducir brechas en temas de competencias profesionales

9.2.5 Medición del nivel de satisfacción, adopción e impacto a través de indicadores

Aspecto cubierto	Indicador	Descripción
Satisfacción	Nivel de percepción de los usuarios	Alto: percepción positiva en el uso del nuevo desarrollo Medio: no se perciben cambios significativos con el uso del nuevo desarrollo Bajo: percepción negativa en el uso del nuevo desarrollo
Adopción	Población capacitada	Personal cubierto con las acciones de capacitación en el uso del nuevo desarrollo
	Cobertura	# de personas que usan el desarrollo / población total objetivo
Impacto	Población beneficiada	Personal interno total cubierto que obtiene un beneficio con el desarrollo realizado
	Población sensibilizada	Personal interno cubierto con las acciones de sensibilización para la adopción del nuevo desarrollo

Tabla 46. Medición del nivel de satisfacción, adopción e impacto a través de indicadores. Fuente: Corporación Colombia Digital - CCD.

10 MANTENIMIENTO

10.1 PROBLEMAS FRECUENTES.

- La calidad del *software* se deteriora por la falta de planeación del proceso de mantenimiento. En las entidades las correcciones se realizan con precipitación, las metodologías no suelen contemplar la participación del usuario y, si no satisfacen las necesidades, hay que realizar un esfuerzo adicional para adaptar el *software,* lo que implica un mayor costo.
- Se detecta con alguna frecuencia que la documentación existente sobre el sistema de información está desactualizada o incompleta y el código fuente no está correctamente documentado, lo cual conlleva a que los desarrolladores utilicen más tiempo para comprender el código.
- Los sucesivos cambios producidos por el mantenimiento hacen que el código sea más difícil de modificar y aumentan los costos del proyecto.
- No se realizan las pruebas adecuadas para identificar los problemas que llevan a modificar el *software*, por lo que se invierte tiempo y esfuerzo en encontrar los fallos.
- Se observa que para los sistemas de información muy robustos es necesario contar con herramientas de gestión con las que se atiendan las solicitudes de mantenimiento.
- Se detectan oportunidades de mejora en la pertinente gestión de requerimientos por parte del proveedor, para poder suministrar el servicio de mantenimiento: permisos, accesos, licencias, etc.
- En algunos casos se observa que los proyectos definidos como mantenimiento de *software* son en realidad proyectos de desarrollo.
- Se evidencia una alta rotación del personal que asigna el proveedor al proyecto de mantenimiento.

10.2 ASPECTOS GENERALES.

Se define el mantenimiento del *software* como “la modificación de un producto de *software* después de haber sido entregado a los usuarios o clientes, con el fin de corregir defectos, mejorar el rendimiento u otros atributos, o adaptarlo a un cambio en el entorno”. (Estándar IEEE 1219 (IEEE, 1993)).

La fase de mantenimiento se centra en el cambio asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del *software* y a las mejoras producidas según los requisitos cambiantes del cliente. El objetivo es modificar el producto *software* existente preservando su integridad. Este proceso incluye la migración y retirada del producto.

En el ciclo de vida del desarrollo, el mantenimiento comienza después de un período de garantía o soporte posterior a la implementación de actividades de entrega. Históricamente, el desarrollo de *software* ha tenido un perfil mucho más alto que el mantenimiento, además que no ha recibido el mismo grado de atención que las otras fases de desarrollo. El mantenimiento consume una parte muy importante de los recursos financieros en el ciclo de vida del *software*.

Elemento transversal – Medición de esfuerzo: *La experiencia por juicios de expertos se utiliza a menudo para estimar el esfuerzo de mantenimiento; está claro que el mejor enfoque para hacerlo es combinar los datos históricos y la experiencia. El costo de una modificación (en términos del número de personas y la cantidad de tiempo) es derivado de la estimación del mantenimiento de datos históricos por medio de un programa que realiza el cálculo. Los atributos que pueden ser tenidos en cuenta para esta medición son:*

- *El proceso de mantenimiento (tamaño, complejidad, calidad, comprensibilidad, mantenibilidad y esfuerzo)*

- *El personal*

IEEE 14764 identifica las actividades primarias de mantenimiento de *software* como la implementación de procesos, problema y análisis de la modificación, la modificación de la migración, y el retiro. El mantenimiento debe tomar artefactos de *software* desde el desarrollo (por ejemplo código o documentación), modificarlo y mantenerlos durante el ciclo de la vida del desarrollo.

El mantenimiento es necesario para asegurar que el *software* continúa satisfaciendo los requisitos del usuario y puede aplicarse a cualquier desarrollo, cualquiera sea la metodología que haya usado (por ejemplo espiral o lineal). Los productos de *software* cambian debido a las acciones correctivas y no correctivas.

Estas son cinco características claves que incluyen las actividades de mantenimiento:

- Mantener el control sobre el funcionamiento del *software* día a día
- Mantener el control sobre la modificación
- Perfeccionar las funciones existentes
- Identificar amenazas a la seguridad
- Prevenir el bajo rendimiento a niveles inaceptables

10.2.1 Tipos de mantenimiento

Tres tipos de mantenimiento han sido definidos: correctivo, adaptativo y perfectivo. IEEE 14764 incluye una cuarta categoría denominada preventiva, como se muestra en la siguiente figura:

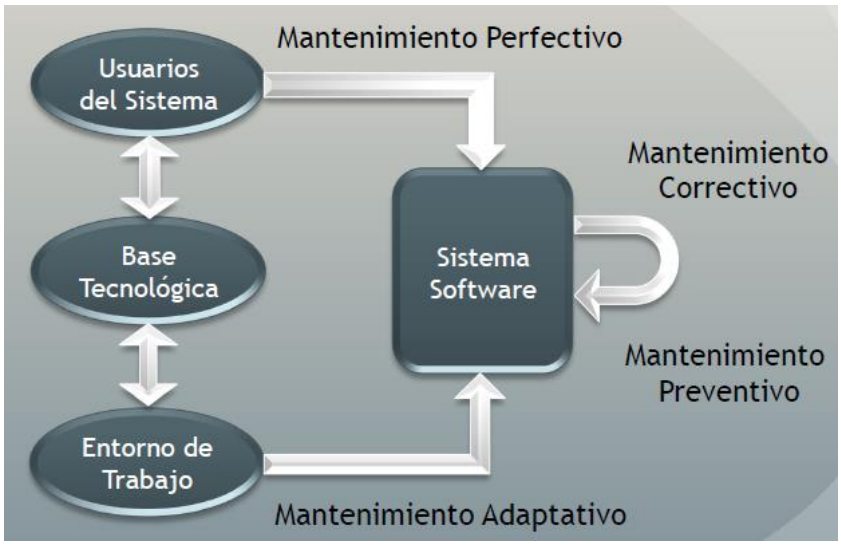


Figura 75. Tipos de mantenimiento. Fuente: Grupo Kybele, 2012.

Mantenimiento correctivo

Aun habiendo superado las etapas de prueba y verificación, el *software* puede contener defectos, por lo que este tipo de mantenimiento tiene como objetivo encontrar y eliminar esos defectos que pueden darse por:

- Procesamiento: salidas incorrectas en un programa
- Rendimiento: demasiado tiempo de respuesta
- Programación: diseño inconsistente de un sistema
- Documentación: diferencias entre la funcionalidad de un programa y el manual de usuario

En la siguiente figura se puede ver que la mayor parte de los defectos se encuentran en la fase de levantamiento de los requisitos, luego en la codificación y por último en el diseño:

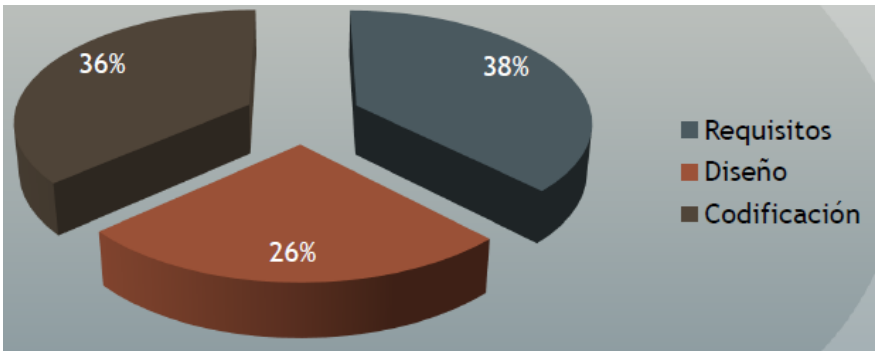


Figura 76. Origen de los defectos de software. Fuente: Grupo Kybele, 2012.

Mantenimiento adaptativo

Este tipo de mantenimiento responde a una situación cuando se produce algún cambio en el *software* o *hardware* del entorno en el que se ejecuta el sistema. Estos cambios pueden deberse a:

- Cambio en el sistema operativo
- Cambio del tipo de arquitectura en la que se ejecuta
- Entorno de desarrollo del *software* (nuevos elementos y herramientas)

Mejor práctica: *Es necesario realizar pruebas para validar los cambios. Las pruebas verificarán que no se han introducido otros errores, incluso el cambio más pequeño puede inducir defectos que reduzcan la calidad y la fiabilidad del software.*

Mantenimiento perfectivo

Este tipo de mantenimiento está asociado con la modificación de un producto después de la entrega para proporcionar mejoras para los usuarios, en la documentación del programa y el rendimiento del *software.*

A su vez, este tipo de mantenimiento se puede dividir en dos:

- Mantenimiento de ampliación: orientado a incorporar nuevas funcionalidades.

- Mantenimiento de eficiencia: busca la mejora de las prestaciones del sistema en tiempo de ejecución.

Mantenimiento preventivo

Modifica un producto de *software* después de la entrega para detectar fallas latentes antes de que sean fallas operativas. De igual manera, el mantenimiento preventivo sirve para mitigar o evitar las consecuencias de las fallas, para ello se debe:

- Comprobar la validez de los datos de entrada
- Reestructurar el *software* para mejorar la legibilidad y su futuro mantenimiento
- Adicionar comentarios

Mejor práctica: *Para que un mantenimiento sea exitoso se debe comprender el software y los cambios que se realizarán, conocer la funcionalidad, el objetivo, la estructura interna y los requisitos.*

Elemento transversal – Riesgos: Si no se tiene en cuenta la mejor práctica anterior, se podrían introducir nuevos errores que conllevan a más gastos por mantenimientos adicionales.

Mejor práctica: *El técnico de mantenimiento de software debe replicar o verificar los problemas mediante la ejecución de pruebas adecuadas, entre las que se encuentra pruebas de regresión, para ahorrar tiempo y dinero.*

Aspectos para tener en cuenta en el mantenimiento de *software:*

Mantenibilidad: IEEE 14 764 define mantenibilidad como la capacidad del producto de *software* para ser modificado. Los cambios pueden incluir correcciones, mejoras o adaptación por variaciones en el entorno, así como en requisitos y especificaciones funcionales.

Como una característica de calidad de *software*, el mantenimiento debe especificarse, revisarse y controlarse durante las actividades de desarrollo, con el fin de reducir los costos. La presencia de metodologías, procesos maduros, técnicas y herramientas ayuda a mejorar la capacidad de mantenimiento de *software*.

Mejor práctica: *Las entidades deben definir desde el inicio del proyecto, quién es el responsable de realizar el mantenimiento del software, por lo general los equipos que desarrollan no son los que modifican.*

Procesos: El ciclo de vida del *software* es un conjunto de actividades, métodos, prácticas y transformaciones para desarrollar y mantener el *software* y sus productos asociados. En cuanto a procesos, las actividades de mantenimiento tienen mucho en común con el desarrollo, por ejemplo la gestión de la configuración de *software* es una actividad crucial en ambas fases.

Medidas específicas del mantenimiento de *software*: El modelo de la calidad del *software* sugiere medidas que son específicas para su mantenimiento y que incluyen:

- Capacidad de análisis: Se refiere a las medidas para calcular el esfuerzo o los recursos gastados para diagnosticar deficiencias o causas del fracaso, o para identificar las partes que deben ser modificadas.
- Estabilidad: Son las medidas del comportamiento inesperado de *software*, incluyen medición calculada durante las pruebas.
- Capacidad de prueba: Es la medida del técnico de mantenimiento y del esfuerzo de los usuarios al probar la modificación *software*.
- Otras medidas que el técnico de mantenimiento de *software* utilizan son: el tamaño, la complejidad, la comprensibilidad y *mantenibilidad*.

Actividades específicas del mantenimiento de *software*: Hay una serie de procesos, actividades y prácticas que son únicos para el mantenimiento del *software:*

- Comprensión del programa: incluye actividades necesarias para obtener un conocimiento general de lo que es un producto de *software*, qué hace y cómo sus partes trabajan juntas.
- Transición: es una secuencia de actividades controlada y coordinada durante la cual el *software* se transfiere progresivamente desde el desarrollador al técnico de mantenimiento.
- Modificación: solicitud de aceptación o rechazo de las modificaciones que se solicitan. Dependiendo del grado de capacidad, esfuerzo y complejidad pueden ser rechazadas por el técnico de mantenimiento y enviadas a un desarrollador.
- Análisis de impacto: es una técnica para identificar las áreas afectadas por un cambio potencial.
- Acuerdos de mantenimiento de nivel de servicio (SLA) y licencias de mantenimiento y contratos: son acuerdos contractuales que describen los servicios y los objetivos de calidad.
- Apoyo: se refiere a la documentación, configuración de la gestión, verificación, validación, resolución de problemas, aseguramiento de la calidad del *software*, revisiones y auditorías.

Elemento transversal – Talento TI: *El técnico en mantenimiento de software debe cumplir con las siguientes actividades:*

- *Realizar pruebas de calidad*
- *Planificar las actividades del mantenimiento*
- *Estimar los costos*
- *Identificar posibles conflictos y desarrollar alternativas*
- *Evaluar los riesgos*
- *Establecer cómo los usuarios solicitarán modificaciones o reportarán problemas*

- *Informar a todas las partes interesadas*

Gestión de la configuración de *software*: IEEE 14764 describe la gestión de la configuración del *software* como un elemento crítico del proceso de mantenimiento. Los procedimientos deben prever la verificación, validación y auditoría de cada paso requerido para identificar, autorizar, ejecutar y entregar el producto.

10.2.2 Técnicas para mantenimiento de *software*

Programa de comprensión

Los programadores gastan un tiempo considerable leyendo y entendiendo los programas, con el fin de poner en práctica los cambios. Los navegadores de código son herramientas clave para facilitar esa comprensión pues organizan y presentan el código fuente.

Reingeniería

Es el examen y alteración de *software* para reconstituirlo en una nueva versión; incluye la ejecución posterior de esa nueva versión. A menudo se realiza para reemplazar el antiguo *software* y no para mejorar la capacidad. *Refactoring* es una técnica de reingeniería que tiene como objetivo la reorganización de un programa sin cambiar su comportamiento, se trata de mejorar su estructura y su mantenibilidad y puede ser utilizada para las modificaciones de menor importancia.

Ingeniería inversa

Es el proceso de análisis de *software* para identificar sus componentes e interrelaciones, de tal manera que se puedan crear representaciones en otra forma o en mayores niveles de abstracción. La ingeniería inversa es pasiva, no cambia el *software* ni da lugar a uno nuevo.

Migración

El *software* puede ser modificado para funcionar en distintos entornos. El técnico de mantenimiento debe determinar las acciones necesarias para migrar el *software*, desarrollar y documentar. Los pasos que deben seguirse se llevan a cabo mediante un plan que cubre los requisitos de migración, las herramientas, la conversión del producto y los datos, la ejecución, la verificación y el apoyo.

Retiro

Una vez que el *software* ha alcanzado el final de su vida útil, debe ser retirado; sin embargo, se debe analizar esta decisión. Este análisis debe ser incluido en el plan de retiro, que cubre los requisitos, el impacto, la sustitución, el horario y el esfuerzo, lo que conlleva una serie de actividades similares a la migración.

10.2.3 Herramientas

Las herramientas son particularmente importantes en la etapa de mantenimiento del *software*, especialmente en los casos donde el *software* está siendo modificado. A continuación se listan y explican brevemente las herramientas de mantenimiento comúnmente utilizadas:

- Analizadores estáticos: permiten tener una visión general y resumir los contenidos del programa.
- Analizadores dinámicos: le permiten al técnico de mantenimiento trazar la ruta de ejecución de un programa.
- Analizadores de flujo de datos: le permiten al técnico de mantenimiento rastrear todos los posibles flujos de datos de un programa.
- Analizadores de dependencia: ayudan a los técnicos de mantenimiento a analizar y comprender las interrelaciones entre los componentes de un programa.

11 RECOMENDACIONES

Tras el diagnóstico y la formulación de alternativas para mejorar la probabilidad de éxito de los proyectos de desarrollo de software, se presenta un nuevo desafío: difundir el documento de resultados y garantizar la apropiación y uso de las buenas prácticas y herramientas identificadas para cada una de las etapas del ciclo de vida de desarrollo.

Identificar las mejores prácticas y herramientas fue un trabajo complejo. Sin embargo, cómo aprovechar esta oportunidad y garantizar el impacto potencial en las entidades son las mayores preocupaciones que trae la etapa de implementación del contenido de este documento. Es por esto que a continuación se proponen cinco objetivos principales para esta segunda fase:

- Dar a conocer a las entidades los resultados del diagnóstico hecho sobre la problemática en los procesos de contratación de desarrollo de sistemas de información.
- Presentar las herramientas y buenas prácticas que las entidades pueden implementar para mejorar la probabilidad de éxito de los proyectos de desarrollo de software.
- Asegurar la apropiación de los conceptos y criterios de selección necesarios para elegir e implementar las mejores prácticas y herramientas según el contexto particular de cada proyecto y entidad.
- Promover el uso del contenido de este documento en todo el Estado.
- Identificar las oportunidades de mejora y las herramientas que se entregan en este documento, con el fin de garantizar la evolución continua de la propuesta.

Para cumplir con el desarrollo de los cinco objetivos anteriormente planteados, se sugieren las siguientes estrategias:

- Crear un equipo de acompañamiento que dé a conocer los resultados del diagnóstico, que presente las buenas prácticas y herramientas, y que apoye a las entidades para acelerar la curva de aprendizaje y transformar los proyectos de desarrollo de software en lo público. El equipo además debe detectar las oportunidades de mejora y complementar o corregir la información del documento.
- Desarrollar una herramienta didáctica que permita que el usuario final utilice el conocimiento de forma sencilla y efectiva.
- Diseñar e implementar un esquema que evalúe la efectividad de la herramienta didáctica y que garantice que ésta provee un escenario sobre el cual los funcionarios practiquen los conceptos adquiridos.

12 REFERENCIAS

Schwalbe, K. (2014). *Information Technology Project Management* (7 ed.).

Inteco. (2009). Guía de mejores prácticas de calidad de producto

Lledo, Pablo. (2013). Director de proyectos.

PMBOK Guide – 5th edition

Swebok v3.0. (2013). Guide to the Software Engineering Body of Knowledge

Sharp, H., Galal, G., & Finkelstein, A. (1999). Stakeholder Identification in the Requirements Engineering Process. *Database and Expert Systems Applications*, 387 - 391. Obtenido de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.1495&rep=rep1&type=pdf

Wiegers, K., & Beatty, J. (2013). *Software Requierements - Developer best practices.* Microsoft Press.