GUÍA DOMINIO

MGGTI.G.SI - GESTIÓN DE SISTEMAS INFORMACIÓN

Ministerio de Tecnologías de la Información y las Comunicaciones 2023

MGGTI

Ministerio de Tecnologías de la Información y las Comunicaciones

Viceministerio de Transformación Digital

Dirección de Gobierno Digital

Subdirección de Estándares y Arquitectura de Tecnologías de la Información

Equipo de trabajo

Óscar Mauricio Lizcano Arango - Ministro de Tecnologías de la Información y las Comunicaciones

Sindey Carolina Bernal Villamarín - Viceministra de Transformación Digital

Ana María Sterling Bastidas – Directora de Gobierno Digital

Luis Clímaco Córdoba Gómez - Subdirector de Estándares y Arquitectura de TI

Jairo Alberto Riascos Muñoz – Equipo de la Subdirección de Estándares y Arquitectura de TI

Claudia Milena Rodríguez Alvarez- Equipo de la Subdirección de Estándares y Arquitectura de TI

Luis Martin Barrera Pino - Equipo de la Subdirección de Estándares y Arquitectura de TI

Samuel Antonio Peña Navarro - Equipo de la Subdirección de Estándares y Arquitectura de TI

Julio César Anaya Esteves - Equipo de la Subdirección de Estándares y Arquitectura de TI

Empresa consultora Yobiplex

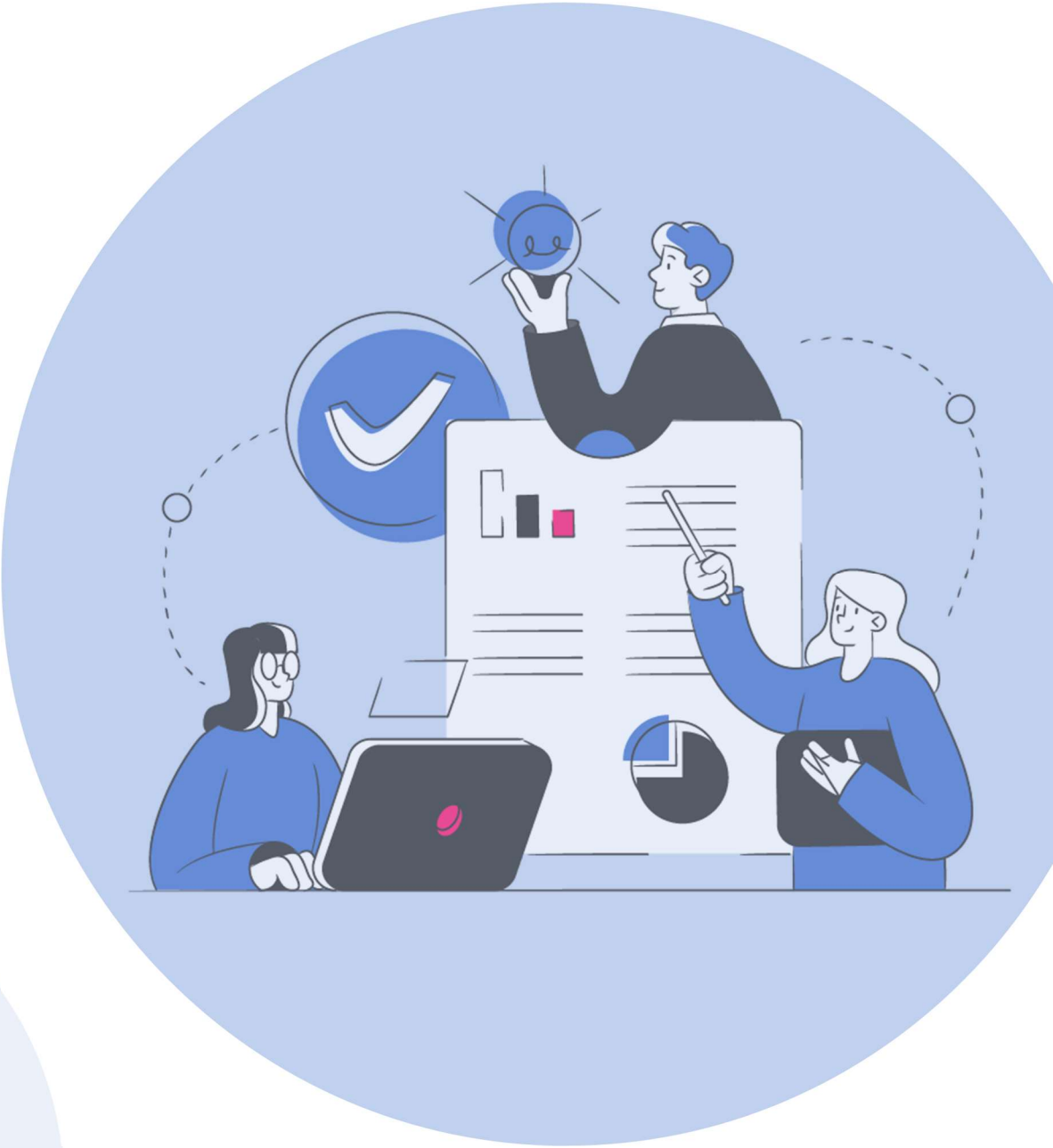
Versión	Observaciones
Versión 1.0 **Mayo 2023**	Guía Gestión de Sistemas de Información

Tabla de contenido

Listado de ilustraciones

Listado de tablas

1 . Introducción

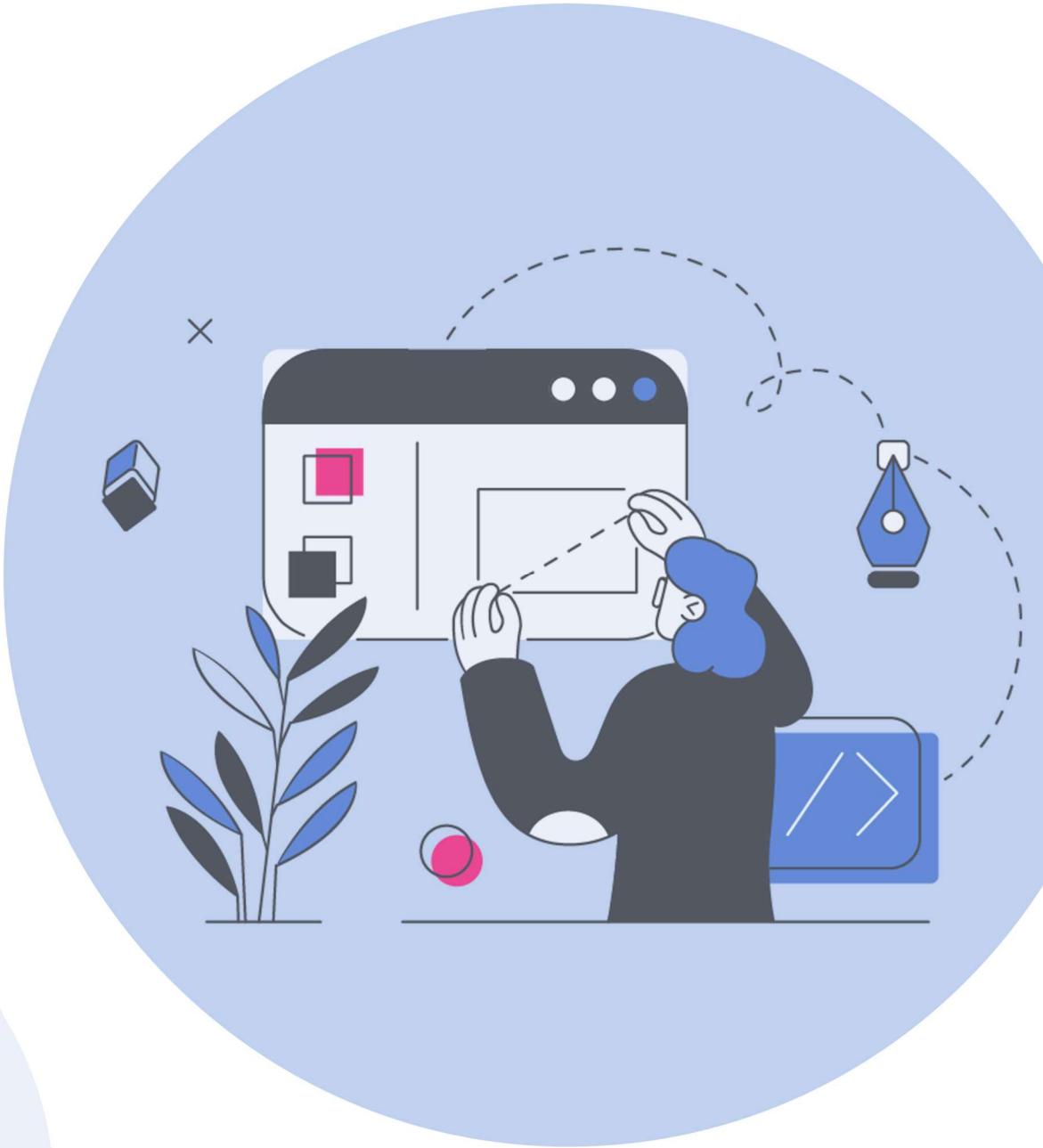


Para apoyar los procesos misionales y de apoyo en una entidad es importante contar con sistemas de información que, permitan: i)habilitar las transacciones de los procesos que generan la información; ii) garantizar la calidad de la información; iii) ser una fuente de datos útiles para la toma de decisiones corporativas; iv) disponer de recursos de consulta para los grupos de interés; y v) cumplir con los atributos de calidad que garanticen que son mantenibles, escalables, interoperables, seguros, funcionales y sostenibles financiera y técnicamente.

Este documento brinda los conceptos, las herramientas y los pasos, para que la entidad pueda aplicar los lineamientos del dominio de sistemas de información. La guía orienta desde la gestión de la arquitectura de sistemas de información, la gestión de los requerimientos de sistemas de información, el desarrollo y/o adquisición, pruebas hasta su implantación, garantizando la calidad durante todo el proceso.

(**Tenga en cuenta**) Las guías tienen un carácter orientador y no son de obligatoria aplicación (ARTÍCULO 2.2.9.1.2.2. Decreto 767 de 2022. Política de Gobierno digital). Esta guía busca orientar a las entidades públicas para aplicar los lineamientos del dominio de sistemas de Información.

2 . Lineamientos



Los lineamientos son orientaciones de carácter general y corresponden a disposiciones o directrices que deben ser ejecutadas en las entidades del Estado colombiano para implementar el Modelo de Gestión y Gobierno de TI. Los sistemas de información y aplicaciones deben soportar los procesos de las entidades públicas. Los lineamientos de este dominio permiten que la entidad diseñe, desarrolle, modifique o adquiera e implante los sistemas de información que soporten de forma adecuada los procesos y procedimientos de la entidad para ofrecer mejores trámites y servicios a los ciudadanos.

En la siguiente tabla se presentan los lineamientos que aplican al dominio de Gestión de Sistemas de Información.

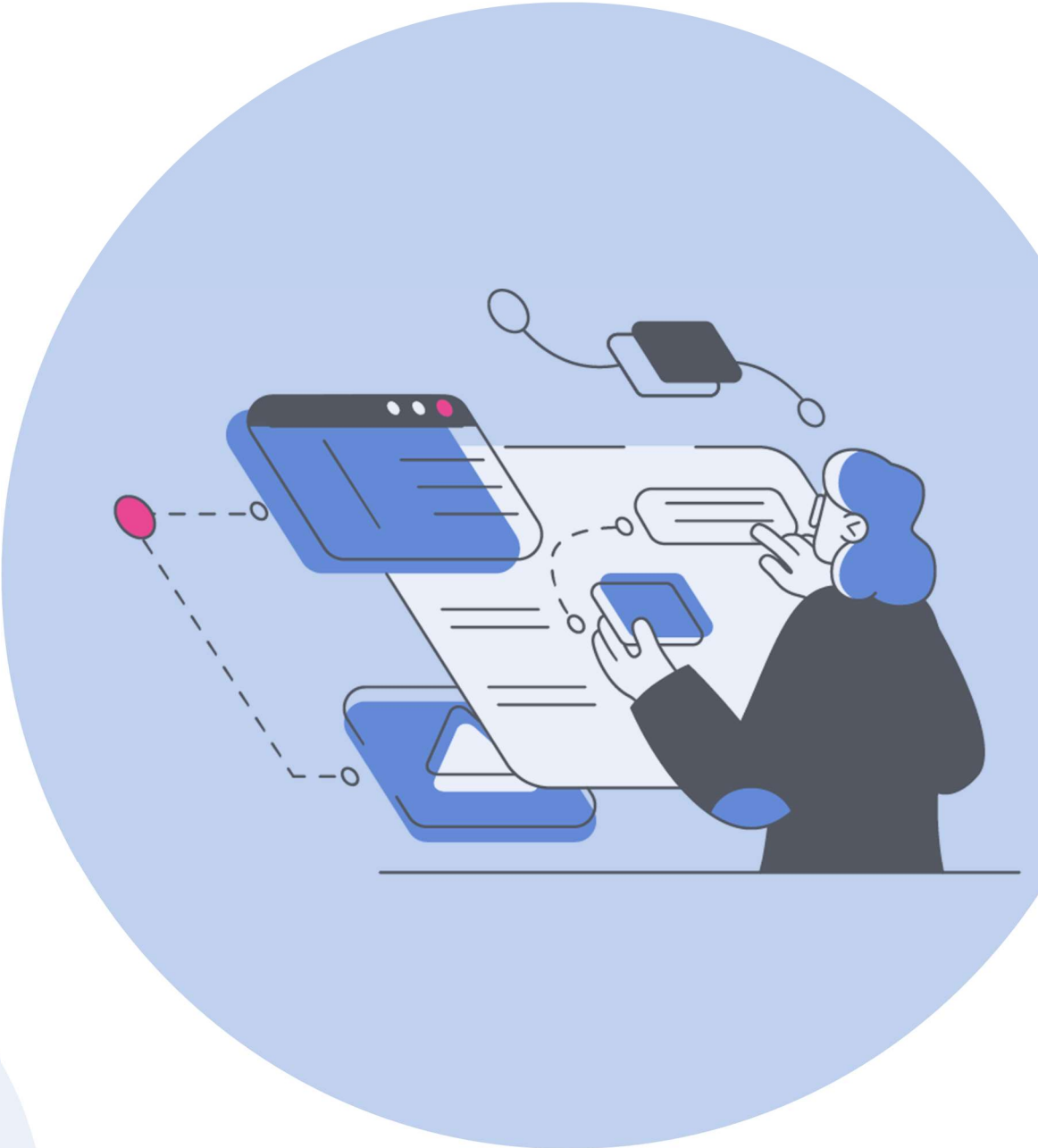
Código	Título	Descripción
MGGTI.LI.SI.01	Metodología para el desarrollo de sistemas de información	La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe adoptar y personalizar una metodología alineada a las mejores prácticas para el desarrollo y mantenimiento de software, que oriente los proyectos de construcción o evolución de los sistemas de información que se desarrollen internamente o a través de terceros.
MGGTI.LI.SI.02	Catálogo de Sistemas de Información	La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe garantizar la construcción y gestión del catálogo de sistemas de información, el cual integra la caracterización de cada uno de sus sistemas de información. Las entidades cabeza de sector adicionalmente deben consolidar y mantener actualizado el catálogo de sistemas de información sectorial.
MGGTI.LI.SI.03	Guía de estilo y usabilidad	La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe definir o adoptar una guía de estilo y usabilidad para la institución. Esta guía debe incorporar los lineamientos de gov.co y elementos de accesibilidad web.
MGGTI.LI.SI.04	Ambientes independientes en el ciclo de vida de los sistemas de información	La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe implementar y mantener ambientes independientes (desarrollo, pruebas, producción) durante el ciclo de vida de los sistemas de información.
MGGTI.LI.SI.05	Análisis de requerimientos de los sistemas de información	La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe incorporar dentro de sus procesos, las actividades formales de análisis y gestión de requerimientos de software en el ciclo de vida de los sistemas de información de manera que se garantice su trazabilidad y cumplimiento.
MGGTI.LI.SI.06	Integración, entrega y despliegue continuo durante el ciclo de vida de los sistemas de información	La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe implementar dentro del proceso de desarrollo de sistemas de información, estrategias de integración, entrega y despliegue continuo en las actividades de desarrollos de sistemas de información.

MGGTI.LI.SI.07	Plan de pruebas durante el ciclo de vida de los sistemas de información.	. La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe estructurar un plan de pruebas que cubra aspectos funcionales y no funcionales sobre los nuevos desarrollos y mantenimientos evolutivos de los sistemas e información. La aceptación de cada una de las etapas de este plan debe estar vinculada a la transición del sistema de información a través de los diferentes ambientes.
MGGTI.LI.SI.08	Manual del usuario, técnico y de operación de los sistemas de información	La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe asegurar que todos sus sistemas de información cuenten con la documentación técnica y funcional debidamente actualizada
MGGTI.LI.SI.09	Plan de mantenimiento de los sistemas de información	La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe elaborar un plan de mantenimiento anual de los sistemas de información de la Entidad.
MGGTI.LI.SI.10	Servicios de mantenimiento de sistemas de información con terceras partes	La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe establecer criterios de aceptación y definir Acuerdos de Nivel de Servicio (ANS) cuando se tenga contratado con terceros el mantenimiento de los sistemas de información. Los ANS se deben aplicar en las etapas del ciclo de vida de los sistemas de información que así lo requieran y se debe velar por la continuidad del servicio.
MGGTI.LI.SI.11	Plan de calidad de los sistemas de información	La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe implementar un plan de aseguramiento de la calidad que contemple con claridad criterios de aceptación que generen valor durante el ciclo de vida de los sistemas de información.
MGGTI.LI.SI.12	Requerimientos no funcionales o atributos calidad de los sistemas de Información	La Dirección de tecnologías y Sistemas de la Información o quien haga sus veces, para la construcción o evolución de los Sistemas de Información, debe identificar los requerimientos no funcionales aplicables asociados a los atributos de calidad, garantizando su cumplimiento una vez entre en operación el sistema.
MGGTI.LI.SI.13	Accesibilidad	La Dirección de tecnologías y Sistemas de la Información o quien haga sus veces, debe garantizar que los sistemas de información y portales web que estén disponibles para el acceso a la ciudadanía o aquellos que de acuerdo con la caracterización de usuarios lo requieran, deben cumplir con las características de accesibilidad web.
MGGTI.LI.SI.14	Arquitectura de software	

		La dirección de tecnologías y Sistemas de la Información o quien haga sus veces debe definir, documentar y mantener actualizadas las arquitecturas de software de cada sistema de información de la Entidad.

Tabla 1. Lineamientos del dominio de Gestión de Sistemas de Información

3 Etapas



En esta sección se describen aspectos relevantes que se deben considerar para la adecuada gestión de los Sistemas de Información, desde la gestión de la arquitectura, la gestión del ciclo de vida de los sistemas de información que incluye el ciclo de desarrollo de software y el aseguramiento de calidad.

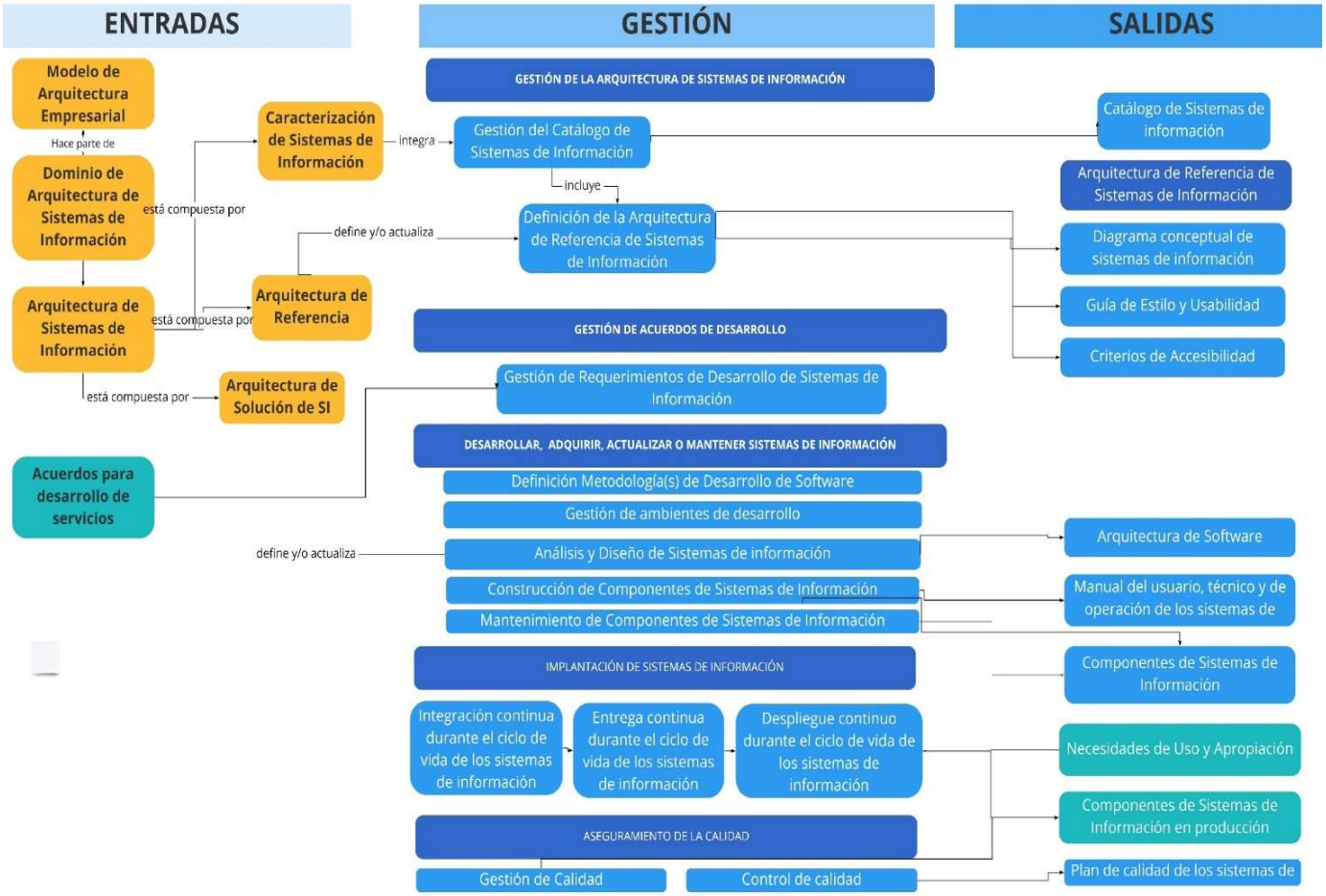


Ilustración 1. Modelo conceptual de Gestión de Sistemas de Información (Elaboración propia)

El dominio de gestión de Sistemas de Información recibe las necesidades de los procesos de negocio y genera los componentes de información que satisfacen estas necesidades. Este esquema incluye las relaciones del modelo con la Estrategia de TI y Gobierno TI, toda vez que los sistemas de información deben desarrollarse en el marco de la estrategia de TI definida y teniendo en cuenta los esquemas de gobernabilidad establecidos de TI en la entidad, la definición de estándares y los lineamientos para la gestión de los proyectos de desarrollo e implantación de sistemas de información.

La gestión de los sistemas de información se hace desde un enfoque arquitectural que, junto con la gestión de la arquitectura de TI de la entidad realizada de acuerdo con lo definido en la Estrategia de TI, permite una evolución coherente de la arquitectura de sistemas de información y una administración del catálogo de sistemas de información procurando generar el mayor valor a la entidad.

A partir de la identificación de necesidades de información y de sistematización que se requieren para apoyar la estrategia de la organización y sus procesos, se establecen y acuerdan los desarrollos de software para soportar procesos, trámites y servicios Estos acuerdos de desarrollo de servicios y sistemas de información se construyen siguiendo una metodología de desarrollo de software definida por la entidad y que incluye tanto los desarrollos con la capacidad interna o tercerizado a través de fábricas de software. Esta metodología debe garantizar las actividades necesarias hasta lograr la puesta en producción de los servicios y sistemas de información que, junto con las actividades de uso y apropiación de TI, se garantice que estos servicios y sistemas de información habilitan los procesos de negocio de la entidad. De manera transversal al proceso se debe asegurar la calidad tanto del proceso de desarrollo como del producto

3.1 Gestión de la Arquitectura de Sistemas de Información

La gestión de la arquitectura de sistemas de información implica mantener actualizada la documentación de las arquitecturas de solución de cada sistema, los manuales, lineamientos y estándares de construcción y diseño.

La gestión de Sistemas de Información inicia con la definición de la arquitectura de sistemas de información de manera tal que permita, entre otros:

- Identificar y caracterizar los sistemas de información que soportan los diferentes procesos y servicios de la entidad. Esta caracterización se documenta en el catálogo de Sistemas de Información.
- Comprender la manera en que estos Sistemas de Información habilitan los procesos y servicios de la entidad. Comprender los actores y grupos de interés con los que se intercambia información y los flujos de información.
- Observar cómo se comunican e interoperan los Sistemas de Información.
- Analizar las intervenciones que requieren los sistemas de información para tomar decisiones que garanticen que los sistemas de información evolucionan de manera alineada con la estrategia de la entidad y las necesidades de los procesos de la entidad.

De acuerdo con la complejidad y las necesidades de cada entidad, se pueden definir los modelos y artefactos requeridos. A continuación, se describen algunos elementos básicos que hacen parte de la arquitectura de sistemas de información.

Recuerde. Para documentar la Arquitectura de Sistemas de Información puede seguir la orientación definida en el Modelo de Arquitectura Empresarial.

3.1.1 Definición y actualización de la Arquitectura de Referencia de Sistemas de Información

La Arquitectura de Referencia es una arquitectura genérica y es una guía que define los criterios que orientan y deben ser considerados en todas las decisiones técnicas en materia de sistema de información de manera que se logre una implementación y evolución ordenada y homogénea de la tecnología en toda la entidad.

En cuanto a los Sistemas de Información, la Arquitectura de Referencia debe contemplar, entre otros, los siguientes aspectos:

- Principios
- Patrones de arquitectura.
- Decisiones arquitecturales de alto nivel
- Recomendaciones y mejores prácticas tecnológicas y de desarrollo.
- Herramientas específicas para el desarrollo.
- Componentes existentes reutilizables.
- Restricciones tecnológicas

La guía MAE.GE.ASI.01 Guía técnica de soluciones tecnológicas, describe las actividades para el diseño de Arquitecturas de Referencia y Arquitecturas de Solución.

Cada entidad de acuerdo con su contexto y realidad debe definir su arquitectura de referencia.

Tenga en cuenta: En este dominio se definen los aspectos de la Arquitectura de Referencia para Sistemas de Información, sin embargo, también se deben definir Arquitecturas de Referencia para infraestructura tecnológica, de seguridad digital e Información.

3.1.2 Diagrama de flujo de información de contexto

En este diagrama se representan las relaciones entre la entidad y los grupos de interés y actores involucrados con quienes se intercambia información.

Recuerde. En el contexto del Modelo Integrado Planeación y Gestión- MIPG se definen los lineamientos para caracterizar grupos de valor y grupos de interés, los cuales se definen así:

Grupos de Valor[1]: son las personas naturales (ciudadanos) o jurídicas (organizaciones públicas o privadas), grupos étnicos (afrocolombianos, indígenas y ROM) a quienes van dirigidos los bienes y servicios de una entidad. Es importante que la entidad tenga identificado a sus grupos de valor por rasgos como su ubicación, condiciones económicas, sociales, culturales, entre otros.

Grupos de Interés: son los individuos u organismos específicos que tienen un interés especial en la gestión y los resultados de la entidad, como gremios, asociaciones de usuarios (campesinos, agricultores, transportadores), sindicatos, universidades, entre otros.

Partes Interesadas[1]: persona u organización que puede afectar, verse afectada o percibirse como afectada por una decisión o actividad.

Para elaborar este diagrama puede usar la herramienta de modelamiento de Arquitectura empresarial que posee la entidad. Los diagramas de flujo de la información se aplican a los datos que fluyen desde la entrada hasta la salida; ya que adopta un punto de vista del tipo entrada-proceso/sistema-salida. Como se muestra en la Ilustración 2. Diagrama de Flujo de Información de contexto, en donde el proceso o sistema en este caso representa a la entidad, los actores se agrupan en los grupos de interés y se representan los flujos de entrada y de salida con las flechas en la dirección que circula la información.

Tip. El diagrama de flujo de datos DFD, puede elaborarse en varios niveles que representan un mayor nivel de detalle. En este caso se usa el nivel 0 o modelo de contexto en donde se representa la entidad como el proceso central y los grupos de interés y actores externos que entregan o reciben información. También podría elaborar los siguientes niveles especificando los procesos, servicios e inclusive los sistemas de información a través de los cuáles fluye la información.

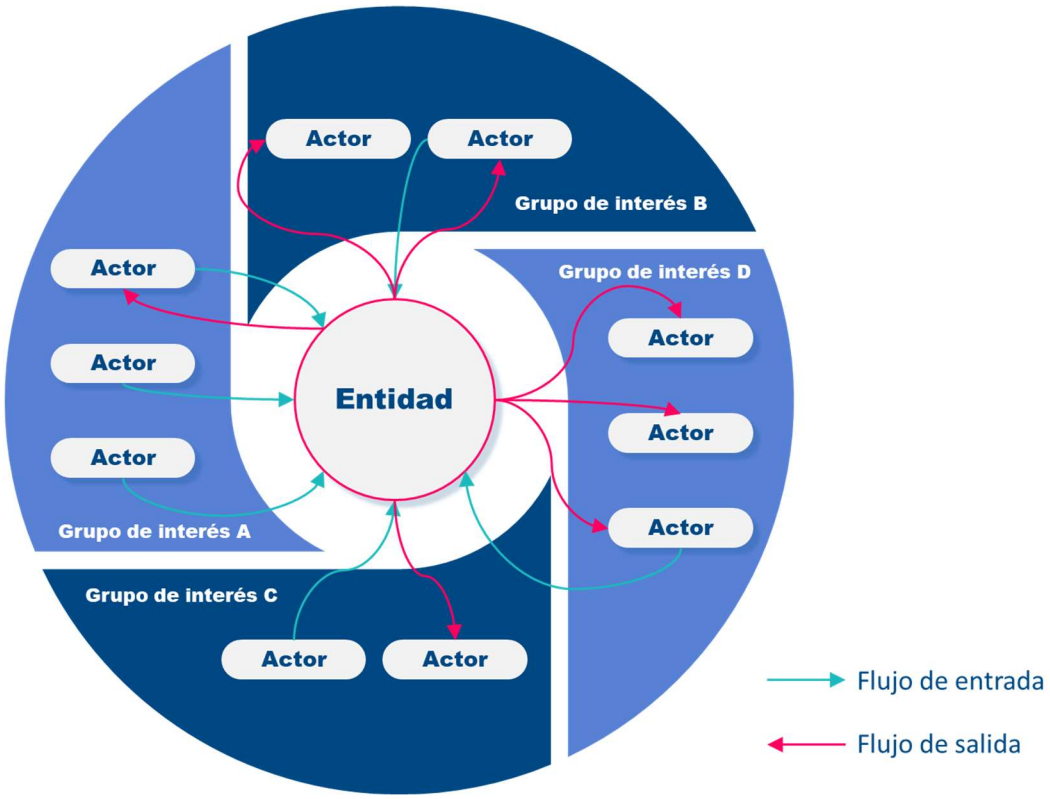


Ilustración 2. Diagrama de Flujo de Información de contexto (elaboración propia)

El diagrama de flujo de información de contexto puede complementarse con la tabla 2. Flujos de Información en la que se específica la información que se intercambia en cada uno de estos flujos.

Grupo de Interés	Actor	Información que entrega	Información que recibe
Grupo de Interés A	Actor A1.	Información Ax1 Información Ax2	Información Ay
Grupo de Interés B	Actor B1	Información Bx1	Información By1
	Actor B2		Información By2
Grupo de Interés C	Actor C1	Información Cx1 Información Cx2 Información Cx3.	

Tabla 2. Flujos de información

3.1.3 Apoyo de los Sistemas de Información a los procesos

Es necesario determinar cómo se relacionan los sistemas de información disponibles en la entidad con los procesos de negocio que respaldan, para lo cual se pueden clasificar de acuerdo con su carácter: misional, apoyo, direccionamiento y servicios de información, de tal manera que se garantice el flujo de información para la gestión, control y toma de decisiones como se muestra en Ilustración 3. Estructura General de los Sistemas de Información.

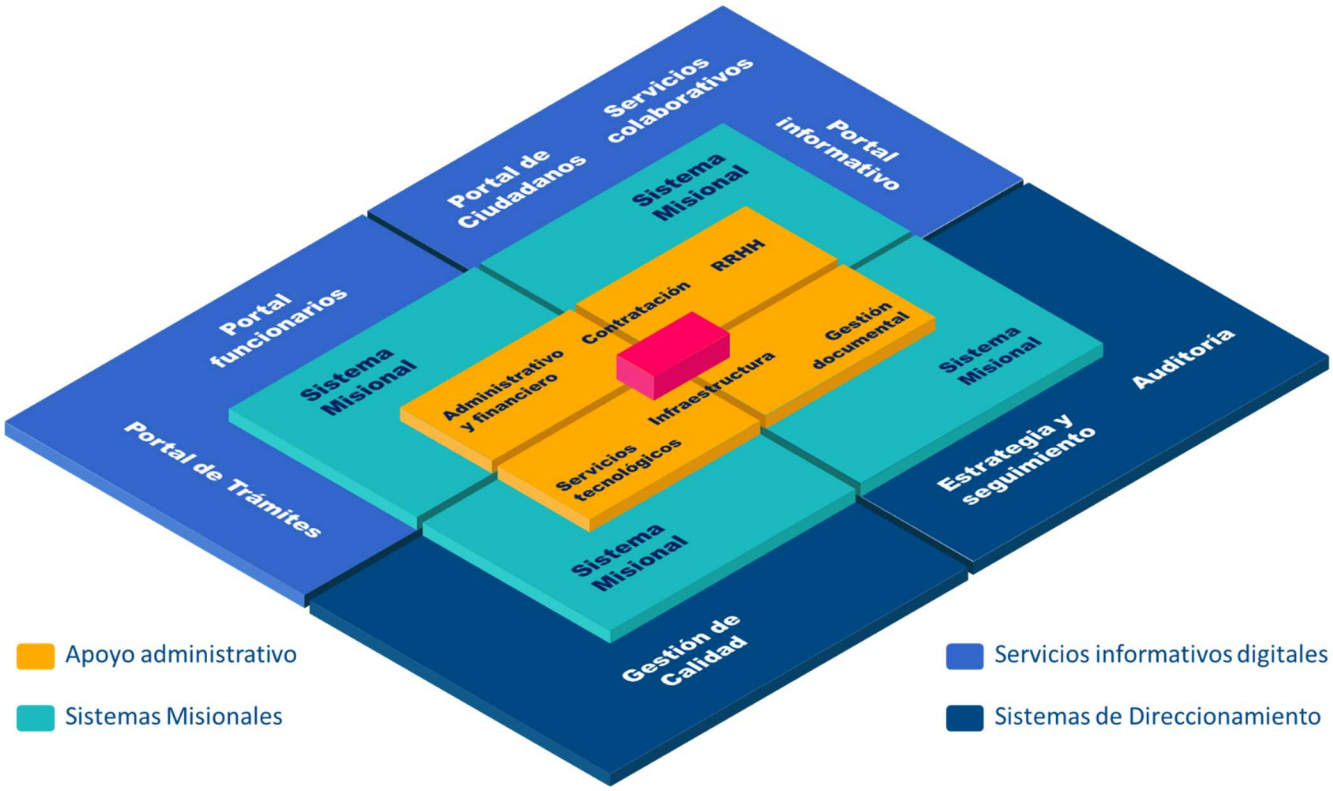


Ilustración 3. Estructura General de los Sistemas de Información (elaboración propia)

En un primer nivel de la estructura, se agrupan los sistemas de información de apoyo administrativo que constituyen el backoffice de la entidad y usualmente contienen facilidades de ERP (presupuesto, contabilidad, tesorería, caja, bancos, inventarios, activos fijos, entre otros), administración de recursos humanos, gestión de infraestructura y gestión de tecnología. En este nivel se realizan las tareas operativas y repetitivas de tipo administrativo.

El segundo nivel es el de los sistemas misionales, los cuales apoyan directamente la misión de la entidad. En el tercer nivel están los sistemas de servicios informativos digitales y los sistemas de direccionamiento estratégico. Los servicios informativos digitales son todas aquellas herramientas que les permiten a los diferentes actores del sistema de información interactuar entre sí y con la información de los sistemas misionales y los de apoyo administrativo, desde una perspectiva de servicio y en un modelo organizado de Portales de información y herramientas colaborativas. Los sistemas de direccionamiento estratégico, por otra parte, son las facilidades que se le disponen a las instancias directivas y de decisión para hacer seguimiento oportuno a la ejecución de la estrategia definida, proporcionando información sobre el avance en el alcance de las metas e información para la toma de decisiones estratégicas.

La arquitectura de sistemas de información además implica que todos sus niveles y las piezas que componen cada nivel estén lógica y adecuadamente interconectadas para permitir el flujo de información definido por los procesos de la organización.

Adicionalmente, propicia que el sistema de información cumpla con las principales premisas que hacen posible el análisis de la información: fuentes únicas de datos, información de calidad, información como servicio, información en tiempo real si se requiere y la información como un bien público.

Dentro de cada nivel de la arquitectura se agrupan los sistemas o subsistemas de acuerdo con la categoría de información que soportan. Un sistema de información a su vez se compone de varios subsistemas o módulos con propósitos específicos.

La Ilustración 3. Estructura General de los Sistemas de Información, se puede complementar con la tabla 3. Matriz de Proceso vs Sistemas de Información, puede observar en una matriz de Procesos de Negocio y Sistemas.

Sistemas de Información / Procesos de Negocio	Sistema de Información 1	Sistema de Información 2	Sistema de Información 3	Sistema de Información 4	Sistema de Información 5
Proceso 1	X				
Proceso 2		X			
Proceso 3			X	X	
Proceso 4				X	

Tabla 3. Matriz de Proceso vs Sistemas de Información

El propósito de esta matriz es representar la relación que existe entre los Sistemas de Información y los procesos de negocio dentro de la entidad. Mapear y analizar esto es un paso importante, ya que permite:

- Asignar a cada proceso de negocio los sistemas de información que usa para su ejecución o identificar si existe algún proceso de negocio quejo este soportado por algún sistema o aplicación de software
- Observar y comprender las necesidades de los procesos de negocio con respecto al soporte de sistemas de información, esto incluye el nivel de digitalización y automatización del proceso.

Dado que los Sistemas de Información, frecuentemente interactúan entre sí para la ejecución de los procesos de negocio, esta interacción implica que crean, leen, actualizan y eliminan datos dentro de otros sistemas de información; lo cual se logrará mediante algún tipo de interfaz, ya sea a través del intercambio de un archivo por lotes que se carga periódicamente, a través de una conexión directa a la base de datos de otro sistema de información o mediante algún tipo de API o servicio web.

Por lo anterior, puede ser necesario identificar y analizar los puntos y mecanismos de integración e interoperabilidad entre los Sistemas de Información; esto se puede observar a través de una matriz de integración de sistemas de información contra sistemas de información comparando interfaz-protocolo como se muestra en la tabla 4. Matriz de integración.

Sistemas de Información	Sistema de Información 1	Sistema de Información 2	Sistema de Información 3	Sistema de Información 4	Sistema de Información 5

Sistema de Información 1	Web service SOAP				Data Base Connection
Sistema de Información 2		Web Service REST			
Sistema de Información 3			DBLink	Archivo con FTP	
Sistema de Información 4				Archivo con SFTP	
Sistema de Información 5					

Tabla 4. Matriz de Integración o interoperabilidad de Sistemas de Información

El propósito de esta matriz es representar los mecanismos de integración e interoperabilidad que existe entre varios sistemas de información. Mapear y analizar esto es un paso importante, ya que permite, entre otros beneficios:

- Comprender el grado de interacción entre sistemas de información, identificando aquellos que son centrales en términos de su dependencia de otros.
- Comprender el número y los tipos de interfaces entre sistemas de información.
- Comprender el grado de duplicación de interfaces entre sistemas de información.
- Identificar las posibilidades de simplificación de interfaces, así como la posibilidad de determinar si faltan integraciones y es necesario crearlas.

Tenga en cuenta: En esta matriz es importante observar integraciones con sistemas de información de otras entidades.

Tip. Otros artefactos que pueden facilitar el análisis de la integración de los Sistemas de Información son: Diagramas de Comunicación de Sistemas de Información y Diagrama de Conectividad de Nodos más conocido como modelo N2.

3.1.4 Gestión del Catálogo de Sistemas de Información

Para facilitar la gestión de los sistemas de información, es determinante construir, mantener y gestionar el catálogo de sistemas de información que es un inventario con la caracterización de todos los sistemas con los que cuenta la entidad. Este catálogo lo constituyen las caracterizaciones de cada sistema de información.

Recuerde que en la caja de herramientas de la política de Gobierno Digital, cuenta con diversos productos tipo (plantillas) , entre otros, el catálogo de sistemas de información el cual está compuesto por cada una de las fichas de caracterización de los sistemas de información.

Dentro de los aspectos más relevantes que se deben caracterizar están los siguientes:

- **Generales:** permiten identificar el sistema de información y entender el contexto de su ejecución, algunos atributos relevantes en esta sección son los siguientes:
    - Nombre del sistema: nombre según la categoría de información.
    - Flujos de información: relaciona las entradas y salidas de información
    - Objetivo: Propósito general para el que se concibe el sistema. Proceso o estrategia de la organización que soporta.
    - Categoría y subcategoría: de acuerdo con la clasificación definida por la entidad, ejemplo: Sistemas Misionales, Sistemas de apoyo administrativo y financiero, Sistemas de Direccionamiento.
    - Módulos o subsistemas que lo conforman: determinar posibles componentes del sistema de información de acuerdo con sus funcionalidades.
    - Plataforma: Plataforma tecnológico sobre la cual se encuentra desarrollado el software del sistema a nivel de aplicaciones y persistencia de datos.
    - Tipo de lenguajes

    Tipos de arquitecturas
    - Fabricantes (concentración de fabricantes) –
    - Donde está desplegado (on premises, nube, híbrido)
        - Fecha de puesta en producción
        - Fecha de última actualización
        - Fecha de vencimiento del soporte

- **Valor para la entidad**: están orientados en determinar la relación que tiene con los procesos, servicios e inclusive objetivos de la entidad, algunos atributos relevantes en esta sección son los siguientes:
    - Objetivos de negocio con que está alineado.
    - Procesos de negocio que soporta
    - Cantidad de usuarios
    - Nivel de cobertura de los procesos de negocio que soporta.
    - Líder funcional
    - Evolución: Cuál es la evolución que la organización tiene prevista para el sistema en el mediano y largo plazo.
    - Recomendaciones: Acciones de mejora propuestas para superar las debilidades en el corto plazo.

- Iniciativas: Planteamiento de nuevas funcionalidades o nuevos proyectos para atender necesidades no cubiertas o evolucionar el sistema.

- **Calidad:** están orientados a verificar el alineamiento y cumplimiento del sistema de información con los estándares y lineamientos definidos en la arquitectura de referencia, así como el cumplimiento de los atributos de calidad esperados en su operación.
  - Documentación técnica
  - Documentación funcional
  - Fortalezas: Identificación de fortalezas con los usuarios líderes y finales del sistema
  - Debilidades: Identificación de debilidades con los usuarios líderes y finales del sistema.
  - Amenazas: Identificación de amenazas con los usuarios líderes y finales del sistema, además de considerar posibles amenazas desde el punto de vista técnico como obsolescencia tecnológica de la plataforma, finalización de soporte, problemas de compatibilidad tecnológica, entre otros.
  - Atributos de calidad
  - Interfaces de integración con otros sistemas
  - Nivel de cumplimiento de estándar de la arquitectura de referencia
  - Eficiencia técnica
    - Cantidad de incidentes y problemas reportados
    - Riesgos tecnológicos
    - Nivel de Satisfacción de los usuarios
    - Monitoreo de rendimiento

- **Económicos:** determinar los costos de operación, adquisición, licencias, mantenimientos e infraestructura que permita realizar análisis de costo y beneficio, se deben considerar los costos asociados con:
  - Adquisición o desarrollo
  - Licenciamiento de software base
  - Operación
  - Mantenimiento evolutivo
  - Mantenimiento correctivo
  - Mantenimiento hardware

De acuerdo con la primera ley de evolución del software propuesta por Meir M. Lehman, “Un programa que se usa en un entorno real necesariamente debe cambiar o se volverá progresivamente menos útil y menos satisfactorio para el usuario”, por esta razón es importante definir una estratega de mantenimiento, la cual se puede fundamentar en el análisis de la información detallada en la caracterización de cada uno de los sistemas de información, dando respuesta a preguntas como las siguientes:

- ¿Cuáles son los sistemas de información que respaldan el cumplimiento de los objetivos estratégicos de la entidad?
- ¿Cuáles sistemas de información soportan los procesos de negocio?
  - ¿Estos sistemas de información satisfacen las necesidades del proceso? ¿Si/No? ¿En qué porcentaje?
  - ¿Estos sistemas de información satisfacen las necesidades de los usuarios?
  - ¿Hay sistemas de información que ofrecen funcionalidades redundantes?
- ¿Hay sistemas de información que tienen riesgos de obsolescencia tecnológica?

- ¿Los sistemas de información satisfacen las necesidades de los procesos y los usuarios?
- ¿Qué sistemas de información presentan la mayor cantidad de incidentes y problemas?
- ¿Hay sistemas de información que no cumplen los lineamientos de la arquitectura de referencia?
- ¿Hay sistemas de información que no soportan ningún proceso o servicio de negocio?
- ¿Cómo es la relación de los costos con respecto al valor que generan los sistemas de información?
- ¿Hay procesos de negocio que no están siendo soportados por sistemas de información? ¿Cuál es la brecha?
- ¿Hay Sistemas de información que requieran fortalecer la infraestructura tecnológica que la soporta?
- ¿Hay Sistemas de información que requieran fortalecer las acciones de uso y apropiación?
- ¿Hay Sistemas de información que requieran mantenimiento correctivos o evolutivos para reducir la cantidad de incidentes y problemas que presenta?

Evaluando las preguntas anteriores, puede clasificar los sistemas de información a través de los cuadrantes TIME (Tolerar – Invertir – Migrar – Eliminar), en el cual se evalúan tres criterios fundamentales: eficiencia técnica, valor al negocio y costos y riegos, como se ilustra en Ilustración 4. Cuadrante TIME, a continuación, se describe cada cuadrante.

- **Tolerar:** en este cuadrante se clasifican los sistemas de información que satisfacen las necesidades de la entidad con un nivel de calidad y riesgo aceptable y a un costo rentable.
- **Invertir o Innovar:** en este cuadrante se clasifican los sistemas de información que ofrecen un alto valor al negocio, tienen características altas en eficiencia técnica y un nivel de costo y riesgo aceptable.
- **Migrar:** en este cuadrante se clasifican aquellos sistemas de información que están en decadencia en su uso o no son compatibles técnicamente, presentando un bajo nivel de eficiencia técnica y altos costos y/o riesgos y requieren ser migrados por el valor requerido para el negocio.
- **Eliminar:** en este cuadrante se clasifican aquellos sistemas de información que ya no aportan un valor estratégico al negocio, empezando por aquellas con menores niveles de eficiencia técnica y altos niveles en costos y riesgos.

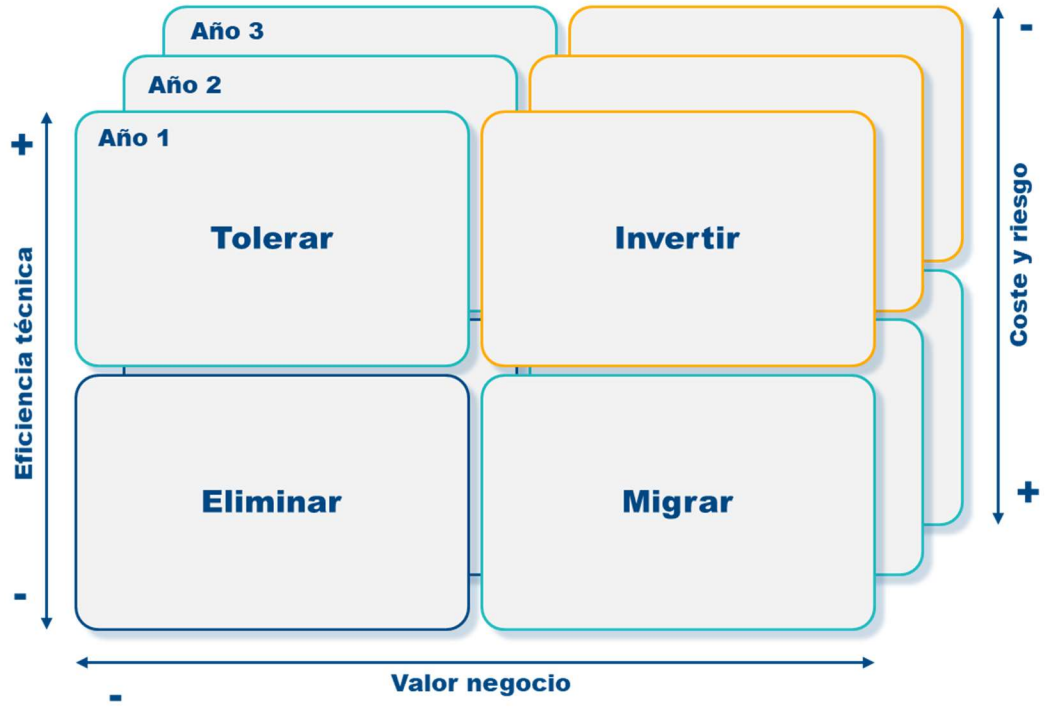


Ilustración 4. Cuadrante TIME

A partir de este análisis se plantean las posibles acciones a corto, mediano y largo plazo que conforman la estrategia de mantenimiento de los sistemas de información. La ejecución de las acciones de mantenimiento de sistemas de información hace parte de las actividades que se abordan en la sección de Mantenimiento de Sistemas de Información.

Meir M. Lehman, formuló las siguientes leyes de evolución de software, conocidas también como leyes de Lehman:

1. **Cambio continuo**: Un programa que se usa en un entorno real necesariamente debe cambiar o se volverá progresivamente menos útil y menos satisfactorio para el usuario.
2. **Complejidad creciente**: A medida que un programa en evolución cambia, su estructura tiende a ser cada vez más compleja. Se deben dedicar recursos extras para preservar y simplificar su estructura.
3. **Autorregulación**: La evolución de los programas es un proceso autorregulado. Los atributos de los sistemas, tales como tamaño, tiempo entre entregas y la cantidad de errores documentados son aproximadamente invariantes para cada entrega del sistema.
4. **Estabilidad organizacional**: Durante el tiempo de vida de un programa, su velocidad de desarrollo es aproximadamente constante e independiente de los recursos dedicados al desarrollo del sistema.
5. **Conservación de la familiaridad**: A medida que un sistema evoluciona también todo lo que está asociado a ello, como los desarrolladores, y usuarios, por ejemplo, deben mantener un conocimiento total de su contenido y su comportamiento para lograr una evolución satisfactoria. Un crecimiento exagerado disminuye esta capacidad. Por tanto, este incremento promedio debe mantenerse.
6. **Crecimiento continuo**: La funcionalidad ofrecida por los sistemas tiene que crecer continuamente para mantener la satisfacción de los usuarios.
7. **Decremento de la calidad**: La calidad de los sistemas de software comenzará a disminuir a menos que dichos sistemas se adapten a los cambios de su entorno de funcionamiento.
8. **Retroalimentación del sistema**: Los procesos de evolución incorporan sistemas de retroalimentación multi-agente y multi-ciclo y estos deben ser tratados como sistemas de retroalimentación para lograr una mejora significativa del producto.

3.2 Gestión del ciclo de vida de los Sistemas de Información

El ciclo de vida de los Sistemas de Información va desde la identificación de la necesidad, el desarrollo o adquisición y mantenimiento, la implantación hasta su retiro. Siendo el desarrollo un subconjunto del ciclo de vida, específicamente el ciclo de vida de desarrollo de software. En todo el ciclo de vida de los Sistemas de Información, se interactúa con otros dominios del Modelo de Gestión y Gobierno de TI e inclusive con el Modelos de Arquitectura Empresarial y el Modelo de Gestión de Proyectos. En esta sección se van a desarrollar los

principales aspectos que son específicos del dominio de Gestión de Sistemas de Información, mencionando los puntos de interacción con otros dominios y modelos.

3.2.1 Definición de la metodología de Desarrollo de Sistemas de Información

Para asegurar los niveles de calidad requeridos tanto técnica como funcionalmente, es necesario definir la metodología que guíe el proceso de desarrollo de software, es decir que guiarán el ciclo de desarrollo desde la fase de análisis de requerimientos y diseños funcionales, la construcción del software e incluye 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.

Para cumplir con su propósito una metodología de desarrollo de software debe cumplir con unas características fundamentales:

- ✓ Debe ajustarse a los objetivos y contexto del proyecto a realizar. Deben considerarse factores como: la complejidad del sistema y la volatilidad de los requerimientos.

- ✓ Debe cubrir todas las fases del ciclo del proceso de desarrollo de software ya que cada etapa tiene un objetivo específico que se debe satisfacer para garantizar que el producto de software obtenido es un producto de calidad.

- ✓ Debe además facilitar la integración de las distintas fases del ciclo de desarrollo permitiendo.

    - **Rastreabilidad**: es decir que es posible moverse no solo hacia adelante en el ciclo de vida, sino hacia atrás de forma que se pueda comprobar el trabajo realizado y se puedan efectuar correcciones.

    - **Fácil interacción entre etapas del ciclo de desarrollo**: es necesaria una validación formal de cada fase antes de pasar a la siguiente. La información que se pierde en una fase determinada queda perdida para siempre, con un impacto en el sistema resultante.

- ✓ Debe facilitar el entendimiento del problema a desarrollar por parte de todos los interesados, de manera que la metodología de alguna manera es también una herramienta que facilita la comunicación entre todos los involucrados. Esto significa que las herramientas de especificación y análisis debe ser amigables y sencillas.

- ✓ La metodología debe ser la base de una comunicación efectiva, no solo entre el analista y el usuario sino en todos los implicados en el proyecto, es decir, no sólo en la determinación de los requisitos sino en la comunicación de los avances y decisiones tomadas en la ejecución del proyecto.

- ✓ La clave del éxito es que todas las partes implicadas puedan intercambiar información libremente.

- ✓ La metodología debe poder emplearse en un entorno amplio de proyectos de software, es decir que:
    - La entidad deberá adoptar una metodología que sea útil para un gran número de sistemas que vaya a construir. Por esta razón no es práctico adoptar varias metodologías. Sin embargo, si es recomendable definir una metodología propia que tome las ventajas de otras metodologías, siempre y cuando se cumplan las características que se están describiendo.

- Las metodologías deberán ser capaces de abordar sistemas de distintos tamaños y rangos de vida.

- La metodología debe servir para sistemas de distinta complejidad.

- Entorno, la metodología debe servir con independencia de la tecnología disponible en la entidad.

La metodología de software a seleccionar o escoger puede ser de alguno de los siguientes enfoques:

1. Modelos de procesos predictivos o tradicionales: enfatizan la definición, la identificación y la aplicación detallada de las actividades y tareas del proceso desde el inicio del proyecto. En este enfoque se debe especificar muy bien los requerimientos y el camino a seguir antes de comenzar la implementación. Si estas metodologías no se usan con un criterio que defina que adaptaciones son necesarias, puede resultar en un proceso demasiado burocrático que genera complejidades innecesarias.

2. Modelos de procesos ágiles: su énfasis está en la maniobrabilidad y la adaptabilidad, son mucho más flexibles que los modelos tradicionales por lo que se adaptan con más facilidad a las condiciones cambiantes. Estas metodologías se emplean para generar victorias tempranas, son iterativas e incrementales

Los beneficios de contar con una metodología que oriente todo el proceso del ciclo de desarrollo del software son[1]:

- ✓ Consistencia y repetibilidad de los resultados a lo largo de los diferentes proyectos de la entidad. Las tareas definidas en un proceso son consistentes a lo largo de los diversos proyectos independientemente de quién las realice, ayudando a que los resultados sean predecibles y repetibles.
- ✓ Mayor eficacia y eficiencia del personal. La entidad puede comunicar de forma mucho más efectiva los procesos a sus colaboradores, y estos pueden comprender de forma clara sus responsabilidades en las tareas que ejecutan y los resultados y objetivos de las mismas.
- ✓ Mayor eficiencia organizacional. La organización puede diferenciar las tareas rutinarias de las tareas creativas, permitiendo abordar las tareas rutinarias de forma más precisa y eficiente. Esto a su vez incide en que se pueda invertir más tiempo en la realización de las tareas creativas.
- ✓ Facilitación de la gestión de los procesos. La definición de los procesos es el primer paso en la gestión de los mismos. En los procesos definidos por una organización se pueden establecer mediciones para controlar el desempeño y definir mejoras.

A continuación, se mencionan aspectos fundamentales en cada una de las fases del ciclo de desarrollo de software.

3.2.2 Fase de definición y gestión de Acuerdos de Desarrollo

[1] Fernández Sánchez, C. M. (2012). Modelo para el gobierno de las TIC basado en las normas ISO. Madrid, Spain: AENOR - Asociación Española de Normalización y Certificación.

En el dominio de Gobierno de TI, se definen acuerdos para el desarrollo de servicios, los cuales incluyen a los sistemas de Información, en donde se debe definir y ejecutar las prácticas adecuadas para gestionar la definición, análisis y gestión de requerimientos de manera que se garantice su trazabilidad y cumplimiento.

La dirección de Tecnologías y Sistemas de la Información o quien haga sus veces debe incorporar dentro de sus procesos actividades formales de análisis y gestión de requerimientos de software en el ciclo de vida de los sistemas de información de manera que se garantice su trazabilidad y cumplimiento, de acuerdo con la metodología de desarrollo definida y empleada.

Tenga en cuenta. Los acuerdos de desarrollo de sistemas de información, que se traducen en requerimientos pueden darse con diferentes niveles de detalle en diferentes momentos del ciclo de vida aplicando los conceptos de abstracción y refinamiento según sea necesario.

Abstracción: Define el nivel de detalle con el que se describe un sistema o alguno de sus componentes. En el nivel más alto se enuncia una solución en términos gruesos con el uso del lenguaje del ambiente o contexto del problema o necesidad. En niveles más bajos de abstracción se da la descripción más detallada de la solución de modo que pueda implementarse.

Refinamiento: es el proceso que complementario al de abstracción. Mientras que la abstracción permite ocultar los detalles de bajo nivel. El refinamiento ayuda a revelar estos detalles a medida que se avanza en el diseño.

Estas prácticas de análisis y gestión de requerimientos proporcionan el mecanismo apropiado para entender y analizar las necesidades de los procesos, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida que se transforman en un sistema funcional.

"Fallamos más a menudo porque resolvemos el problema incorrecto, que porque realizamos una mala solución del problema correcto" Ackoff

Importancia de los requerimientos

Uno de los factores claves de éxito en el desarrollo de un proyecto que busque satisfacer necesidades a través de la construcción o modificación del *software*, es entender con claridad el contexto del negocio en el que se implementará el *software* y los requerimientos que debe satisfacer y, durante todo el desarrollo del proyecto

verificar que todas las acciones y decisiones que se tomen estén orientadas a satisfacer estos requerimientos, los cuales pueden variar en el tiempo.

Los requerimientos son las capacidades y prestaciones que debe tener un sistema para la solución de un problema. Deben incluir todos los aspectos necesarios para satisfacer las necesidades de los usuarios. Para evaluar la calidad del *software* se verifica si este satisface o no los requerimientos del usuario. A continuación, se explica la importancia de los requerimientos:

- ✓ **Para que los usuarios digan y obtengan lo que quieren:** La adecuada definición de requerimientos representa la primera herramienta para que la comunicación entre usuarios y desarrolladores ocurra en forma adecuada. Un proceso correcto en este sentido debe facilitar el proceso para las dos partes, asegurando que los usuarios tengan cómo expresar sus necesidades de una manera útil, facilitando el registro de esas necesidades y ofreciendo mecanismos para integrar sus solicitudes en el proceso de construcción de la solución de software que les será entregada.

- ✓ **Para verificar el diseño:** Una definición adecuada de requerimientos es un plano, mapa o lista de objetivos contra los cuales es posible y de hecho necesario, verificar el diseño del software. Sólo contando con una lista actualizada, completa, clara y real de lo que el cliente necesita, es posible verificar si el diseño propuesto logrará satisfacer al usuario final. Este proceso de verificación debe además hacerse antes de comenzar con la codificación, de tal manera que los cambios que sean necesarios se hagan antes de que se incurran en más gastos y pérdidas de tiempo.

- ✓ **Para medir el progreso:** La definición de requerimientos, junto con la priorización dada a los mismos, son herramientas para verificar de manera rápida el progreso realizado a lo largo del proyecto. De nuevo, tratándose de una lista completa y real de lo que se debe cumplir cuando se termine el software, es supremamente útil como herramienta de contraste para el avance de la implementación.

    - ✓ **Para entregar y aceptar el producto según criterios precisos:** Una lista o definición de requerimientos bien hecha se convierte también en una especie de lista de chequeo mediante la cual se puede constatar que el software entregado corresponde con lo requerido por el cliente o grupo de interes. En este sentido, un proceso consensual de definición de requerimientos generará uno o varios documentos que podrán ser utilizados en la entrega final al cliente para que este verifique que todo lo que solicitó esté incluido en el producto que se le está entregando. De la misma manera, para el desarrollador es útil tener esa lista, para poder contestar a cualquier solicitud por parte del cliente de una funcionalidad que no fue definida desde el principio del proyecto.
    - ✓ **Como herramienta orientadora de los procesos de pruebas:** Si se ha definido de una manera completa y clara la lista de funcionalidades que debe tener la aplicación y los comportamientos que se esperan, los procesos de pruebas cuentan con una base firme para definir los casos de prueba y los resultados idóneos a buscar. Esta es una herramienta invaluable que ahorra tiempo y trabajo al mismo tiempo que ofrece seguridad en un proceso tan importante y delicado como es el proceso de verificación de calidad y pruebas.
    - ✓ **Para generar confianza:** De manera final, tener una definición completa y clara de los requerimientos de la aplicación es una herramienta para que los gerentes, los desarrolladores, los analistas, los encargados de pruebas y desde luego los usuarios tengan tranquilidad y confianza en el proceso. Partir de la base de que todas las personas involucradas en el proceso están trabajando con miras a cumplir el mismo

conjunto de objetivos y funcionalidades, minimizando la probabilidad de desacuerdos y malentendidos posteriores, es una gran ventaja en cualquier proceso de desarrollo de software.

Características de los requerimientos

Según la IEEE Std 830-1998[2] las características deseables en un requerimiento para que cumpla con su propósito son:

Independiente del diseño: la manera como se escribe un requerimiento no debe depender de decisiones o restricciones en términos de plataformas, lenguajes, sistemas operativos. Los requerimientos deben describir funcionalidades que debe tener la aplicación, sin limitarse por la manera técnica en la que serán implementadas. Dicho de otra manera, dice qué debe hacer el sistema, no cómo será implementado.

Priorizable: esta característica responde a la necesidad de poder ordenar los requerimientos en términos de su importancia dentro del proyecto. Esto se hace con el fin de poder dividir el trabajo más adelante y también con propósitos eventuales de negociación, para poder determinar cuáles son los requerimientos que deben incluirse de manera primordial en una liberación parcial del *software* y a cuáles se debe dedicar más tiempo de investigación, desarrollo y pruebas.

Necesario: cualquier requerimiento que se enuncie durante el proceso de construcción de una aplicación de *software* debe ser necesario para su funcionamiento. Requerimientos que no lo sean solamente consumirán tiempo, recursos y esfuerzo, al mismo tiempo que potencialmente podrían enturbiar el proceso de desarrollo y utilización de la aplicación sin agregar valor a la misma.

No ambiguo: la manera en que se describe un requerimiento no debe dejar lugar a dudas respecto a su intención. Si esto sucede, es posible que haya interpretaciones variadas de lo que se desea con dicho requerimiento, y esto puede producir más adelante dificultades durante el proceso de desarrollo e insatisfacción por parte del cliente.

Verificable: un requerimiento debe estar escrito de tal manera que se pueda comprobar su implementación. La manera como se enuncia debe soportar el hecho de que se verifique más adelante que efectivamente fue implementado y que dicha implementación cumple con lo deseado. Clave en este sentido es definir de manera concreta qué se espera como salida o resultado de la correcta implementación y ejecución del requerimiento.

Correcto: posiblemente sobra decirlo, pero un requerimiento debe ser correcto en su enunciado. Es decir, debe reflejar de manera adecuada lo que se espera de la aplicación y debe contener la descripción de una funcionalidad que sea apropiada para la solución de *software* que se está implementando.

Consistente: en estrecha relación con la característica anterior, un requerimiento debe estar alineado con los demás requerimientos de la aplicación. Un requerimiento no debe nunca contradecir lo que otros digan en términos de la funcionalidad de la aplicación. Detectar y corregir este tipo de inconsistencias es lo que

[2] https://standards.ieee.org/standard/830-1998.html

facilita que la aplicación al final de su ciclo de implementación satisfaga de manera cabal la totalidad de las necesidades del cliente.

Realizable: un requerimiento debe ser posible de implementar. A pesar de que en un campo como la tecnología la definición de qué es y no es posible varía con rapidez, debe tratarse de que la definición de los requerimientos de cada aplicación sea, desde el principio, lo suficientemente aterrizada como para que se pueda iniciar el desarrollo con una razonable certeza de que se podrá cumplir con todo lo enunciado.

Trazable: un requerimiento es trazable cuando se le puede encadenar con todos los elementos de la definición del *software* que lo generaron y si es el caso, con los elementos que él mismo genera. Esto quiere decir, que debe ser posible identificar para un requerimiento funcional el requerimiento (o requerimientos) de usuario que ayuda a satisfacer y de manera más general aun, el requerimiento de negocio al que contribuye. A futuro dentro del proceso de desarrollo, también debe ser posible hacer este seguimiento hacia abajo en términos de poder identificar las unidades de código y las clases y métodos que contribuyen a la satisfacción del requerimiento enunciado. De esta manera, a lo largo de un proyecto de *software* debería ser posible identificar qué método de qué clase contribuye a la satisfacción de qué requerimiento de negocio, siguiendo todos los pasos intermedios.

Conciso: un requerimiento debe ser escrito de la manera más breve posible, sin que por ello pierda claridad. Esto facilita la lectura y la comprensión de lo que se desea y puede ser de ayuda si se requiere hacer correcciones más adelante.

Escrito en forma de debe: esta es más una recomendación de forma con el fin de alcanzar consistencia en la documentación de los requerimientos. Por lo general se recomienda que sean escritos de esta manera: la aplicación debe…. así se facilita la lectura de la lista de requerimientos y es posible verificar también la claridad que se tiene al momento de escribirlos. Si un requerimiento no se puede escribir de esta manera, es posible que exista la necesidad de revisar y redefinir qué se espera que describa. Esta es otra herramienta para aclarar los requerimientos de la aplicación.

• Tipos de requerimientos

Para una adecuada identificación y especificación de requerimientos es importante establecer la distinción entre los que requerimientos funcionales y los no funcionales.

• Requerimientos funcionales

Los requerimientos funcionales definen qué hace el sistema (a través de la definición de entradas y salidas), es decir, las funciones que dicho sistema debe cumplir. Este tipo de requerimientos son los que definen de una manera detallada las funcionalidades concretas que el *software* debe contener con el fin de permitir a los usuarios cumplir con sus labores, de tal manera que los requerimientos de negocio sean satisfechos. Los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer.

• Requerimientos no funcionales

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a sus propiedades emergentes como la confiabilidad, disponibilidad, tiempo de respuesta entre otros. Estos requerimientos se pueden establecer a la luz de los atributos de calidad del sistema. Adicionalmente, definen las

restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interface del sistema.

Los atributos de calidad como su nombre lo indica, son características que influyen significativamente en la calidad del software. Bass[3] estableció una clasificación de los atributos de calidad en dos categorías que siguen vigentes hasta ahora:

Observables vía ejecución: son aquellos atributos que se determinan del comportamiento del sistema en tiempo de ejecución, es decir que además son perceptibles por el usuario final, estos se describen en la Tabla 5. Descripción de atributos de calidad observables vía ejecución.

Atributo	Descripción
Disponibilidad (*availability*)	Es la medida de disponibilidad del sistema para el uso
Confidencialidad (*confidentiality*)	Es la ausencia de acceso no autorizado a la información
Funcionalidad (*functionality*)	Habilidad del sistema para realizar el trabajo para el cual fue concebido, es decir, para cumplir los requerimientos funcionales.
Desempeño (*perfomance*)	Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria. Según Smith (1993), el desempeño de un sistema se refiere a aspectos específicos o el número de eventos procesados en un intervalo de tiempo. Según Bass (1998) se refiere además a la cantidad de comunicación e interacción existente entre los componentes del sistema.
Confiabilidad (*reliability*)	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo.
Seguridad externa (*safety*)	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información.
Seguridad Interna (*security*)	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos.

Tabla 5. Descripción de atributos de calidad observables vía ejecución

No observables vía ejecución: son aquellos atributos que se establecen durante el diseño, desarrollo y mantenimiento del sistema, por lo tanto, son más perceptibles para los ingenieros, quienes implementan y mantienen el sistema.

Atributo de calidad	Descripción
Configurabilidad (*configurability*)	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema.

[3] Bass, L (2013). *Descripción de Atributos de Calidad no observables vía ejecución*.

Integrabilidad (*integrability*)	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente para ser integrados.
Integridad (*integrity*)	Es la ausencia de alteraciones inapropiadas de la información.
Interoperabilidad (*Interoperability*)	Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema. Es un tipo especial de integrabilidad.
Modificabilidad (*modifiability*)	Es la habilidad de realizar cambios futuros al sistema.
Mantenabilidad (*maintainability*)	Es la capacidad de someter a un sistema a reparaciones y evolución. Capacidad de modificar el sistema de manera rápida y a bajo costo.
Portabilidad (*portability*)	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser *hardware*, *software* o una combinación de los dos.
Reusabilidad (*reusability*)	Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedas de reutilizados en futuras aplicaciones.
Escalabilidad (*scalability*)	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental.
Capacidad de prueba (*testability*)	Es la medida de la facilidad con la que el *software*, al ser sometido a una serie de prueba, puede demostrar sus fallas. Es la probabilidad de que asumiendo que tiene al menos una falla, el *software* fallará en su ejecución de prueba.

Tabla 6. Descripción de atributos de calidad no observables vía ejecución

• Definición de requerimientos

De acuerdo con Pressman[4], para una adecuada gestión de requerimientos se requiere desarrollas las siguientes tareas: concepción, indagación, elaboración, negociación, especificación, validación y administración.

- **Concepción:** Esta es la etapa en la que se identifica una necesidad u oportunidad de negocio que puede satisfacerse a través de la construcción o modificación de un producto de software, lo que da lugar a concebir el proyecto. En esta etapa se busca entender con claridad cuál es el problema que se requiere resolver a quién le interesa que se dé solución a este problema, así como la viabilidad en el desarrollo del proyecto. Esta etapa, generalmente se desarrolla en el dominio de Gobierno de TI, en donde se evalúan las necesidades de los procesos y se desarrolla la evaluación de la viabilidad del proyecto y se establecen los acuerdos de desarrollo a un alto nivel.

- **Indagación:** Esta etapa tiene como propósito de respuesta a las siguientes preguntas: “cuáles son los objetivos para el sistema o producto, qué es lo que va a lograrse, cómo se ajusta el sistema o producto a las necesidades del negocio y, finalmente, cómo va a usarse el sistema o producto en las operaciones cotidianas.”

[4] Pressman, R. (2010). *Ingeniería del software. Un enfoque práctico*. México, D. F, México: McGraw Hill

Tenga en cuenta. Cristel y Kang definieron algunos problemas que suelen encontrarse en la etapa de indagación.

Problemas de alcance: el límite del sistema está mal definido, los clientes solicitan más de lo que puede o quiere realizarse o incluyen detalles técnicos que no son necesarios y en lugar de esto pueden confundir más que clarificar los objetivos del sistema. Es común que los clientes hablen de cómo creen que debería darse la solución que de identificar el problema.

Problemas de comprensión: los clientes no están completamente seguros de lo que necesitan, tienen una escaza comprensión de las capacidades y limitaciones de su entorno de computación, no existe un total entendimiento del problema, etc. En algunos casos definen requerimientos contradictorios argumentando "necesidades especiales" no comprendidas.

Problemas de volatilidad: los requisitos cambian con el tiempo. Algunos factores que generan cambios son: legislación, cambios en las políticas o estrategias de negocio, reestructuración, integración con otras organizaciones, etc.

- **Elaboración:** Esta etapa se centra en desarrollar modelos refinados de los requerimientos a partir de la información recopilada en las fases de concepción e indagación. Estos modelos pueden ser de los siguientes tipos:
    - Modelos basados en el escenario de los requerimientos desde el punto de vista de distintos actores del sistema. Ejemplo: casos de uso, historias de usuario, storytelling.
    - Modelos de datos, que ilustran el dominio de información del problema. Ejemplo: modelos conceptuales.
    - Modelos orientados al flujo, que representan los elementos funcionales del sistema y la manera como transforman los datos a medida que se avanza a través del sistema. Ejemplo: diagramas de flujos de datos.
    - Modelos de comportamiento, que ilustran el modo en el que se comparte el *software* como consecuencia de "eventos externos." Ejemplo: prototipos.
    - Modelos orientados a clases, que representan clases orientadas a objetos (atributos y operaciones) y la manera en la que las clases colaboran para cumplir con los requerimientos del sistema.
- **Negociación:** Luego de identificar los requerimientos, es común que se encuentren requerimientos que exceden las posibilidades dados los recursos con los que se cuenta, o que se presenten conflictos con requerimientos que se contradicen. En esta etapa se deben identificar dichos conflictos y priorizar los

requerimientos y resolver los conflictos, eliminando, combinando o modificando los requerimientos buscando la mayor satisfacción posible a todos los involucrados.

- **Especificación:** En esta etapa se debe lograr un documento escrito que formalice a través de lenguaje natural y modelos gráficos la descripción de los requerimientos.

- **Validación:** En esta esta se analiza la especificación de los requerimientos para garantizar que cumplen con las características requeridas. En esta etapa deben participar tanto los ingenieros que desarrollan la especificación como los usuarios o clientes.

- **Administración de requerimientos:** Es el conjunto de actividades que al equipo del proyecto a identificar, controlar y dar seguimiento a los requerimientos y a sus cambios en cualquier momento del desarrollo del proyecto.

El resultado de esta fase es uno o varios documentos que reflejen la totalidad de lo que el cliente necesita de la aplicación. Estos documentos servirán como punto de partida para que el equipo de análisis y diseño defina la mejor manera de abordar el proceso de construcción de *software* y para que de esta manera se lleve a cabo un proceso ordenado y eficiente desde el principio, orientándose a la satisfacción real de las necesidades del cliente.

(**Importante**) Para lograr la alineación con los procesos y acordar los alcances se conforman los comités de acuerdo de desarrollo de sistemas de información liderados por el área de TI y en los que participan los líderes de las áreas misionales y de apoyo de la organización.

3.2.3 Fase de desarrollo o adquisición y mantenimiento

En esta fase se consideran todas las actividades y aspectos que van a permitir obtener los artefactos y la solución de software que satisfacen las necesidades identificadas en la fase anterior garantizando que su calidad.

Desarrollar internamente o adquirir

Una de las preguntas a las que se enfrentan constantemente las entidades es decidir si realizar los desarrollos de software con equipos internos o tercerizar esta labor a fábricas de software. Es recomendable desarrollar un enfoque mixto en donde se cuente con una capacidad interna para hacer desarrollos puntuales o coyunturales que permitan suplir necesidades rápidamente, así como para dar soporte y mantenimiento y tercerizar el desarrollo de proyectos de mayor envergadura.

Cada uno de estos enfoques representa diferentes desafíos en la gestión de sistemas de información, a continuación, se mencionan algunas recomendaciones a tener en cuenta en el caso de tercerizar o desarrollar y/o fortalecer la capacidad de desarrollo interna. (Algunos aspectos son transversales a los dos enfoques)

Se debe definir una metodología de gerencia de proyectos estandarizada y basada en las mejores prácticas.

(**Importante**) Tanto los proyectos de desarrollo de software con una capacidad interna como tercerizados a través de fábricas de software, es importante considerar los lineamientos definidos en el Modelo de Gestión de Proyectos de TI.

- Un esquema tercerizado de desarrollo de software (fábrica de software o esquema de desarrollo, mantenimiento y evolución de software) debe garantizar eficiencia en la gestión desde la definición de las necesidades hasta la entrega de productos, logrando cumplimiento en los tiempos, costos acordes al esfuerzo realmente invertido.
- De cualquier manera, se debe trabajar en un esquema de planeación y aseguramiento y control de calidad.
- Es necesario contar con equipos técnicos con conocimientos especializados para desarrollar la supervisión y auditorías técnicas sobre los entregables de cada proyecto.
- Los procesos de entrega a producción de los productos deben garantizar un correcto, oportuno y adecuado despliegue en el ambiente de producción, gestionando cambios, capacidad, minimizando el riesgo de impactar la operación de la tecnología en funcionamiento
- Se deben definir acuerdos de nivel de servicio y gestionarse adecuadamente para prevenir incumplimientos y en caso de darse estos deberán implicar menores costos para la entidad contratante.
- Los terceros que se encarguen de implementar el modelo de fábrica de software deberán ser empresas que demuestren tener conocimiento y experiencia en análisis, diseño, desarrollo, parametrización, mantenimiento y evolución de software, así como tener los equipos de trabajo competentes para realizar su tarea en el marco de una metodología de desarrollo claramente definida.
- Los equipos de trabajo deben contar con los roles que se requieran para llevar a cabo los proyectos, entre otros, los siguientes: gerentes de proyectos, ingenieros de calidad, analistas de requerimientos, especialistas técnicas, arquitectos de software, desarrolladores, analistas de pruebas, diseñadores gráficos, documentadores, capacitadores, ingenieros de infraestructura tecnológicas, ingenieros de red, ingenieros de seguridad, administradores de bases de datos, científicos de datosenre otros según requiera o amerite el proyecto.
- En el esquema de fábrica de software es fundamental contar con la metodología y las herramientas adecuadas para determinar, anticipadamente y de la forma más objetiva posible, el nivel de esfuerzo que implica entregar productos de software, que se traduzca en horas hombre de los diferentes roles que participan en las distintas fases de desarrollo de software.
- Es necesario implementar herramientas que permitan hacer seguimiento y control al cumplimiento de la metodología y del cubrimiento de los requerimientos solicitados, la ejecución de la bolsa de horas de desarrollo, así como herramientas que permitan medir el cumplimiento de los acuerdos de niveles de servicio pactados, que se debe traducir en ahorros para la entidad. El seguimiento y control del cumplimiento de lo pactado y los acuerdos de niveles de servicio puede ser asumido por el interventor de la fábrica, que también puede ser un tercero o un gerente de proyecto interno

3.2.4 Análisis y diseño arquitectural y detallado de Sistemas de Información

En la etapa de análisis se transforman los requerimientos de las partes interesadas en un conjunto de requisitos técnicos deseados que guiarán el diseño del sistema. En cuanto al diseño, se definen dos niveles de abstracción, el diseño arquitectural y el diseño detallado.

Con respecto al diseño arquitectural se busca generar el diseño de alto nivel de los principales elementos o componentes del sistema y cómo están relacionados para satisfacer los requisitos técnicos definidos en el análisis, omitiendo detalles de implementación. Mientras que el diseño detallado tiene un nivel de detalle suficiente para orientar la codificación y las pruebas.

Recuerde. "La arquitectura del software de un programa o sistema informático es la estructura o estructuras del sistema, que comprende elementos de software, las propiedades visibles externamente de esos elementos y las relaciones entre ellos. En otras palabras, son los planos y las bases de un edificio" (Bass, 2012).

La arquitectura de software se define desde las siguientes perspectivas:

1) La arquitectura conceptual que describe el sistema en términos de sus principales elementos de diseño y las relaciones entre ellos,

2) La arquitectura de interconexión de módulos abarca dos estructuras ortogonales: descomposición funcional y capas,

3) La arquitectura de ejecución describe la estructura dinámica de un sistema y

4) La arquitectura del código describe cómo se organizan el código fuente, los binarios y las bibliotecas en el entorno de desarrollo.

Adicionalmente, la arquitectura de software debe hacerse cargo de: 1) relacionar los elementos arquitecturales y sus relaciones con la satisfacción de las necesidades de los interesados y 2) el diseño y evolución del sistema, estos dos aspectos se logran a través de las decisiones arquitecturales, a continuación, se define este concepto.

- **Decisiones Arquitecturales**

La estructura del sistema, que se representa por medio de la arquitectura no se define de manera aleatoria, sino que debe tomarse de manera que promueva los objetivos del sistema y el cumplimiento de los atributos de calidad deseados. Para esto el arquitecto debe decidir entre distintas alternativas cómo va a fraccionar el sistema, es decir, qué elementos harán parte del sistema y cómo se relacionarán dichos elementos. Dichas decisiones son llamadas decisiones arquitecturales. Una decisión arquitectural es: "Una elección entre las alternativas de diseño arquitectural. Esa elección se propone para alcanzar uno o más atributos de calidad del sistema, por medio de la(s) estructura(s) arquitecturales que esta envuelve o define"[5] Además, las decisiones arquitecturales deben cumplir con tres características fundamentales: descripción, objetivos y fundamentación.

- **Descripción:** es la explicación detallada de lo que se haya decidido para el sistema, ya sea generación o modificación de un elemento arquitectural en su estructura o comportamiento, la relación de un

[5] Arias, A., Durango, A. (2016). Ingeniería y Arquitectura del Software. IT Campus Academy.

elemento con otro, o la decisión de agregar diversos elementos para generar otro elemento que formar un servicio o componente más complejo o cualquier otra decisión que permita la evolución del sistema.

- **Objetivo**: es el propósito que se persigue al implementar dicha decisión, generalmente el objetivo debe estar definido en términos de que objetivos de negocio o atributos de calidad deseados se buscan favorecer. Es relevante tener en cuenta que un objetivo de negocio es favorecer un atributo de calidad del sistema y a su vez restringir otro u otros. Además, para lograr atender los atributos de calidad de un sistema, generalmente se deben implementar varias decisiones arquitecturales así que esto debe quedar muy claro en la definición del objetivo para poder evaluar si el objetivo de la decisión arquitectural logra el propósito esperado y rastrear las decisiones que han tomado en el tiempo para favorecer un atributo de calidad.
- **Fundamentación**: explica o justifica por qué se tomó esta decisión y no otra. Para esto el arquitecto puede apoyarse en estándares, buenas prácticas o patrones definidos por la industria que permite justificar que esta decisión se toma porque hay evidencia previa de que, de los mejores resultados en circunstancias similares, o porque al evaluar varias alternativas se puede prever que está es la alternativa que dará mejores resultados.

A continuación, se listan algunos criterios que propone Pressman[6] para determinar la calidad de un buen diseño:

1. Debe tener una arquitectura que 1) se haya creado con el empleo de estilos o patrones arquitectónicos reconocibles, 2) esté compuesta de componentes con buenas características de diseño, y 3) se implementen en forma evolutiva, de modo que faciliten la implementación y las pruebas.
2. Debe ser modular, es decir, el *software* debe estar dividido de manera lógica en elementos o subsistemas.
3. Debe contener distintas representaciones de datos, arquitectura, interfaces y componentes. Debe conducir a estructuras de datos apropiadas para las clases que se van a implementar y que surjan de patrones reconocibles de datos.
4. Debe llevar a componentes que tengan características funcionales independientes y desacoplados.
5. Debe conducir a interfaces que reduzcan la complejidad de las conexiones entre los componentes y el ambiente externo.
6. Debe obtenerse con el empleo de un método repetible motivado por la información obtenida durante el análisis de los requerimientos del *software.*
7. Debe representarse con una notación que comunique con eficacia su significado.

Además de los lineamientos mencionados anteriormente, la Arquitectura de Software debe garantizar que se toman decisiones arquitecturales frente a los requerimientos no funcionales descritos en la tabla 5. Descripción de atributos de calidad observables vía ejecución y tabla 6. Descripción de atributos de calidad no observables vía ejecución, e incorporar los principios de diseño de servicios digitales que definen que todo proyecto que haga uso de las TIC, debe ser diseñado para que desde el principio garantice interoperabilidad, seguridad de la información, accesibilidad, usabilidad, apertura y ubicuidad, teniendo en cuenta las necesidades y características de los usuarios. Para el caso de los proyectos que buscan hacer mejoras o ajustes a los sistemas existentes, deberán incorporar estos atributos de acuerdo con el tipo de infraestructura tecnológica con que cuente la entidad, el presupuesto y su ruta de transformación digital.

[6] Pressman, R. (2010). Ingeniería del software. Un enfoque práctico. México, D. F, México: McGraw Hill (pp. 1,26)

Los principios a considerar o incorporar en el diseño de los componentes de sistemas de información son:

- **Interoperabilidad**: todo proyecto que incorpore el uso de las TIC y genere información, debe asegurar la implementación del marco de interoperabilidad, a fin de desarrollar capacidades para el intercambio fácil, seguro y transparente de la información entre entidades públicas y de ser necesario, con entidades privadas. Para ello, haga uso de los documentos relacionados con el Marco de Interoperabilidad[7].
- **Seguridad de la información**: todo proyecto que incorpore el uso de las TIC, debe aplicar desde su concepción y diseño, el modelo de seguridad y privacidad de la información a nivel de procesos, sistemas de información e infraestructura tecnológica. Para ello haga uso de los documentos del modelo de seguridad y privacidad de la información[8]
- **Accesibilidad y usabilidad**: todo proyecto que incorpore el uso de las TIC, debe contar con características que faciliten el acceso y uso de las herramientas web dispuestas para los usuarios, de manera que estos puedan percibir, entender, navegar e interactuar adecuadamente, así como usarlas fácilmente.
- **Apertura**: todo proyecto que incorpore el uso de las TIC y genere información, debe contar con sistemas de información que permitan la generación de datos abiertos de manera automática para su publicación, uso y reutilización. De igual manera, es deseable que se privilegie el uso o desarrollo de software con licencias de código abierto, a fin de posibilitar su publicación y reutilización sin costo. Adicionalmente, la entidad debe cumplir con lo dispuesto en la Resolución No. 3564 de 2015, en donde se establecen los estándares para publicación y divulgación de la información.
- **Ubicuidad**: todo proyecto que incorpore el uso de las TIC y disponga de herramientas para la consulta y aprovechamiento de información, así como para realizar transacciones con usuarios, ciudadanos y grupos de interés, debe facilitar el acceso a dichos servicios a través de dispositivos móviles como celulares y tabletas, logrando que el servicio pueda ser usado en cualquier momento, en cualquier dispositivo y cualquier lugar. Lo anterior, de acuerdo con las necesidades y características de los usuarios.
- **Uso de tecnologías emergentes**: Todas las entidades deben identificar y evaluar la posibilidad de incorporar tecnologías emergentes para el desarrollo de sus proyectos. Actualmente en el sector público se vienen aplicando tecnologías emergentes como internet de las cosas IoT, blockchain, robótica, inteligencia artificial entre otras, para mejorar tiempos de respuesta al ciudadano, contar con información en tiempo real para la toma de decisiones, realizar actividades de monitoreo y generación de alertas tempranas, entre otros.

3.2.5 Construcción de los componentes de Sistemas de Información

En esta fase se producen o desarrollan los distintos componentes o elementos del software del sistema especificado. Incluye la construcción de las unidades de software que son ejecutables y que es conforme con el diseño previamente definido, la ejecución de las diferentes de pruebas previstas en el plan de pruebas, la integración en la que se combinan las diferentes unidades y se verifica que cumplen con los requerimientos funcionales y no funcionales.

[7] http://lenguaje.mintic.gov.co/marco-de-interoperabilidad
[8] https://www.mintic.gov.co/gestion-ti/Seguridad-TI/Modelo-de-Seguridad/

3.2.6 Documentación de los Sistemas de Información

Durante todo el proceso se debe garantizar que se desarrolla la documentación técnica y de usuario de todos los componentes del sistema. La Documentación contribuye al agestión del conocimiento y facilita procesos de mantenimiento correctivo y evolutivo del software.

Todas las etapas del ciclo vital del software en el desarrollo de cualquier software se registran en la documentación. La documentación sirve como información escrita sobre definición de requerimientos, especificaciones generales del sistema, especificación de cada componente y los planes integrales de prueba y mantenimiento. Las herramientas de gestión de configuración también son de gran utilidad en la documentación del software.

La importancia de la documentación para desarrolladores radica en el hecho de que contiene la información sobre las operaciones del sistema de software. Esta información posibilita la reproducción del software o su adaptación para mantenimiento. Además de garantizar la transferencia del conocimiento ya que las personas que conforman los equipos suelen rotar frecuentemente.

Por otro lado, los manuales de usuario final facilitan el uso y apropiación del sistema, facilitando las actividades de capacitación y reduciendo el tiempo de entrenamiento para lograr su uso productivo. Adicionalmente, la documentación es la referencia para los administradores del sistema y el personal de soporte del mismo ya que detallan la forma de operarlo.

Tip. La documentación técnica tiende a ser extensa, pormenorizada y suele actualizarse con frecuencia, por lo que resulta especialmente importante contar con las herramientas adecuadas para garantizar la exactitud y ahorrar tiempo en su desarrollo.

3.2.7 Mantenimiento de los Sistemas de Información

En esta fase se maneja la corrección de errores que se detecten en el software, implementaciones que se vayan realizando debido a los nuevos requerimientos del cliente, y cambios en el software debido a las modificaciones que se presenten en su entorno, en esta fase se presentan cuatro tipos de cambios:

- **Corrección**: Se realiza para corregir errores que el cliente encuentre en el software.
- **Adaptación**: Debido a cambios que se puedan presentar en el entorno exterior del software, será necesario hacer adaptaciones al mismo, se pueden presentar en: cambios del sistema operativo, equipo, reglas de negocio que van cambiando, entre otros.
- **Mejora**: Cuando el cliente a medida que utiliza el software va descubriendo funciones extra que van a ser de su beneficio; este mantenimiento va más allá de los requerimientos iniciales.
- **Prevención**: Este mantenimiento consiste en realizar cambios con la finalidad de que el software se pueda adaptar, corregir y realizar mejoras de manera más fácil debido a que el software se deteriora por los cambios que se generan en su entorno.

Es necesario establecer procedimientos claramente definidos y estandarizados para el mantenimiento software, que se basen en técnicas y herramientas claramente definidas y validadas. Asignarle los recursos adecuados, tanto físicos y económicos como humanos. Las actividades de mantenimiento deben desarrollarse con un enfoque de gestión de cambio en el que se consideren las actividades sugeridas en la ilustración 5. Proceso de evaluación del sistema.

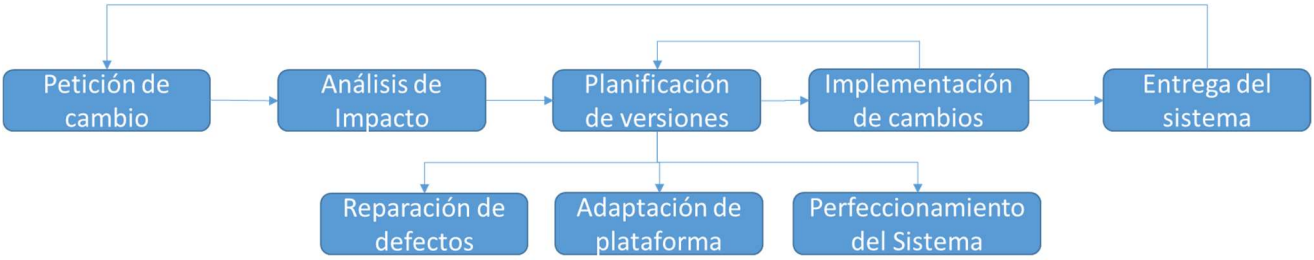


Ilustración 5. Proceso de evaluación del sistema. (elaboración propia)

En cuanto a los sistemas de información que son provistos por terceros, es necesario definir y gestionar los acuerdos de nivel del servicio para mantenimiento, los cuales deben incluir entre otros, los siguientes aspectos:

- Descripción
- Servicio y/o Funcionalidad sobre el que aplica el ANS.
- Métricas asociadas.
- Rango permitido para la métrica.
- Sanción o penalidad en caso de incumplimiento.
- Frecuencia de medición.
- Responsabilidades.

3.2.8 Fase de Implantación

En esta fase se consideran todas las actividades y aspectos que van a permitir que el sistema de información se despliegue en el ambiente productivo de una manera controlada y pueda ser usado y apropiado por los usuarios. Estas prácticas transcienden los límites del dominio de Sistemas de Información para complementarse con los dominios de Infraestructura Tecnológica, que se encarga de gestionar tanto la plataforma tecnológica como la operación del sistema de información, mientras que el dominio de Uso y Apropiación de TI, define e implementa todas las actividades requeridas para lograr el uso y apropiación efectivo del sistema de información hasta incorporarse en la cultura organizacional.

Además, la fase de implantación se sustenta especialmente en las actividades de aseguramiento de calidad que define y ejecuta las actividades de pruebas, verificación y validación que garanticen que los componentes del sistema de información cumplen con los requisitos y que se integrará de manera adecuada en el ambiente de producción.

Dentro de las prácticas recomendadas para garantizar una adecuada puesta en producción, se requiere considerar las siguientes:

3.2.9 Gestión de ambientes de desarrollo

Un aspecto fundamental que es necesario definir para el desarrollo de todo el ciclo de vida de desarrollo de sistemas de información, es la separación de ambientes de desarrollo con lo cual se busca determinar cuál es la configuración de los ambientes que permiten asegurar que los componentes del sistema de información que son desplegados en el ambiente de producción cumplen con las condiciones de calidad requeridas. De acuerdo con las necesidades y posibilidades de cada entidad y el sistema de información se pueden dar diferentes configuraciones, la Ilustración 6. Vista de separación de ambientes una posible configuración.

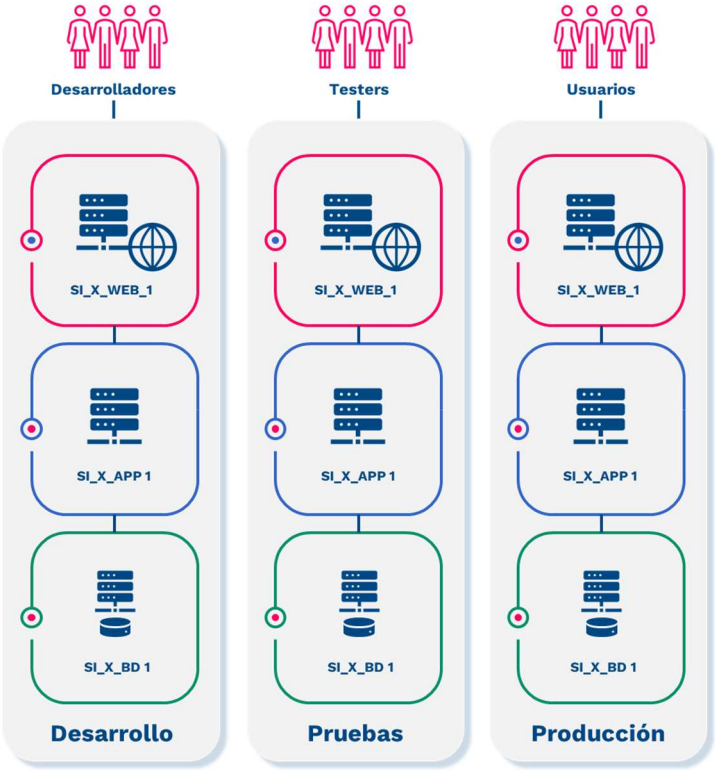


Ilustración 6. Vista de separación de ambientes (elaboración propia)

Tenga en cuenta. El ejemplo presentado en la Ilustración 6. Vista de separación de ambientes es una vista lógica, es decir que presenta los posibles servidores lógicos desplegados en cada ambiente, sin embargo, por medio de la virtualización pueden darse que varios servidores lógicos e inclusive varios ambientes estén contenidos en un servidor físico.

Algunos ambientes que pueden estar presente son:

- **Ambiente de Desarrollo:** es el ambiente donde trabajan los desarrolladores generando la codificación y configuración de los componentes del sistema de información, así como los documentos, manuales y demás artefactos producidos durante la construcción. A este ambiente sólo tienen acceso los desarrolladores.

- **Ambiente de Pruebas:** En este ambiente pueden dividirse en más ambientes en donde se realicen los diferentes tipos de pruebas que garanticen la calidad de los componentes construidos. En donde se desarrollen desde las pruebas unitarias del código, pruebas de integración, pruebas de calidad, pruebas de rendimiento, pruebas de aceptación de usuario, entre otros. A este ambiente tienen acceso los usuarios que desarrollen funciones de pruebas.

- **Ambiente de Producción:** Este es el ambiente donde se despliegan los componentes para su ejecución y al quien tienen acceso los usuarios finales. En este ambiente se debe garantizar la configuración que asegure los atributos de calidad incluidas condiciones para garantizar la continuidad y disponibilidad.

Integración continua

La integración continua es una práctica por la cual los desarrolladores integran o combinan el código en un repositorio común, facilitando la realización de test o pruebas para detectar y resolver posibles errores. Con la práctica de integración continua se impide que se desarrollen distintas divisiones de una aplicación que luego puedan tener conflictos entre sí. Los desarrolladores integran periódicamente su código en el repositorio de control de versiones central en lugar de realizarlo de forma aislada al final del ciclo de producción. De esta forma se descubren antes los conflictos entre los nuevos códigos y los existentes, haciendo que la resolución de los mismos sea más sencilla y acarree un menor costo.

En la integración continua, las pruebas son fundamentales para poder garantizar que el código es confiable y será necesario contar con un sistema automatizado para poder realizar todo el proceso de integración de código y pruebas.

Dentro de los beneficios[9] que se obtienen al implementar estas prácticas se pueden considerar los siguientes:

- Se incrementa la calidad del producto mediante la colaboración entre los desarrolladores, lo que hace más simple la resolución de problemas de integración de forma continua, y evita problemas durante las sucesivas fases de entrega del producto.
- Se incrementa la calidad del proceso: la automatización y la monitorización de las pruebas continuas por parte del servidor de integración permiten crear un proceso limpio de retroalimentación con los desarrolladores, que definirán una filosofía de desarrollo óptima, que reducirá al mínimo cualquier error en el producto final.
- Se incrementa la calidad de las personas: la implementación de la integración continua permite a los equipos trabajar en el desarrollo y la mejora continua de pruebas, así como ejercer buenas prácticas de programación que a la vez repercutirán en un código de mayor calidad.
- El proceso da seguridad al equipo, ya que en todo momento sabrá que el proyecto está funcionando.

[9] Guijarro Olivares, J. Caparrós Ramírez, J. y Cubero Luque, L. (2019). DevOps y seguridad cloud. Barcelona, Editorial UOC.

- Permite consolidar de manera centralizada el código fuente de todas las aplicaciones y sistemas de información que posee una entidad, así como devolver o recuperar una versión anterior del sistema en caso de ser necesario.

Entrega continua

La entrega continua se entiende como la evolución de la integración continua, de tal modo que para que los cambios realizados en desarrollo puedan ser susceptibles de ser entregados en producción en el menor tiempo posible, minimizando los riesgos de implantación. Gracias a este proceso, Es posible evitar largas puestas en producción, las sorpresas de última hora y los cambios muy grandes y costosos, evitando que haya demasiada incertidumbre a la hora de la entrega y teniendo un mayor control sobre todo el proceso.

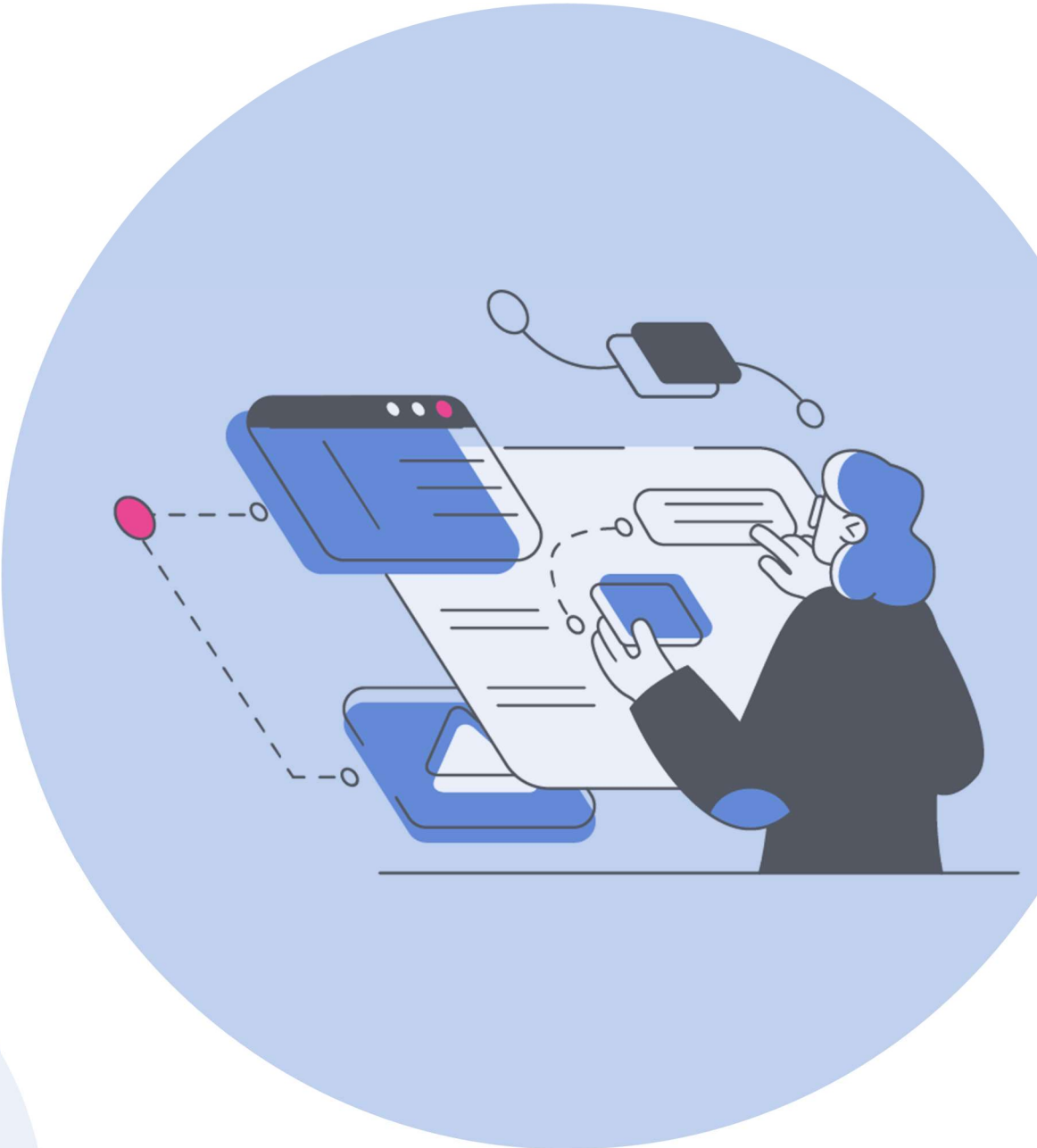
Despliegue continuo

El despliegue continuo o CD (continuous deployment) está relacionado estrechamente con la entrega continua. Con el despliegue continuo se va un paso más allá de la entrega continua, automatizando todo el proceso de entrega de software al usuario, eliminando la acción manual o intervención humana necesaria en la entrega continua.

Todo el proceso de despliegue sigue una serie de pasos que deben ejecutarse en orden y de forma correcta. Si alguno de estos pasos no se concluye de forma satisfactoria, el despliegue no se llevará a cabo. El despliegue continuo libera de carga a los equipos de operaciones de procesos manuales, que son una de la principal causa de retrasos en la distribución de aplicaciones. El uso de estas prácticas de mejora en los procesos de desarrollo de software aporta una serie de beneficios como la entrega o liberación inmediata de código, una simplificación del trabajo colaborativo o de equipo, permite una detección temprana de errores, disminuye costes de desarrollo (ahorra tiempo y esfuerzo) y favorece la comunicación entre todos los implicados en el proceso de desarrollo.

Tenga en cuenta. Las necesidades actuales a la hora de desarrollar y entregar software de calidad a los clientes de forma periódica, ha propiciado que metodologías como DevOps hayan sido adoptadas en la industria del desarrollo de software para poder automatizar procesos y aplicar técnicas como la integración continua, la entrega continua y el despliegue continuo. Estas técnicas ayudan a que los desarrolladores puedan entregar su código de forma periódica, compartiéndolo en un repositorio común donde puede ser comprobado y corregido, para así poder entregar el software al cliente de forma rápida y fiable.

4 Roles

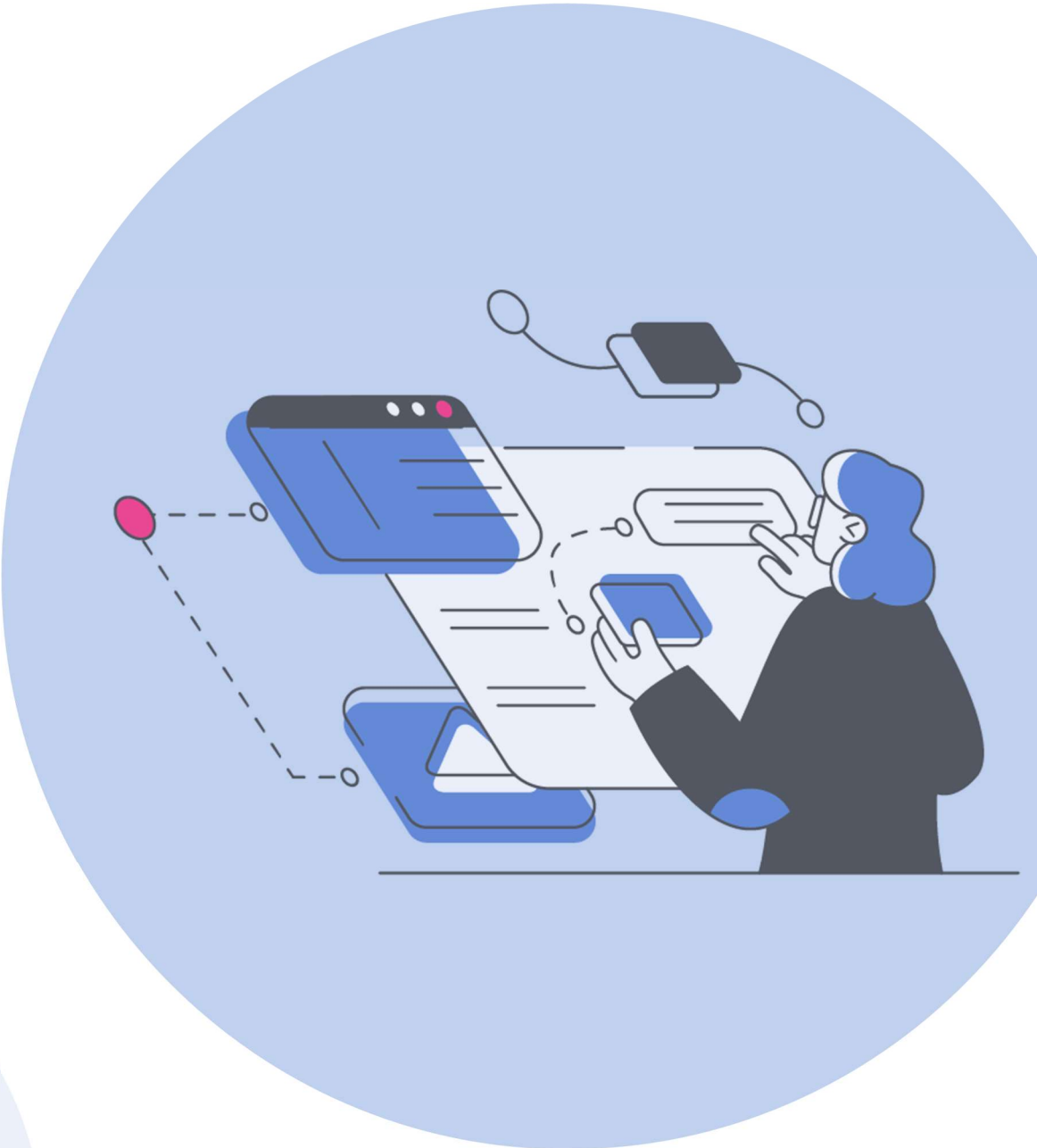


En el desarrollo de las actividades para definir y gestionar los sistemas de información deberían participan los siguientes roles:

Rol	Responsabilidades
Director(a) de Tecnología y Sistemas de la Información	• Liderar la definición y estrategia de Transformación digital.
Arquitecto de soluciones	• Definir y mantener actualizada la arquitectura de referencia de sistemas de información y las arquitecturas de solución de los sistemas de información. • Definir estándares y lineamientos para la construcción de soluciones de software.
Analista en experiencia de usuario	• Diseñar los prototipos (*Mockups*) de las interfaces de usuario (*front End*) de las aplicaciones y sistemas de información aplicando criterios de usabilidad y accesibilidad de acuerdo con los grupos de interés.
Gerente de proyecto	• Coordinar y gestionar los proyectos de desarrollo de software. • Gestionar los grupos de interés que tienen expectativas y requerimientos en los proyectos de software.
Analista de calidad y pruebas	• Estructurar los planes de calidad y pruebas sobre las soluciones y sistemas de información. • Ejecutar los planes de calidad y pruebas.
líder de desarrollo de software	• Definir y construir los planes de mantenimiento evolutivo y correctivos de las soluciones de software y sistemas de información de la entidad. • Coordinar, gestionar y hacer seguimiento a los planes de trabajo de los desarrolladores internos o externos a la entidad. • Definir e implementar el procedimiento de control de cambios de las aplicaciones y soluciones de software.} Gestionar el repositorio de versionamiento de código fuente.
Analista de requerimientos	• Elaborar la especificación de requerimientos de software o historias de usuario de acuerdo con las metodología y formatos definidos por la entidad.
líder de infraestructura	• Aprovicionar la infraestructura necesaria para el despliegue de los diferentes ambientes de las soluciones o aplicaciones de software (desarrollo, pruebas, producción).

Tabla 7 Roles

5 Artefactos



Durante el proceso de gestión de sistemas de información, pueden usarse entre otros, los siguientes artefactos propuestos:

Tipo	Nombre	Descripción
Documento	Plan de Mantenimiento evolutivo y correctivo de sistemas de información	Contiene los sistemas de información, las fechas y responsables de ejecutar el plan de mantenimiento correctivo o evolutivo
Documento	Fichas de especificación de requerimientos o historias de usuario	Contiene la descripción de los requerimientos funcionales y no funcionales.
Documento	Plan de pruebas de software	Contiene la planeación y estructuración de los tipos de pruebas, responsables, tiempos, recursos entre otros.
Documento	Arquitectura de Referencia	Este documento contiene varios artefactos y puede ser actualizada durante el proceso o procedimiento de gestión de sistemas de información.
Documento	Arquitectura de solución	Este documento contiene varios artefactos que son actualizados durante la ejecución del proceso o procedimiento de gestión de sistemas de información.
Documento	Prototipos de interfaces de usuario (Mockups)	Hace referencia a los diseños de las interfaces gráficas de los aplicativos de acuerdo con la guía de estilo de la entidad.
Matriz	Catálogo de sistemas de información	Contine el inventario y caracterización de los sistemas de información.
Documento	Manual de usuario	Contiene la descripción de las funcionalidades del sistema de información.
Documento	Manual de instalación, configuración y despliegue	Contine la descripción detallada y paso a paso de las instrucciones para el despliegue y configuración de las aplicaciones o soluciones de software.
Código fuente	Código fuente de los aplicativos o soluciones de software	Contiene el código fuente de todos los componentes, servicios, APIs, y demás que conforman el sistema de información.
Hojas de estilo (CSS)	Hojas de estilo	Corresponde a las hojas de estilo de las aplicaciones de software o sistemas de información.
Documento	Plan del proyecto	Contiene la planeación del proyecto. Este artefacto puede o no existir dependiendo de la metodología, teniendo que cuenta que si se está trabajando metodologías agiles se habla de producto Backlog.

Tabla 8 Artefactos