WISSEN
ESCUELA DE EMPRESAS

INGENIERÍA DEL SOFTWARE

Desarrollo de Software

Innovación que
TRANSFORMA

WISSEN
SCUELA DE EMPRESAS

FICHA
GENERAL

DESARROLLO DE SOFTWARE

Modalidad: Híbrida

Materia: Ingeniería del Software

Docente: Fernando Uyaguari

2023-2024

AUTOR / OBRA
PhD. Fernando Uyaguari

REVISORES INTERNOS
Ing. Álvaro Uyaguari

COLABORADORES
Diego Vintimilla

DISEÑO Y DIAGRAMACIÓN
Diego Vintimilla
Andrés Bernal

Primera Edición, 2024

Av. 10 de agosto y J. María Sánchez Cuenca – Ecuador
(+593) 0987934934
investigacion@wissen.edu.ec
www.wissen.edu.ec

Año de publicación: 2024

INTRODUCCIÓN

La presente guía didáctica tiene como objetivo dotar al estudiantado de las competencias necesarias para abordar el desarrollo de sistemas de software de manera profesional, práctica y eficiente. A través de la presente guía, se busca que el estudiantado adquiera los conocimientos fundamentales sobre el análisis de requisitos, diseño de soluciones, y el uso de metodologías ágiles y tradicionales en la gestión de proyectos de software.

Además, se hace hincapié en el desarrollo de habilidades prácticas para escribir un código limpio, modular y bien documentado, y en la aplicación de principios de ingeniería de software como la reutilización de código y la realización de pruebas para asegurar la calidad de los productos.

Descripción de la asignatura

La asignatura desarrolla la capacidad de analizar requisitos y diseñar soluciones de software mediante la creación de diagramas y artefactos que estructuran el sistema. Se enfoca en la planificación, dirección y coordinación de proyectos de software, utilizando metodologías ágiles o tradicionales y gestionando recursos, tiempos y riesgos.

Objetivo General de la asignatura

Formar a los futuros profesionales en el conocimiento y aplicación de técnicas y herramientas de ingeniería del software que les permita desarrollar proyectos de software de lata calidad y gestionar con éxito todas las etapas del ciclo de vida del software.

Resultados de Aprendizaje

- **Resultados de aprendizaje de conocimiento:**
  a. Conocer los principios de la ingeniería del software (IS), incluyendo sus metodologías, técnicas y herramientas.
  b. Comprender los conceptos clave de calidad de software y sus implicaciones en el desarrollo de proyectos.
- **Resultados de aprendizaje de destrezas:**
  a. Aplicar técnicas y herramientas de la ingeniería de software en el diseño, desarrollo y pruebas de proyectos de software.
  b. Implementar prácticas de calidad en el proceso de desarrollo de software para asegurar productos efectivos y alineados con los requisitos del cliente.
- **Resultados de aprendizaje de valores y actitudes:**
  a. Fomentar una actitud de compromiso con la calidad y la mejora continua en los procesos de desarrollo de software.

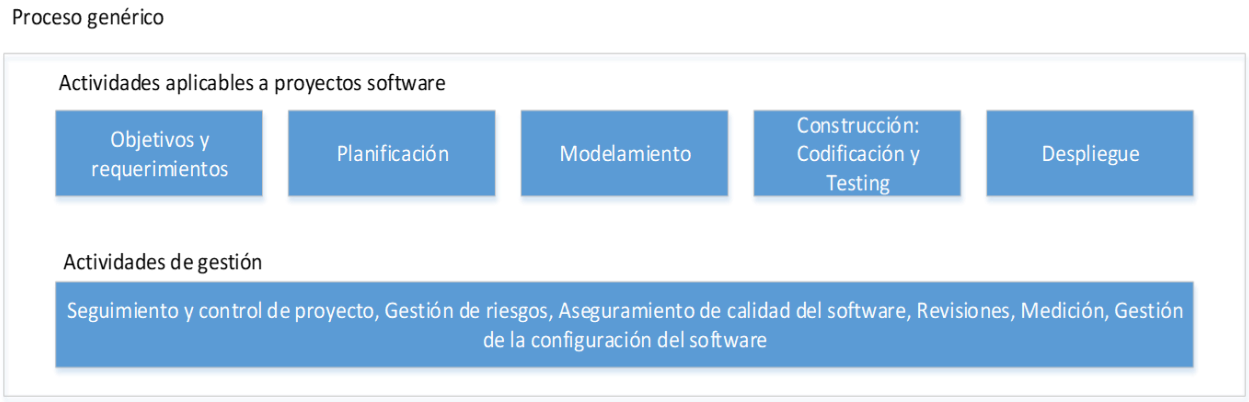
1.1 ¿Qué son los IS?

La aplicación de un enfoque sistemático, disciplinado, cuantificable al desarrollo, operación y mantenimiento de software [IEEE610]. Pero, ¿por qué toma tanto tiempo conseguir un software terminado? ¿Por qué no podemos encontrar todos los errores antes de entregar el software al cliente?

- Los requerimientos de TI a nivel individual, empresarial, gobierno han crecido al igual que la complejidad. El software está en los computadores y los dispositivos electrónicos
- Individuos, empresas, gobierno confían cada vez más en el software para las decisiones estratégicas y operativas del día a día. Por lo que el software debe mostrar alta calidad.

1.2 Proceso de Software

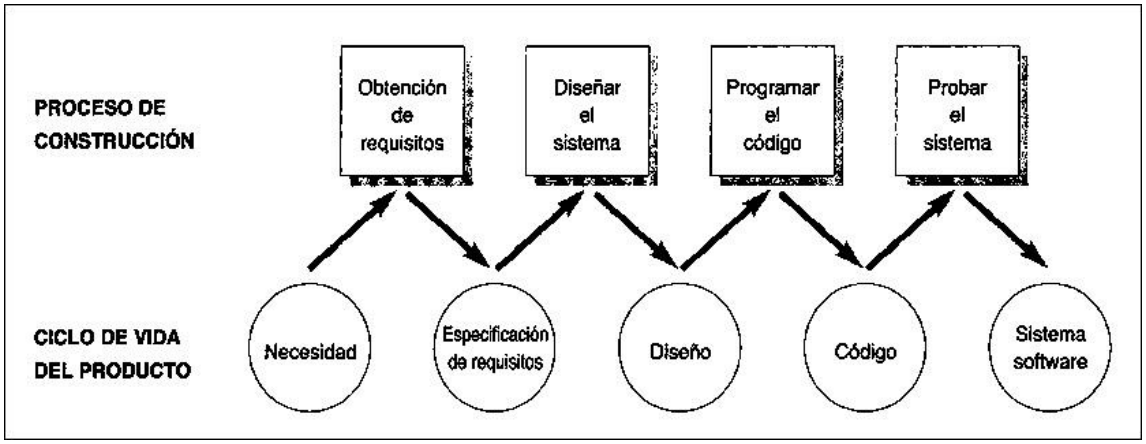
Un proceso es una colección de actividades cuando un producto está siendo creado. En el contexto del software, no es una actividad rígida sino adaptable. Se selecciona un conjunto apropiado de acciones y tareas.



El ciclo de vida apropiado se elige en base a los siguientes parámetros: cultura de la corporación, el deseo de asumir riesgos, el área de aplicación y la volatilidad de los requisitos (Ver Figura 1).

Figura 1

Proceso de construcción



Fuente: Elaboración Propia.

- **Un ciclo de vida debe:**

El ciclo de vida del software debe determinar el orden estructural del mismo. Por lo cual, es imperativo establecer criterios de transición para pasar de una fase a otra. Este debe construir una implementación parcial del sistema global y posteriormente ir aumentando la funcionalidad del sistema. Algunas de las características más importantes son:

→ De inicio se conocen la mayoría de los requisitos.
→ Produce un sistema operacional más rápidamente.

- **Prototipado**

La idea básica en la construcción inicial de un proyecto es que ayude a comprender los requisitos del usuario; sin embargo, esto se complica, dado que en la mayoría de los proyectos del software el cliente (externo), no tiene una idea muy detallada de lo que verdaderamente necesita. Por ello, en el ciclo de vida se debe incluir los siguientes puntos:

→ Análisis preliminar y especificación de requisitos.
→ Diseño, desarrollo e implementación del prototipado.
→ Prueba del prototipado
→ Refinamiento iterativo del prototipado
→ Refinamiento de las especificaciones de requisitos.

Es fundamental elegir el modelo de ciclo de vida más apropiado para nuestro proyecto, dado que esto nos ayudará a trazar el mapa de actividades para su operativización. Existen dos tipos de prototipado: 1) Evolutivo (implementación parcial de los requisitos conocidos) y/o 2) Desechable (implementación rápida y no parcial). Su elección dependerá de los objetivos y forma del proyecto.

EJEMPLO 1

Se trata del Departamento de Informática de una empresa aeronáutica. En el equipo de trabajo para este proyecto hay tanto profesionales experimentados como inexpertos. Los expertos ocuparán los puestos de jefe de proyecto, analista senior, etc. Es decir, los puestos críticos y los profesionales con poca experiencia estarán bajo el mando de los expertos. El producto software a desarrollar está en el dominio de la aeronáutica, conocido para los ingenieros software del Departamento de IT. El sistema, es altamente novedoso, puesto que intenta resolver un problema hasta ahora poco tratado con software.

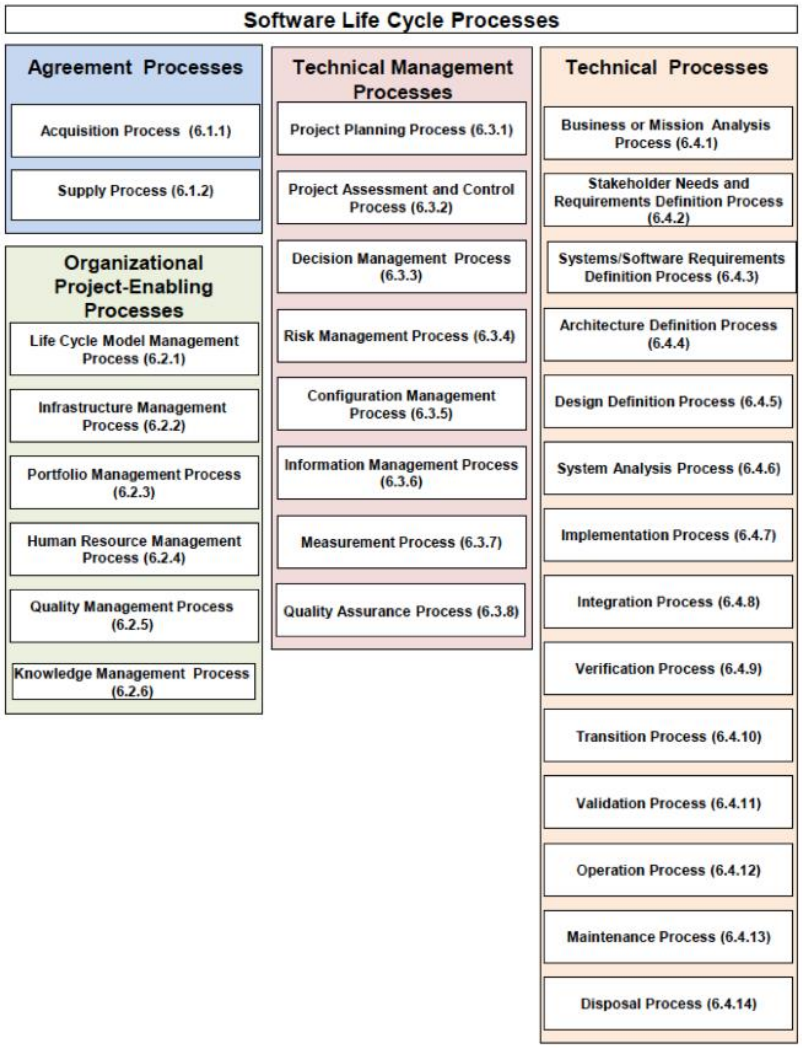
El Departamento IT desea tener los riesgos del proyecto constantemente controlados, pues, aunque se trate de un producto software innovador, el proyecto no es de investigación. Simplemente, se desea intentar construirlo y cortar el proyecto en cuanto se observe que el desarrollo del sistema puede sobrepasar los recursos asignados. Dado que el software es nuevo, y que por tanto no se conoce perfectamente el problema, parece recomendable entregar cuanto antes un producto al usuario para confirmar sus necesidades.

1.3 Procesos del ciclo de vida del software

El ciclo de vida del software abarca todas las etapas desde la concepción (idea), hasta la entrega y mantenimiento del software. Además, incluye las fases inherentes a su implementación, como: procesos de acuerdo, procesos de gestión técnica, procesos técnicos y procesos organizacionales que habilitan el proyecto (Ver Figura 1).

Figura 1

Software life Cycle Processes



Nota. La figura 1 fue extraída de……

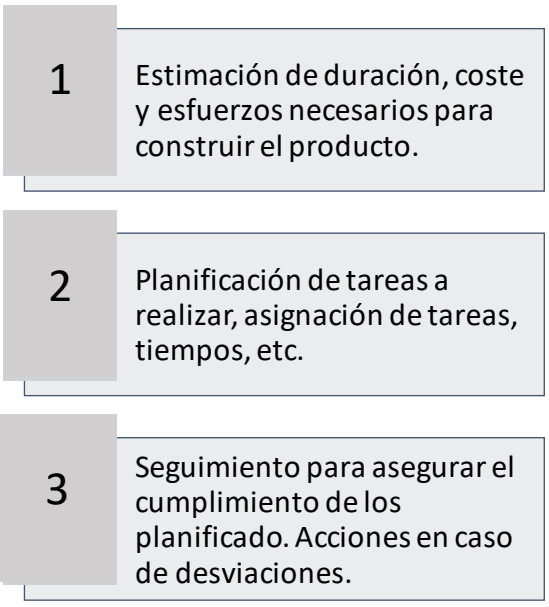
2.1 Gestión Operativa

Es la utilización de las técnicas y actividades de gestión requeridas para conseguir un producto de software de alta calidad de acuerdo a las necesidades de los usuarios, dentro de los presupuestos y con una planificación de tiempos establecidas previamente. Se puede considerar que un proyecto de software es un esfuerzo temporal emprendido para crear un producto o servicio único.

En la materialización de este producto o servicio iniciada por la empresa intervienen un compendio de áreas, como: recursos humanos, financieros y materiales que se organizan de una forma integral para realizar un trabajo genuino y diferenciador al resto de la competencia. Además, al dar algunas especificaciones – límites de coste y tiempo –, se intentan conseguir un cambio beneficioso para la organización, relacionado directamente con sus objetivos cualitativos y cuantitativos (Ver Figura 1).

Figura 1

Tareas críticas en la gestión de proyectos



Fuente: Elaboración Propia.

- **Estimación de proyectos**

La estimación de proyectos es un proceso clave que permite asignar un valor a diversas variables necesarias para la ejecución de un trabajo. Esto incluye la predicción del personal requerido, el esfuerzo necesario, los costos asociados y todos los recursos necesarios para completar las actividades del proyecto. Para lograr una estimación precisa, se hace uso de datos históricos y herramientas especializadas que ayudan a proyectar los posibles escenarios y necesidades del proyecto.

- **Planificación de Proyectos**

La definición de actividades es un proceso fundamental para alcanzar un objetivo, seguido de la correcta asignación de estas tareas a los responsables correspondientes. Mientras que la estimación está enfocada en el proyecto como un todo, la planificación se orienta hacia la distribución de actividades entre los individuos, como podría ser el caso de planificar un viaje para conocer Colombia. El seguimiento de proyectos permite monitorear y evaluar el progreso, asegurando que todo avance conforme a lo planificado.

→ La recolección y almacenamiento de datos sobre tiempos, recursos, costos y hitos asociados con un proyecto son fundamentales para su gestión efectiva. Para analizar y evaluar la evolución real del proyecto, es necesario comparar el progreso alcanzado con lo que se había planificado inicialmente

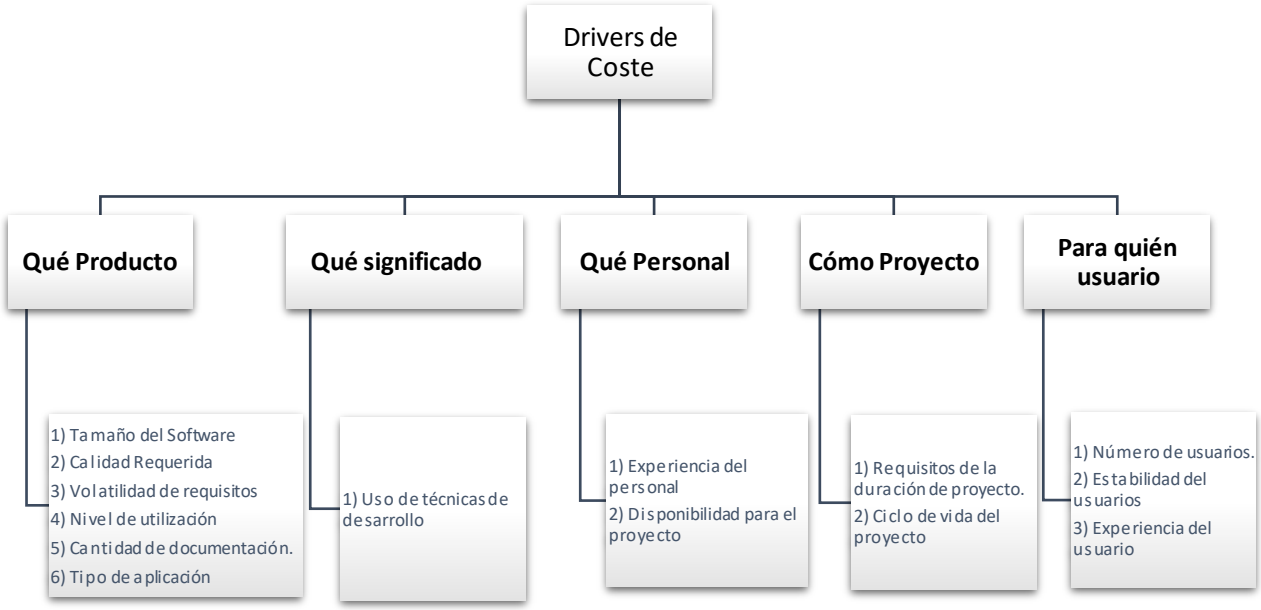
→ Entre los costos que no se registran son: 1) el tiempo extra de profesionales, costos de viajes y reuniones, 3) esfuerzo de los directivos, 4) esfuerzo de los usuarios en participación en tareas técnicas, escritura de manuales, pruebas o participación en revisiones.

2.2 Estimaciones del Software

Una estimación es un conjunto aproximado de valores para algo que ha de ser hecho. En el desarrollo de software se considera la estimación como una actividad que permite obtener principalmente respuestas aproximadas a las siguientes preguntas ¿Cuánto costará? ¿Cuánto tiempo llevará hacerlo?

- **Drivers de Coste**
  - → El producto que se tiene que desarrollar: Qué
  - → El significado de la producción: Con Qué

→ El personal de producción: Con Quién
→ La organización de producción: Cómo
→ Usuario / Organización del usuario: Para Quién



- **El estimador**

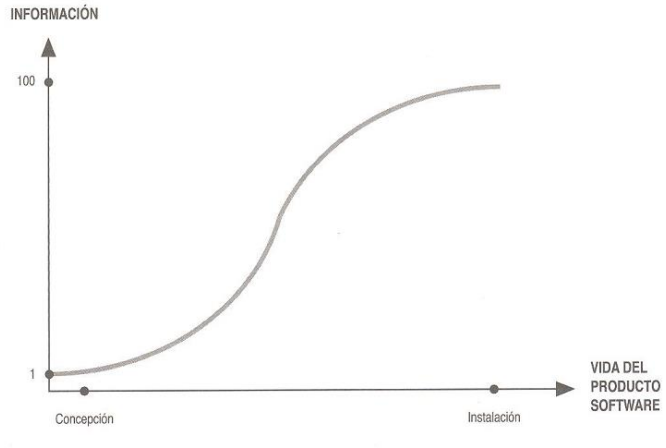
El estimado debe ser un profesional que no tenga ningún interés (directo o indirecto), en los resultados del proceso de estimación y que está guiado únicamente por su profesionalidad. Además, de ofrecer estimaciones de calidad y no coincidir con expectativas de la dirección. Los requisitos que tiene un buen estimador son: 1) Formación y experiencia profesional, 2) Posición que le permita adoptar un juicio independiente, 3) Basarse en un método, 4) Hacer uso de una herramienta que soporte el método, 5) Documentar su estimación.

- **Marco Temporal**

Es un proceso continuo que a medida que más se avanza se conoce sus macro características y micro características (Ver Figura 2).

Figura 2

Características del Marco Temporal



Fuente: Elaboración Propia.

- **Salidas del proceso de estimación**

El costo y el tiempo requerido para ejecutar un proyecto dependerán en gran medida del método elegido para su realización. Cada enfoque puede implicar diferentes recursos, niveles de complejidad y plazos de ejecución:

→ ¿Cuántas personas se necesita?
→ ¿Qué tipo de perfiles?
→ ¿Cuáles son los riesgos?
→ ¿Cómo afectan las restricciones?
→ ¿Cuánto esfuerzo se necesita para cada fase?
→ ¿Cuál será el tamaño en líneas de código?
→ ¿Cuántos defectos tendrá el producto?
→ ¿Cuánta documentación será generada?

- **Técnicas básicas**

  → **La opinión de expertos:** Basado en la experiencia profesional de los participantes.
  → **La analogía:** Se ajusta dependiendo de la diferencia entre el proyecto basado y el nuevo.
  → **La descomposición:** Consiste en la descomposición de un producto en componentes.
  → **Ecuaciones de estimación:** la medida del tamaño del producto y determinan el esfuerzo necesario.

- **Puntos de caso de uso:**

Propuesto originalmente por Gustav Karner y refinado posteriormente por otros autores, este enfoque se basa en la asignación de pesos a un conjunto específico de factores. Se calcula en base a la cantidad de transacciones que se dan en el caso de uso.

Cálculo – PCUSA = FPA + FPCU

  → **PCUSA:** Puntos de caso de uno sin ajustar.
  → **FPA:** Factor de peso de los actores.
  → **FPCU:** Factor de peso de los CU sin ajustar.

Tipo de Actor	Descripción	Factor de Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface)	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3

- **Constructive Cost Model**

El objetivo es desarrollar un modelo para estimar costos y cronogramas acorde a las prácticas de ciclos de vida de las décadas de los 90 y 2000.

Modelo Intermedio:

→ **PREC:** Precedencia.
→ **FLEX:** Flexibilidad del desarrollo.
→ **RESL:** Arquitectura y Resolución de Riesgos.
→ **TEAM:** Cohesión del equipo.
→ **PMAT:** Madurez del proceso.

El Constructive Cost Model incorpora 15 variables de predicción que influyen en el coste del proyecto. Estas se agrupan en cuatro categorías:

→ Atributas del producto software.
→ Atributos del computador.
→ Atributos del personal.
→ Atributos del proyecto.

- **Atributos del producto del Software**

→ **RELY:** Fiabilidad requerida del software. Un rango muy bajo se utiliza cuando el defecto tenga que ser eliminado por los desarrolladores, pero sin ninguna otra consecuencia. Cuando haya posibles pérdidas de vidas humanas es muy alto.
→ **DATA:** Tamaño de la base de datos (impacta en el esfuerzo en las pruebas) Indicador = Tamaño de la BD (bytes) / SLOC. Bajo (menor a 10), 100, muy alto (mayor a 1000).
→ **CPLX:** Complejidad del producto. Puede variar desde muy bajo si el módulo utiliza expresiones matemáticas simples a extra alto con operaciones muy complejas. Desde simples generadores de reportes hasta multimedia compleja y realidad virtual, desde manejo simple de arreglos hasta coordinación de bases de datos distribuidas.
→ **RUSE:** Esfuerzo adicional para construir componentes reutilizables. Desde bajo (ningún esfuerzo) hasta extra alto (reusabilidad a través de múltiples líneas de productos).
→ **DOCU:** Necesidades de Documentación para el ciclo de vida seleccionado, muy bajo (muchas fases no necesitan ser cubiertas), hasta muy alto (necesidades excesivas de documentación para el ciclo de vida).

- **Atributos del computador**

  → **TIME:** Limitaciones en el tiempo de ejecución. Se expresa por la relación entre el porcentaje del tiempo de ejecución que se espera utilice el producto y otros productos que comparte con él. Nominal cuando su porcentaje es 50% y extra alto cuando llega al 95%.

  → **STOR:** Limitaciones de memoria principal. Porcentaje de memoria que se espera use el producto con respecto a otros. Nominal 50% hasta Extra Alto 95%

  → **PVOL:** Volatilidad de la plataforma. Se entiende por plataforma al conjunto de hardware y software que el producto utiliza para realizar su tarea. Durante el desarrollo esta máquina puede cambiar. Bajo (cambios pequeños cada mes, cambios grandes cada 12 meses), muy alto (cambios menores cada 2 días, cambios mayores cada 2 semanas)

  → **TURN:** Señala el tiempo de respuesta experimentado desde que el desarrollador introduce un trabajo hasta que obtiene los resultados. Este parámetro ha perdido importancia. Bajo para un sistema interactivo hasta muy alto cuando el tiempo de respuesta es mayor a 12 horas. X

- **Atributos del personal**

  → **ACAP** (Capacidad de los analistas): Habilidades para el análisis, Eficiencia y calidad en el trabajo y Habilidad para comunicarse y cooperar.

  → **PCAP** (Capacidad de los programadores): Similares atributos que el parámetro PCAP, pero para los programadores.

  → **AEXP** (Experiencia en aplicaciones por parte del equipo): Indica el nivel de experiencia en aplicaciones del equipo, varia desde muy bajo (menos de 2 meses de experiencia), nominal (1 año) y muy alto (más de 6 años).

  → **PEXP** (Experiencia en la plataforma): Es el tiempo de experiencia en el entorno hardware y software del equipo que desarrolla. Muy bajo (dos meses), Nominal (1 año), Alto (tres años), Muy alto (6 años). No se considera el lenguaje de programación.

  → **LTEX** (Experiencia en el lenguaje de programación y herramienta): Puede variar desde muy bajo (menos de dos meses), nominal (1 año), hasta alto (tres años de experiencia), muy alto (6 años).

  → **PCON** (Estabilidad del personal): muy bajo (48% de cambios), bajo (24% cambios), nominal (12%), alto (6%), muy alto (3% de cambios).

- **Atributos del Proyecto**

→ **TOOL**: Uso de herramientas para el desarrollo software. El rango varía entre muy bajo cuando solo usa herramientas básicas (edición y depuración), nominal (básicas para el ciclo de vida, moderadamente integrado), hasta muy alto (Bien integradas con los procesos, métodos, reusó).

→ **SCED**: Limitaciones en la planificación. Se define mediante el porcentaje de retraso o aceleración con respecto a la planificación impuesta al equipo de desarrollo. Cualquier aceleración (muy bajo) 75%, nominal (100%), retraso (muy alto) 160% requerirá mayor esfuerzo.

→ **SITE**: Desarrollo multisitio: muy bajo (internacional); multiciudad o multicompañía (nominal), mismo edificio (muy alto).

→ **MODP**: Prácticas modernas de programación (antiguo). Se valora el grado de uso de las siguientes prácticas: Análisis de requisitos, diseño estructurado, desarrollo incremental, revisiones, inspecciones, peer-review, librerías.

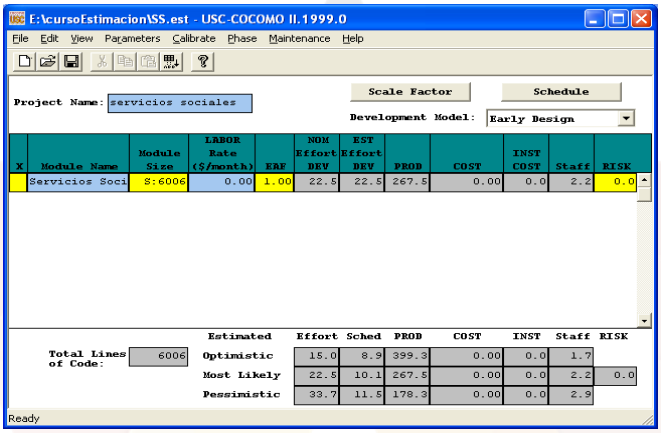
- **Multiplicadores de esfuerzo**

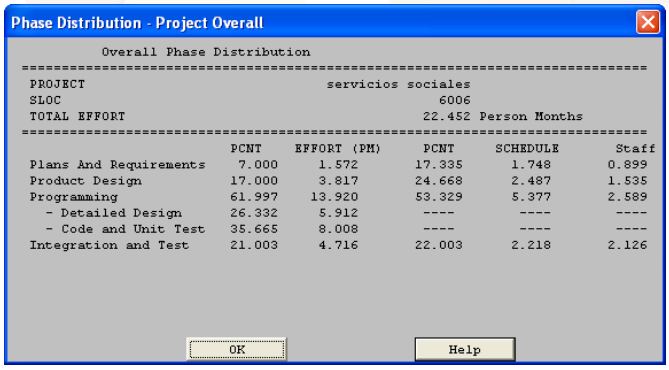
DISEÑO PRELI	POST ARQUITECTURA
RCPX	RELY, DATA CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PERS	ACAP, PCAP, PCON
PREX	AEXP, PEXP, LTEX
FCIL	TOOL, SITE
SCED	SCED

Ejercicio 1:

Ejercicio en clases

Práctica de estimación con COCOMO





2.3 Ingeniería de Requerimientos

La mayor dificultad en la construcción de un sistema de software radica en definir con precisión qué es lo que se debe desarrollar. No existe tarea que pueda causar más daño al sistema cuando se ejecuta incorrectamente, ni otra tan complicada de corregir una vez realizada

- **El Problema de la IR**

Según Boehm (1975), el 45% de los errores en proyectos de software se originan en la etapa de requisitos y diseño preliminar, mientras que DeMarco (1984) señala que el 56% de los errores se deben a una mala especificación de requisitos. El Chaos Report destaca que los principales factores que conducen al fracaso de proyectos de software son:

→ Falta de comunicación con los usuarios.
→ La presencia de requisitos incompletos
→ Los cambios en los requisitos.

Si el sistema final no satisface a los usuarios, esto generará desacuerdos entre ellos y los desarrolladores. Además, puede volverse imposible demostrar si el software cumple o no con los requisitos establecidos. Entre las consecuencias más importantes se encuentran:

→ Se gastará tiempo y dinero en construir el sistema equivocado.
→ Se realizará una estimación incorrecta creando falsas expectativas en los usuarios.
→ Los errores en las especificaciones de requisitos son lo más caros de corregir.

- **¿Por qué es importante la Ingeniería de Requisitos?**

Para mitigar esta situación, surge la Ingeniería de Requisitos (IR), que se enfoca en los principios, métodos, técnicas y herramientas destinadas a descubrir, documentar y mantener los requisitos de sistemas basados en computadora de manera sistemática y repetible. Los documentos de requisitos contienen información clave para este proceso. Un documento de requisitos contiene los siguientes elementos:

→ Información acerca del problema.
→ Propiedad y comportamiento del sistema.
→ Restricciones de diseño y fabricación del producto.
→ Descripciones acerca de cómo el futuro sistema ayudará a sus usuarios a realizar mejor sus tareas.

→ Restricciones acerca de la tecnología que será utilizada en la construcción del sistema.
→ Restricciones acerca de las propiedades emergentes del sistema (requisitos no funcionales).

- **Escritura de requisitos**

En realidad, no existe una forma única de escribir requisitos, aunque el uso del lenguaje natural es lo más recomendable. Cada requisito debe expresarse de manera concisa, utilizando frases cortas como: "el sistema realizará X" o "se proporcionará Y". Por otro lado, el lenguaje natural, complementado con diagramas y/o notaciones formales, es una opción efectiva. La notación elegida dependerá de quién redacta los requisitos como de quién los interpreta. A continuación, se presenta algunos ejemplos de requisitos:

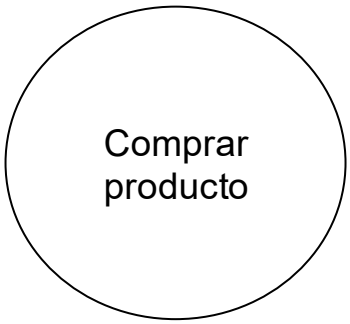
1	El sistema permitirá mantener la temperatura de la caldera entre 70º y 80º.
2	El sistema permitirá a los usuarios realizar una búsqueda por título, autor o ISBN.
3	El interfaz de usuario se implementará sobre un navegador Web.
4	El sistema deberá soportar al menos 20 transacciones por segundo.
5	El sistema permitirá que los nuevos usuarios se familiaricen con su uso en menos de 15 minutos.

- **Requisitos funcionales y no funcionales**

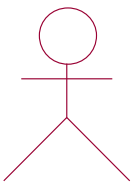
Los requisitos funcionales definen los servicios o funciones que el sistema debe proporcionar como, por ejemplo, “el sistema aceptará pagos con VISA”. Por otro lado, los requisitos no funcionales imponen restricciones sobre los requisitos funcionales, especificando aspectos adicionales como el rendimiento o la seguridad. Un ejemplo de requisito no funcional sería “el sistema aceptará pagos con VISA de forma segura y con un tiempo de respuesta inferior a 5 segundos”.

- **Diagramas de casos de uso**

  → Representa una unidad de funcionalidad de un sistema.
  → Son descripciones narrativas de los procesos.
  → No son específicamente las funcionalidades ni los requerimientos.

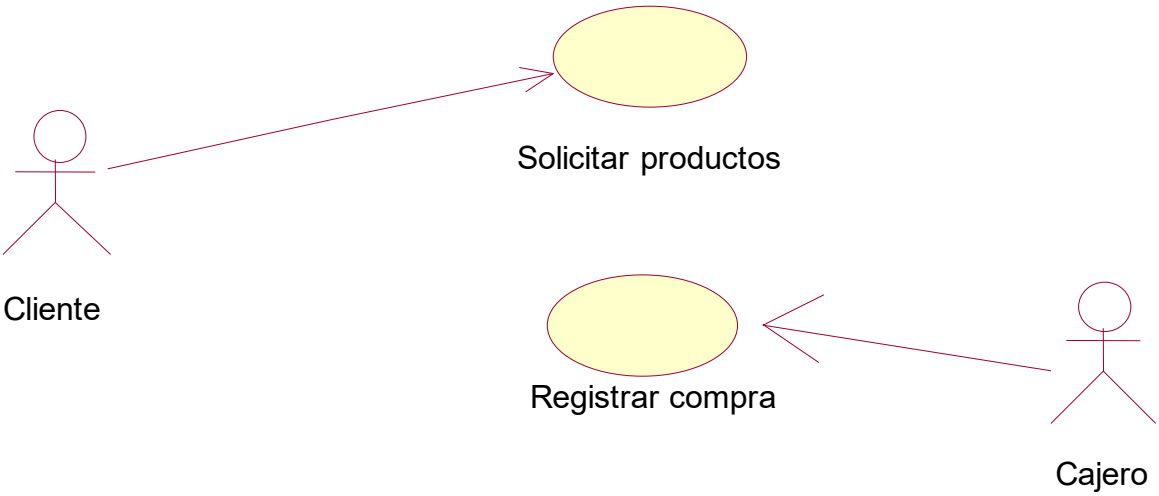


- **Actores**
  → Es algo o alguien que interactúa con el sistema.
  → Es externo al sistema.
  → Existen los actores iniciadores de los casos de uso y los actores participantes.
  → Pueden ser, por ejemplo: papeles que desempeñan las personas, sistema de cómputo.



- **Diagrama I**

Representa gráficamente un conjunto de casos de uso, mostrando los actores involucrados y la relación entre ellos. Además, las flechas indicarán el flujo de información, proporcionando una visión general del sistema.



- **Formatos**
  - → Los casos de uso se pueden expresar en diversos formatos.
  - → Los casos de uso se pueden expresar en diversos grados de detalle, como: 1) Formato de alto nivel, 2) Formato expandido.

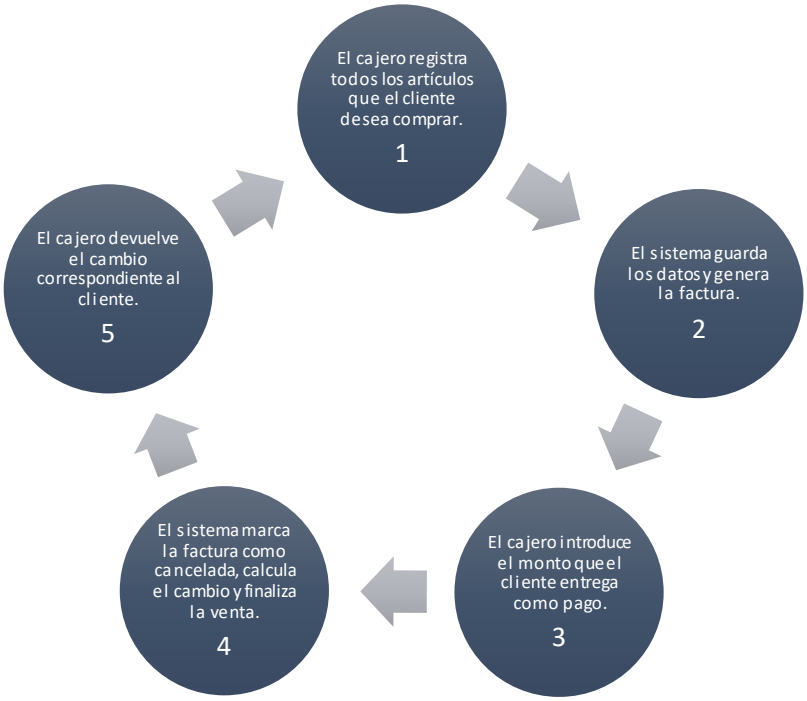
- **Casos de uso: Formato expandido I**

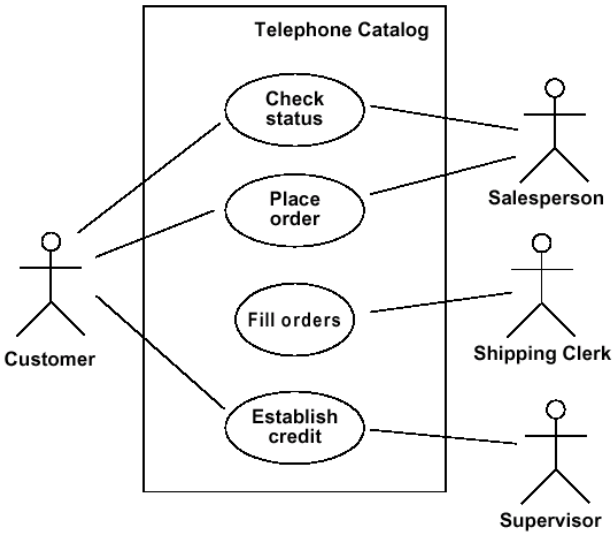
Útiles para alcanzar un conocimiento más profundo de los procesos y la funcionalidad.

- → **Caso de uso:** Registrar Compra.
- → **Actor:** Cajero.

→ **Objetivo:** Registrar compra de un producto.

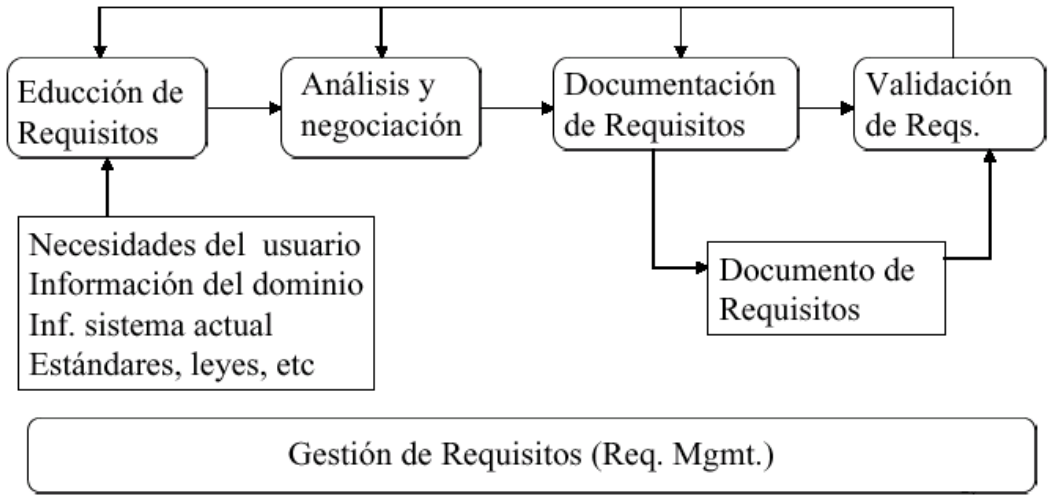
- **Curso normal de eventos:**





- **Relaciones**
  - → Asociación: Es la participación de un actor en un caso de uso.
  - → Generalización: B es una especialización de C.
  - → Inclusión: El caso de uso B está incluido en el caso de uso C.

El proceso de los requisitos



- **Los requisitos pueden proceder de:**
  - → Metas: Factores críticos de éxito.
  - → Conocimiento del dominio de la aplicación.
  - → Los afectados por el sistema.
  - → El entorno físico y organizacional.

- **Técnicas de educción.**
  - → Entrevistas: Preparar preguntas y hacerlas con las personas adecuadas.
  - → Brainstorming: Recoger por escrito un gran número de ideas y luego evaluarlas.
  - → Observación y análisis de tareas (posibilidades de mejora).
  - → Prototipado: Cuando la incertidumbre es total acerca del nuevo sistema.

- **Análisis de requisitos**
  - → Detectar y resolver conflictos entre requisitos.

→ Se pone los límites al sistema y la interacción con el entorno.
→ Se clasifica, negocia, prioriza los requisitos.

- **Clasificación y negociación de requisitos**

La clasificación de requisitos puede realizarse de varias formas: En primer lugar, se pueden dividir en funcionales (aquellas que describen lo que el sistema debe hacer) y no funcionales (aquellos que describen cómo debe hacerlo. Además, existen otros factores que influyen directamente, como: 1) Prioridades, 2) Coste de implementación, 3) Acuerdo a su estabilidad, 4) Casos de uso, 5) Volatibilidad/estabilidad.

Por otro lado, la negociación de requisitos es un proceso clave dentro del proceso de ingeniería de requisitos (IR), en el que participan distintos individuos con intereses a veces opuesto. Cuando surgen conflictos, estos activan un proceso de negociación para llegar a un acuerdo. Resulta importante entender que los conflictos no deben ser rechazados ni resueltos por imposición, ya que son una fuente de nuevos requisitos que contribuyen al enriquecimiento del proyecto. Los documentos que son necesarios para su elaboración son:

→ Es un medio de comunicar requisitos.
→ Casos de uso: Describir los procesos y metas a alcanzar
→ ERS: Responde a la pregunta ¿Qué características debe poseer el sistema para alcanzar los objetivos?
→ Procesador de textos, Bases de Datos y Rational RequisitePro.

- **Esquema IEEE 830**

El esquema IEE 830 proporciona guías para la creación de un Documento de Especificación de Requisitos del Software (SRS, por sus siglas en inglés). Este esquema ayuda a estructurar de forma integral y ordenada los requisitos del software, facilitando la comunicación entre los desarrolladores, clientes y otros interesados. Dentro de las características principales se encuentran:

Características Principales ERS
• No Ambigua • Compleja • Correcta • Comprensible • Verificable • Realizable • No redundante • Precisa • Concisa

Características Principales ERS
• Independiente del diseño • Trazable • Electrónicamente almacenada • Anotada por importancia • Anotada por estabilidad • Anotada por versión • Trazada • Organizada • Con referencias cruzadas

- **Validación de Requisitos**

El objetivo es descubrir los problemas que presenta el documento antes de proceder con su implementación. Para ello, se realiza una revisión detallada para detectar las siguientes características:

→ Ambigüedades
→ Conflictos
→ Omisiones

Uno de los métodos más efectivos para detectar problemas es llevar a cabo revisiones grupales con los usuarios mediante reuniones y acuerdos, así como utilizar una lista de verificación para identificar problemas habituales.

Métodos secundarios

→ Prototipado: Permite identificar rápidamente si el usuario está satisfecho con los requisitos documentados.
→ Validación de la "testeabilidad": Esta revisión es realizada por el equipo de pruebas.

- **Gestión de Requisitos**

La gestión de cambios de los requisitos es fundamental para asegurar la consistencia entre los requisitos y el sistema que se está construyendo o que ya se ha construido. Sin embargo, este proceso consume una gran cantidad de tiempo y esfuerzo, ya que abarca todo el ciclo de vida del producto.

Ejercicio 2:

Ejercicio en clases

Escribir los requisitos funcionales y no funcionales del siguiente proyecto.

Los servicios públicos de salud locales almacenan los datos de consultas de pacientes en hospitales y centros de salud. Actualmente, esta información se utiliza exclusivamente con fines estadísticos. Sin embargo, los hospitales y centros de salud asignan distintos números de historia clínica a los pacientes, lo que implica la ausencia de un identificador único para cada individuo.

El Ministerio de Salud tiene como objetivo implementar un sistema unificado para establecer una base de datos centralizada de historias clínicas. Esta base de datos proporcionará a hospitales y centros de salud acceso inmediato a los historiales médicos de los pacientes, facilitando la toma de decisiones médicas sin demoras relacionadas con la transferencia de registros. El sistema está diseñado para migrar los datos existentes a una nueva base de datos histórica. Una vez desplegado, se espera que todos los pacientes reciban un único número de historia clínica que pueda ser utilizado en cualquier centro médico u hospital del país.

El proyecto tiene un plazo de dos años para su implementación completa y un presupuesto asignado de 4 millones de dólares estadounidenses. Los requisitos del sistema ya han sido definidos en un proyecto previo de ingeniería de requerimientos. El desarrollo se llevará a cabo utilizando la técnica *Test Driven Development* (TDD). El equipo encargado está compuesto por desarrolladores junior, con un máximo de dos años de experiencia en programación y sin experiencia previa en TDD.

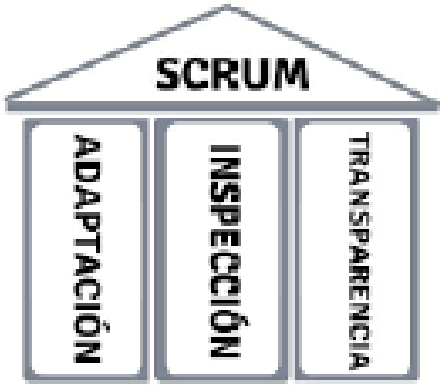
Es necesario evaluar los niveles de integridad y seguridad del sistema, ya que existen varios riesgos inherentes. Uno de los principales riesgos es que los datos migrados desde los sistemas antiguos no sean correctamente identificados y asignados a los pacientes correctos, o que no se vinculen adecuadamente los historiales previos de los pacientes en el nuevo sistema. Esto podría ocasionar diagnósticos o tratamientos incorrectos debido a la falta de acceso completo a la información médica del paciente.

2.4 Desarrollo ágil del software

La metodología Scrum es un marco de trabajo ágil que se utiliza principalmente en el desarrollo de software, aunque su aplicación se ha extendido a otros campos como la gestión de proyectos, marketing, investigación, entre otros. Se basa en principios de transparencia, inspección y adaptación para permitir a los equipos enfrentar de manera efectiva la complejidad y la incertidumbre en proyectos. En este documento, exploraremos todos los aspectos esenciales de la metodología Scrum, desde sus principios fundamentales hasta sus roles, eventos y artefactos.

- **Principios Fundamentales**

Los principios fundamentales de Scrum son cruciales para su implementación efectiva, y se sustentan en tres pilares: la transparencia, que garantiza que todos los aspectos del proceso sean accesibles y visibles para todos los participantes en el proyecto; la inspección, que conlleva la revisión regular de los artefactos y del progreso, con el fin de detectar desviaciones o problemas; y la adaptación, que faculta al equipo para ajustar y modificar su enfoque en función de los hallazgos derivados de la inspección, promoviendo así una mejora continua en el desarrollo del proyecto.

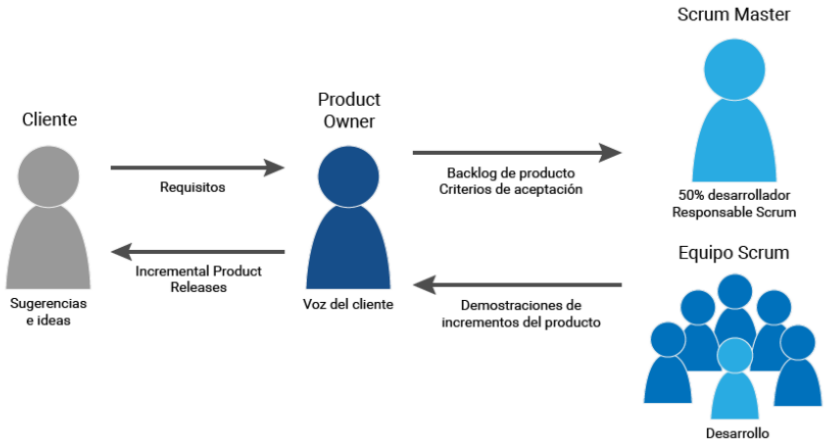


- **Roles de Scrum**

→ **Product Owner:** Es responsable de definir y priorizar los requisitos del producto, mantener una visión clara del producto y asegurarse de que el equipo de desarrollo esté trabajando en las tareas más valiosas en todo momento. Además, el Product Owner es el encargado de tomar decisiones sobre qué características se incluirán en cada iteración del desarrollo y de proporcionar retroalimentación continua al equipo, gestionar el Product Backlog.

→ **Scrum Master:** Es responsable de garantizar que el equipo de desarrollo siga los principios y prácticas de Scrum. Su rol incluye eliminar obstáculos que puedan afectar la productividad del equipo, facilitar reuniones y eventos de Scrum, como las reuniones diarias, la planificación de sprint y la retrospectiva, y también servir como un mentor para el equipo, ayudándolos a comprender y adoptar correctamente los valores y principios de Scrum. El Scrum Master también actúa como un facilitador de procesos, fomentando la colaboración y la autoorganización dentro del equipo. Facilita el proceso Scrum, elimina obstáculos y promueve un ambiente de trabajo productivo.

→ **Equipo de Desarrollo**: El equipo de desarrollo en Scrum es responsable de convertir los requisitos del producto en incrementos potencialmente entregables de funcionalidad al final de cada sprint. Esto implica planificar y llevar a cabo el trabajo necesario para alcanzar los objetivos del sprint, colaborar estrechamente con el Product Owner para entender los requisitos del producto y con el Scrum Master para asegurar que el proceso de Scrum se siga correctamente. El equipo de desarrollo es autoorganizado y multifuncional, lo que significa que tiene todas las habilidades necesarias para completar el trabajo dentro del sprint. Además, el equipo de desarrollo se esfuerza por mejorar continuamente su desempeño y la calidad del producto entregado. Son autoorganizados y multifuncionales, responsables de convertir los elementos del Product Backlog en incrementos del producto.



- **Eventos de Scrum**

→ **Sprint:** Es un periodo de tiempo fijo, usualmente de 2 a 4 semanas, en el que se desarrolla un incremento del producto.

→ **Reunión de Planificación del Sprint: Se** realiza al inicio de cada sprint para seleccionar elementos del Product Backlog y definir el objetivo del sprint.

→ **Daily Scrum:** Una breve reunión diaria donde el equipo de desarrollo sincroniza sus actividades y planea el trabajo para las próximas 24 horas.

→ **Revisión del Sprint:** Al finalizar el sprint, se revisa el incremento del producto y se obtiene retroalimentación de los stakeholders.

- **Artefactos**

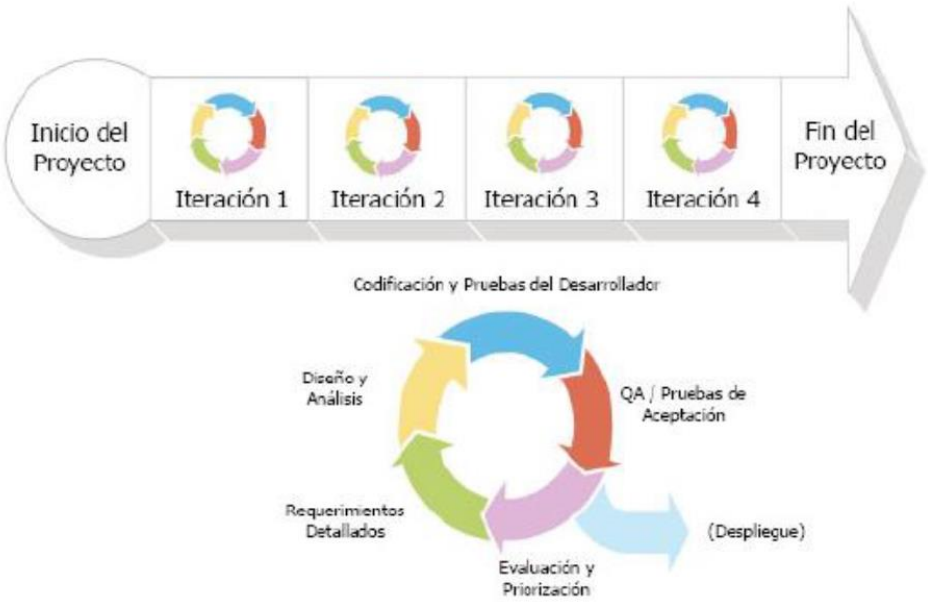
  → **Product Backlog:** Es una lista priorizada de todas las funcionalidades, cambios y mejoras que se desean en el producto.

  → **Sprint Backlog:** Lista de elementos del Product Backlog seleccionados para ser completados durante el sprint.

  → **Incremento del Producto:** El resultado tangible y potencialmente entregable al final de cada sprint.

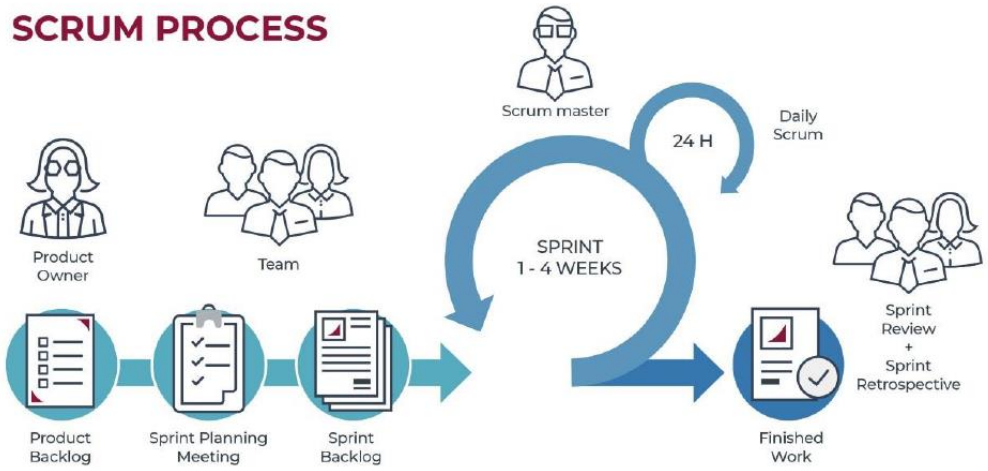
- **Implementación:**

  → **Inicio:** Se forma el equipo Scrum y se establecen los roles y responsabilidades.
  → **Planificación:** Se crea el Product Backlog y se realiza la primera reunión de planificación del sprint.
  → **Ejecución:** El equipo trabaja en los elementos del Sprint Backlog para completar el incremento del producto.
  → **Revisión y Retroalimentación:** Se lleva a cabo la revisión y retrospectiva del sprint para evaluar el progreso y hacer mejoras.
  → **Iteración:** Se repiten los Sprints, adaptando y mejorando continuamente el proceso y el producto.



- **Beneficios:**

  → **Flexibilidad:** Scrum facilita la adaptación rápida a los cambios en los requisitos o en el entorno del proyecto.
  → **Transparencia:** Todos los aspectos del proyecto son visibles y comprensibles para todos los involucrados.
  → **Entrega Continua:** Los incrementos del producto se entregan de manera regular y frecuente, lo que permite obtener retroalimentación temprana y valiosa.



La metodología Scrum proporciona un enfoque flexible y colaborativo para la gestión de proyectos, particularmente en entornos complejos y dinámicos. Al enfocarse en la entrega incremental de valor y en la mejora continua, Scrum capacita a los equipos para adaptarse rápidamente a las demandas del mercado y atender de manera efectiva las necesidades de los stakeholders.

3.1 Calidad del Software

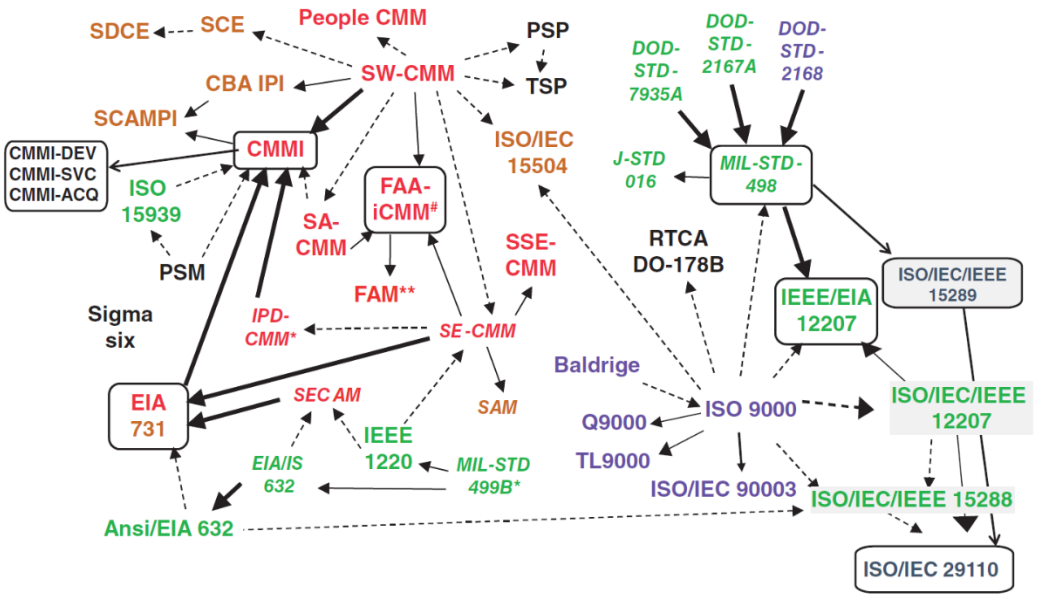
La evaluación y mejora del proceso de desarrollo de software son estrategias fundamentales para elevar la calidad del software producido. Estas prácticas surgieron a finales de los años 80 como respuesta a la crisis del software, marcada por problemas endémicos como el aumento del tamaño y la complejidad de los proyectos. La evaluación del proceso de software se inspira en métodos aplicados en otras disciplinas, donde la gestión de procesos garantiza la calidad en la construcción de productos.

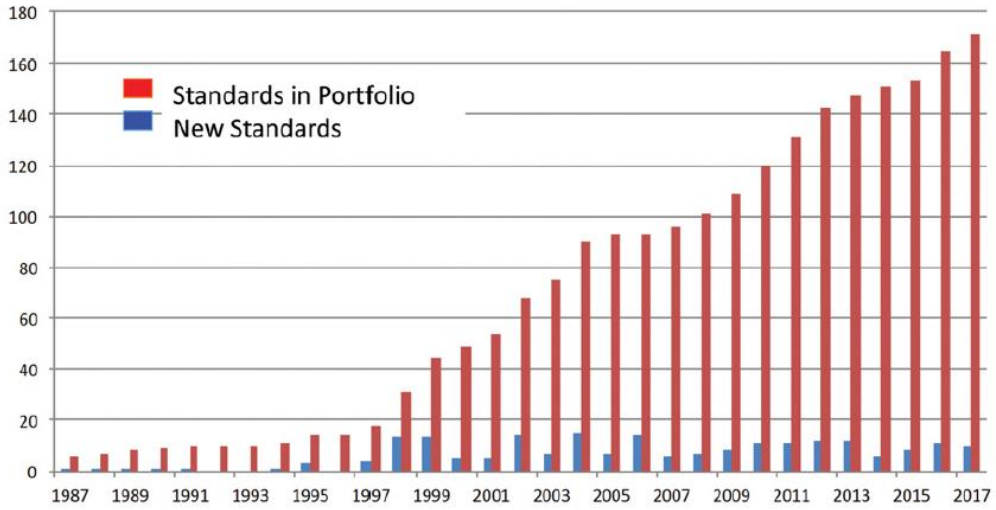
La calidad del producto software está directamente vinculada a la calidad del proceso de desarrollo. Por ello, no es posible establecer una estrategia de mejora efectiva sin primero conocer la situación actual de las actividades involucradas en el desarrollo del software. Sin una evaluación adecuada, las mejoras implementadas podrían resultar ineficaces o, incluso, generar consecuencias negativas.

El proceso de mejora del desarrollo de software se basa en dos componentes interrelacionados: la evaluación y la optimización. La evaluación permite obtener una certificación o acreditación del estado actual del proceso de software en una organización. Este enfoque auditor asume que los resultados de la evaluación reflejarán la calidad de los productos finales. Además, la certificación suele ser un requisito clave en la selección de proveedores dentro de la industria.

- **Procesos estándares**

  → Principios para el desarrollo de estándares ISO.
  → Las normas ISO satisfacen la necesidad del mercado.
  → Son badsadas en la experiencia a nivel global.
  → Son el resultado de un proceso en el que intervienen múltiples partes interesadas.
  → Las normas ISO se basan en el consenso.





- **Principales estándares**

→ ISO 12207 (Framework): estándar para el ciclo de la vida del software.
→ Arquitectura de los procesos del ciclo de vida del software (procesos, actividades, tareas).
→ Estándar en la familia ISO 9000: 1) ISO 9000:2025 – Conceptos básicos y lenguaje, 2) ISO 9001:2015 – Sistema de gestión de calidad y requisitos.
→ ISO/IEC 0993 – Guía para la aplicación de ISO 9001 al software.
→ ISO/IEC 15504 – ISO/IEC 33001.

→ Evaluación del proceso software.
→ ISO/IEC 29110: Perfiles de ciclo de vida para pequeñas organizaciones.
→ ISO 29110: Organizaciones pequeñas o departamentos (hasta 25 personas).

En Europa más del 92% de las organizaciones tienen entre 1 a 9 empleados. Por otro lado, en Canadá el 80% de las organizaciones tienen entre 25 empleados o más (depende de la organización) (Ver Tabla 1).

Tabla 1

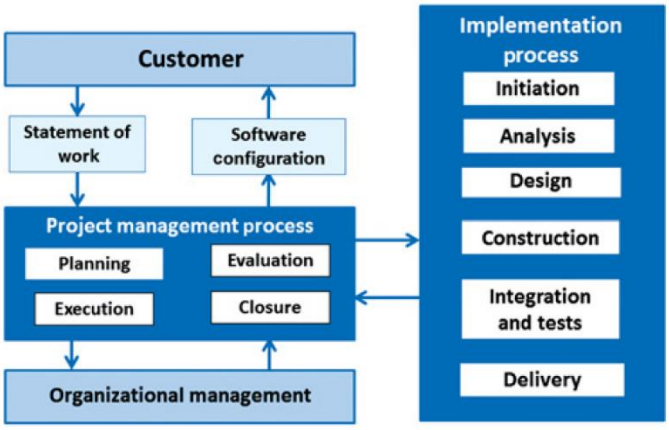
Descripción de las organizaciones

PERFIL	DESCRIPCIÓN
Entrada	Este perfil es para organizaciones que trabajan en proyectos pequeños (ejemplo proyecto de hasta 6 personas/mes) y emprendimientos.
Básico	Perfil para organizaciones que desarrollan solo un proyecto a la vez con uno solo equipo.
Intermedio	Perfil para organizaciones que realizan desarrollo simultáneo de más de un proyecto con más de un equipo.
Avanzado	Este perfil es para organizaciones que desean mejorar significativamente la gestión de sus negocios y su competitividad.

Fuente: Elaboración Propia.

Figura 1

Gestión e implementación del proceso para el perfil básico



A continuación, se presenta un ejemplo de la descripción de una tarea en la actividad de análisis de requerimientos (Ver Tabla 2).

Tabla 2

Descripción de actividades

ROLES	TAREA	PRODUCTOS DE ENTRADA	PRODUCTOS DE SALIDA
CLI AN	SI.2.4 Validar y obtener la aprobación de la Especificación de Requisitos Validar que la especificación de requisitos satisfaga las necesidades y las expectativas acordadas, incluyendo la usabilidad de la interface de usuario. Los resultados de la validación son documentados y se hacen las correcciones hasta que el documento sea aprobado por el Cliente.	Especificación de Requisitos - Verificada	Resultados de la validación Especificación de Requisitos – Validada

Fuente: Elaboración Propia.

- **Casos de estudio: Peruvian Startup**

La organización ha implementado prácticas ágiles, como Scrum, TDD e integración continua, adaptando el perfil básico a estas metodologías ya establecidas. Los nuevos procesos se aplican actualmente en el desarrollo de un proyecto de software para una compañía de seguros, con un esfuerzo total estimado de 900 horas/persona.

Development phase	Prevention (hours)	Execution (hours)	Evaluation (hours)	Correction (hours)
Preparing the environment (e.g., server)	14			
Developing the project plan		15	3	7
Implementation and project control		108		
Implementation (sprints)		90		
Assessment and control (sprint review)		18		
Software specification		107	28	58
Statement of work		12	3	7
Specification of use cases		95	25	51
Architecture design		35	18	14
Development of the test plan		45	8	11
Coding and testing		253	70	62
Documentation of maintenance and operating guide		14	5	7
Software implementation		6		
Project closure		2		
Total (hours)	**14**	**585**	**124**	**159**

El porcentaje de esfuerzo invertido en la corrección de defectos (retrabajo) fue del 18%, equivalente a 159 horas de un total de 882 horas de trabajo. En cuanto a los niveles de madurez

del Capability Maturity Model (CMM), se observa que el esfuerzo dedicado a la corrección de errores disminuye a medida que se alcanza un mayor nivel de madurez en los procesos, optimizando la calidad del software desde las fases iniciales del desarrollo (Ver Tabla 3).

Tabla 3

Niveles de madurez CMM y esfuerzo de corrección

CMMI nivel de madurez	% esfuerzo en la corrección
1	23,2%
2	14,3%
3	9.5%
4	6.8%

Fuente: Elaboración Propia.

- **Capability Maturity Model Integration (CMMI)**

CMMI es un modelo de evaluación y mejora para los procesos de desarrollo, mantenimiento y operación de los sistemas de software. Asimismo, se debe definir una estrategia estructurada que especifique las actividades a llevar a cabo y su secuencia de implementación, garantizando que la mejora sea eficaz y factible. Esta estrategia debe contemplar las fases de crisis, control del proceso, estandarización, medición y optimización. Los niveles de madurez permiten caracterizar el avance de la organización en relación con un conjunto de áreas de proceso, mientras que los niveles de capacidad se enfocan en la mejora de un área de proceso individual (Ver Tabla 4).

Tabla 4

Niveles de Madurez

Niveles	Descripción
INICIAL	Los procesos son, en su mayoría, ad-hoc y caóticos. El éxito de la organización depende más de los esfuerzos extraordinarios del personal que de la aplicación de procesos establecidos y comprobados. Aunque la organización logra entregar productos y servicios funcionales, a menudo supera los presupuestos y los plazos previstos.
GESTIONADO	En los proyectos se asegura que los procesos sean planificados y ejecutados conforme a las políticas establecidas. Las partes interesadas relevantes son involucradas, y los proyectos se desarrollan y gestionan siguiendo los planes documentados. Los compromisos entre las partes interesadas se establecen y ajustan cuando es necesario para garantizar el cumplimiento de los objetivos.
DEFINIDO	Los procesos están bien caracterizados y comprendidos, y se documentan en estándares, procedimientos, herramientas y métodos. En el nivel de madurez 3, los estándares y descripciones de procesos, así como los procedimientos para un proyecto, se adaptan a partir de un conjunto de procesos estándar de la organización. Esta adaptación permite que los procesos sean más consistentes y se adecuen a las necesidades específicas de cada proyecto.
GESTIONADO CUANTITATIVAMENTE	La organización y los proyectos establecen objetivos cuantitativos para la calidad y el rendimiento de los procesos. Las predicciones se realizan a partir de análisis estadísticos de datos detallados sobre el proceso.
OPTIMIZACIÓN	La organización mejora continuamente sus procesos, fundamentándose en una comprensión cuantitativa de sus objetivos comerciales y necesidades de rendimiento. Se enfoca en optimizar el desempeño de los procesos a través de mejoras incrementales e innovadoras, tanto en los procesos como en la tecnología utilizada.

Fuente: Elaboración Propia.

- **La Madurez**

Se trata de una forma de predecir los resultados de proyectos futuros de la organización. A medida que la organización alcanza las metas establecidas en las distintas áreas de proceso, su madurez organizativa aumenta. Este enfoque contempla dos rutas para implementar mejoras y medir el grado de evolución, las cuales se conocen como 'representaciones'.

→ **Representación Continua:** Se consideran áreas de proceso de manera individual y se califican en niveles de capacidad. La organización elige el enfoque de sus esfuerzos de mejora de procesos, eligiendo aquellas áreas de proceso, o conjuntos de áreas de proceso interrelacionados, que más benefician a la organización y a sus objetivos de negocio.

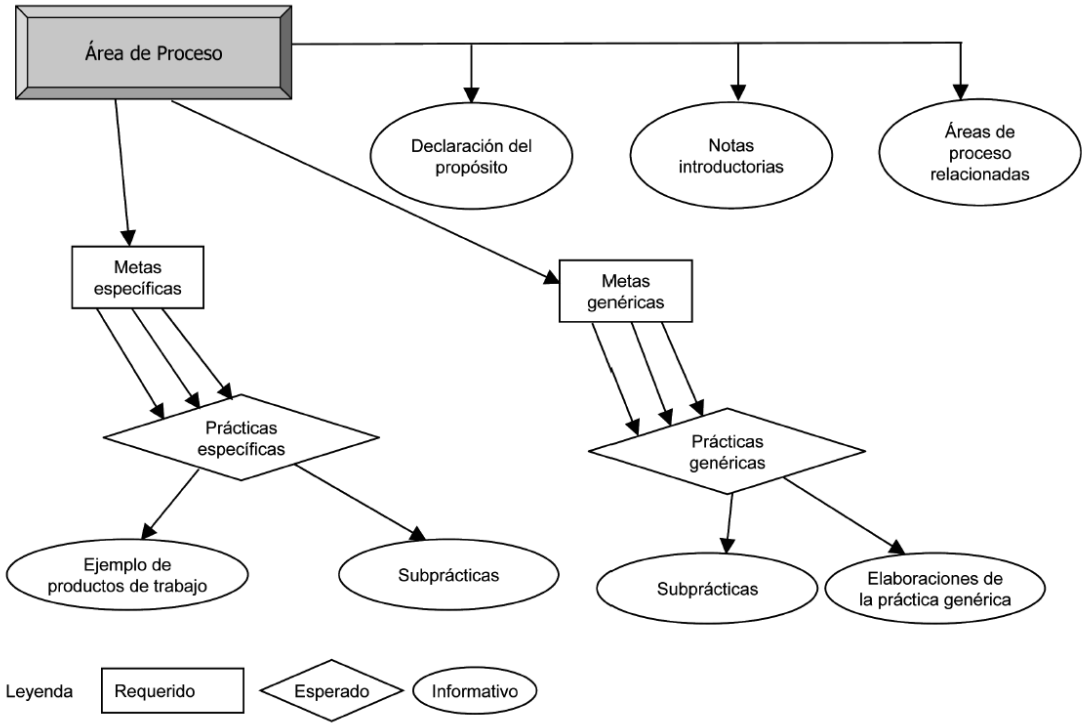
→ **Representación por etapas:** Considera un conjunto preestablecido de áreas de proceso que constituyen un nivel de madurez.

Una organización debe satisfacer todas las metas del área de un proceso o del conjunto de áreas de proceso que son objeto de mejora.

NIVEL DE MADUREZ	ÁREAS DE PROCESO
(1)Inicial **Los procesos son generalmente ad hoc y caóticos**	Basado en la competencia y acciones individuales de las personas
(2) Gestionado **Los proyectos se planifican y ejecutan de acuerdo con las políticas. Los proyectos se realizan y gestionan de acuerdo a sus planes documentados**	Gestión de Configuración Medición y Análisis Monitorización y Control de Proyecto Planificación del Proyecto Aseguramiento de la Calidad del Proceso y del Producto, Gestión de los Requisitos Gestión de acuerdos con proveedores
(3) Definido **Los procesos están bien caracterizados y comprendidos, y se describen en estándares, procedimientos, herramientas y métodos.**	Análisis de Decisiones y Resoluciones Gestión Integrada del Proyecto Definición de Procesos de la Organización Enfoque en Procesos de la Organización Formación en la Organización, Integración de Producto, Desarrollo de Requisitos, Gestión de Riesgos, Solución Técnica, Validación, Verificación
(4) Gestionado Cuantitativa **La organización y los proyectos establecen objetivos cuantitativos**	Rendimiento de Procesos de la Organización Gestión Cuantitativa del Proyecto
(5) Optimizado **Enfoque en la mejora del proceso**	Análisis Casual y Resolución Gestión del Rendimiento de la Organización

- **Área de proceso**

CMMI se centra en un grupo de prácticas que se realizan colectivamente con el objetivo de alcanzar metas específicas; sin embargo, las áreas de proceso en las que se enfoca no son las únicas que deben ejecutarse en la organización, ya que CMMI no describe todos los procesos implicados en el desarrollo y mantenimiento del software



EJEMPLO PRÁCTICO

1). Declaraciones del propósito: Describe la finalidad de un área del proceso

El propósito de Medición y Análisis (MA) es desarrollar y mantener la capacidad de medición utilizada para dar soporte a las necesidades de información de la gerencia.

2). Notas introductorias: Describe los conceptos principales cubiertos por el área de proceso.

Los objetivos de medición se derivan de las necesidades de información que provienen de los objetivos del proyecto, de la organización o del negocio. En éste área de proceso, cuando se utiliza el término "objetivos" sin el calificador de "medición", indica objetivos del proyecto, de la organización o del negocio.

3) Áreas del proceso relacionadas: Enumera las referencias a áreas de proceso relacionadas.

Para más información sobre cómo monitorizar los parámetros de planificación del proyecto, consúltese el área de proceso Monitorización y Control del Proyecto.

Para más información sobre cómo establecer estimaciones, consúltese el área de proceso Planificación del Proyecto.

Para más información sobre cómo gestionar cuantitativamente el proyecto, consúltese el área de proceso Gestión Cuantitativa del Proyecto.

4). Metas específicas (SG): Describe las características únicas que deben estar presentes para satisfacer el área de proceso. Se utiliza en las evaluaciones para ayudar a determinar si se satisface un área de proceso

5). Prácticas específicas (SP): Es la descripción de una actividad que se considera importante para lograr la meta específica asociada. Las prácticas específicas describen las actividades que se espera que produzcan el logro de las metas específicas.

SG 1 ALINEAR LAS ACTIVIDADES DE MEDICIÓN Y ANÁLISIS

Los objetivos y las actividades de medición están alineados con las necesidades de información y los objetivos identificados.

SP 1.2 ESPECIFICAR LAS MEDIDAS

Especificar las medidas para tratar los objetivos de medición.

Los objetivos de medición se refinan en medidas precisas y cuantificables.

La medición del trabajo del proyecto y de la organización normalmente se puede asignar a una o más categorías de medición. Estas

6) Ejemplos de productos de trabajo: Enumera los resultados de una práctica específica.

Ejemplo de productos de trabajo

1. Especificaciones de medidas base y derivadas.

7) Subpráctica: Es una descripción que proporciona orientación para interpretar e implementar una práctica específica.

Subprácticas

1. Identificar las medidas candidatas en base a los objetivos de medición documentados.

   Los objetivos de medición se refinan en medidas. Las medidas candidatas identificadas se clasifican y se especifican por nombre y unidad de medida.

2. Mantener la trazabilidad de las medidas con los objetivos de medición.

   Las interdependencias entre las medidas candidatas se identifican para permitir la validación posterior de los datos y los análisis candidatos, en apoyo de los objetivos de medición.

8) Metas genéricas (GG): Se denomina "genérica" porque la misma declaración de la meta se aplica a múltiples áreas de proceso.

GG 2 INSTITUCIONALIZAR UN PROCESO GESTIONADO

El proceso está institucionalizado como un proceso gestionado.

9) Prácticas genéricas (GP): Se denominan "genéricas" porque la misma práctica se aplica a múltiples áreas de proceso. Describen las actividades que se consideran importantes para lograr la meta genérica y contribuir a la institucionalización de los procesos asociados con un área de proceso.

GP 2.1 ESTABLECER UNA POLÍTICA DE LA ORGANIZACIÓN

Establecer y mantener una política de la organización para planificar y realizar el proceso.

El propósito de esta práctica genérica es definir las expectativas de la organización en relación con el proceso y hacerlas visibles a aquellos miembros de la organización que están afectados. En general, la alta dirección es responsable de establecer y comunicar las directrices, la orientación y las expectativas para la organización.

10) Elaboración de práctica genérica: Aparecen después de las prácticas genéricas para orientar en que pueden aplicarse.

Elaboración de CM

Esta política establece las expectativas de la organización para establecer y mantener las líneas base, para seguir y controlar los cambios a los productos de trabajo (bajo gestión de configuración), y para establecer y mantener la integridad de las líneas base.

Elaboración de PP

Esta política establece las expectativas de la organización para estimar los parámetros de la planificación, para definir compromisos internos y externos, y para desarrollar un plan para gestionar el proyecto.

Ejercicio 4:

Ejercicio en clases

Seleccionar un área de procesos y describir los siguientes puntos:

1. **Declaración del propósito.**
2. **Áreas del proceso relacionadas.**
3. **Metas específicas y prácticas específicas.**

- **Calidad del producto del Software**

Un modelo de calidad es una descomposición jerárquica del concepto de calidad. Dentro de su estructura general influyen factores externos, criterios y métricas.

→ **Factores:** Puntos de vista del usuario y atributos de calidad.
→ **Criterios:** Punto de vista del producto (calidad interna).
→ **Métricas:** Medidas cuantitativas.

Modelo McCall

MANTENIBILIDAD

¿Puedo arreglarlo?

TESTABILIDAD

¿Puedo probarlo?

FLEXIBILIDAD

¿Puedo modificarlo?

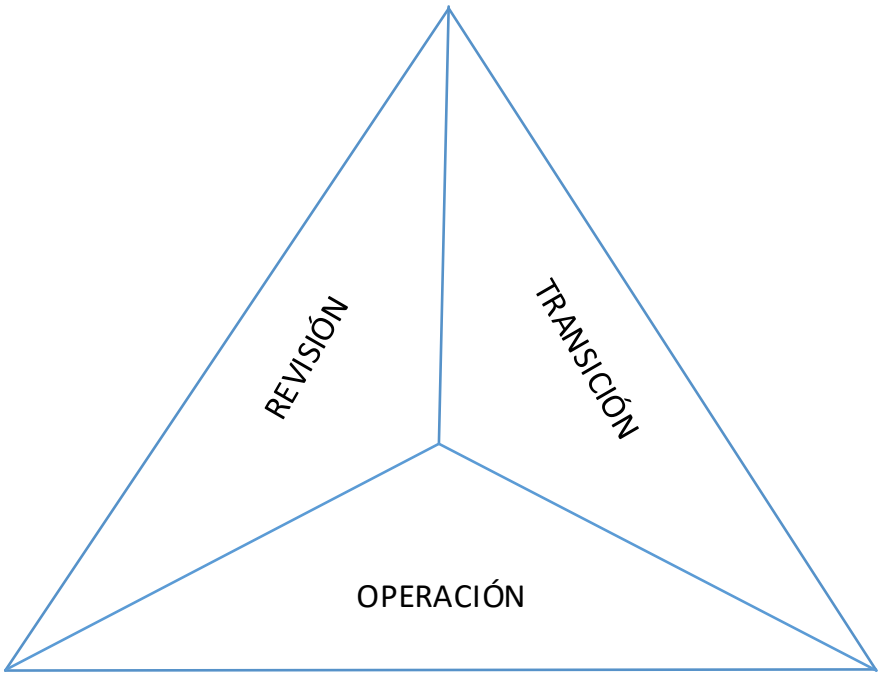
INTEROPERABILIDAD

¿Se puede comunicar con otros sistemas?

PORTABILIDAD

¿Puedo usarlo en otra plataforma?

REUSABILIDAD



CORRECCIÓN ¿El software hace lo que quiero?

CONFIABILIDAD ¿El software lo hace bien siempre?

EFICIENCIA ¿Se ejecuta en mi hardware lo mejor posible?

INTEGRIDAD ¿Es seguro?

USABILIDAD ¿Puedo usarlo?)

Introducción a la ingeniería de software experimental

4.1 Ingeniería del Software

Históricamente, las ingenierías han surgido de prácticas ad-hoc, donde un conjunto de problemas se resolvía de manera arbitraria. En la actualidad, los diseños y las construcciones se fundamentan en un entendimiento teórico, y las ideas sobre la realidad se contrastan con datos empíricos. Para afirmar que un hecho está adecuadamente descrito por una teoría, se emplean operaciones empíricas como observaciones, mediciones y experimentos. En suma, la ingeniería del Software Experimental (en adelante, ISE), traslada a la IS al paradigma experimental:

- → En los años sesenta se compara el rendimiento de los programadores bajo ciertas condiciones controladas.
- → A mediados de los ochenta Basili, presenta un framework para experimentación en Ingeniería del Software, determinando la necesidad de que la experimentación sea sistemática.
- → La IS se considera una disciplina dentro de las ciencias sociales ya que está relacionada con el comportamiento humano, ya que son personas quienes desarrollan software.

4.1.1 El proceso experimental:

El proceso experimental es un método de investigación que implica una alteración deliberada del valor de una o más propiedades para observar cómo se ven afectadas otras propiedades y, en su caso, en qué grado. De esta manera, se identifican las causas que conducen a determinados resultados. A través de los experimentos, se contrastan hechos, suposiciones, asunciones, especulaciones y creencias que son comunes en la investigación científica.

- **Experimentación en IS:**

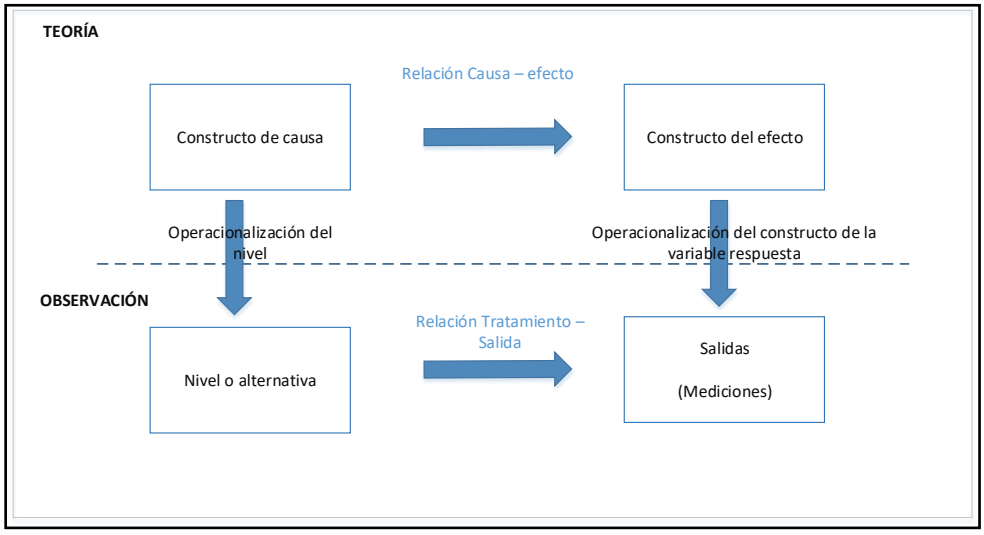
En el ámbito de la ingeniería de software, la mayoría de las tecnologías están fundamentadas en la interacción humana. No existen sistemas de software que sean exactamente iguales; hay una variedad de variables que generan diferencias y que requieren investigación. Además, hay una amplia gama de técnicas y herramientas que prometen resolver diversos problemas; por ejemplo, actualmente existen alrededor de 200 enfoques para la ingeniería de requisitos. Sin embargo, no se reconocen adecuadamente los límites de estas tecnologías en contextos específicos.

A pesar de la diversidad de opciones disponibles, persiste una insuficiente cantidad de análisis y experimentación en la práctica. Esto limita la comprensión de cómo y cuándo aplicar estas tecnologías de manera efectiva, lo que puede afectar la calidad y el éxito de los proyectos en la ingeniería de software.

Experimentos:

→ **Hipótesis:** Es una declaración que propone una posible explicación de algún evento o fenómeno (Given, 2008).

→ **Investigación Empírica:** Es una investigación basada en la observación o en la experiencia.



4.1.2 Terminología del diseño experimental

Tabla 1

Terminología de investigación

Terminología	Descripción
UNIDAD EXPERIMENTAL	Objeto en el cual se ejecuta la unidad experimental. Por ejemplo, el proceso desarrollo, proceso de requisitos.
SUJETO EXPERIMENTAL	La persona quien aplica las técnicas o métodos a la unidad experimental. Ejemplo: El programador, el equipo de desarrollo.
VARIABLE RESPUESTA	Es la salida del experimento. Por ejemplo: calidad del código fuente, productividad del programador.
PARÁMETROS	Cualquier característica que no varía a lo largo del experimento. Son características que no influencian o que no queremos que influencien el resultado del experimento o la variable respuesta. Por ejemplo, la experiencia del Project manager o la experiencia del programador.
FACTOR	Cada característica del desarrollo de software a ser estudiado. Característica del proyecto que es variada intencionalmente durante el experimento y afectan a la variable respuesta. Ejemplo, técnica de Estimación, Técnica de desarrollo.
NIVELES O TRATAMIENTOS	Son los posibles valores de los factores durante cada experimento. Ejemplo, técnica de Estimación: COCOMO, Método de Putnam. Técnica de desarrollo: Test first, test last.
VARIABLES DE BLOQUEO	Supongamos que tenemos un experimento sobre lenguajes de programación y errores en el código fuente. Se espera que la experiencia del programador influya en el número de errores. Pero no tenemos la intención de estudiar el tema de la experiencia del programador. Nuestro objetivo es centrarse solo en la influencia de los lenguajes de programación. Lo ideal sería eliminar la variabilidad debida a la experiencia del programador. (Diseños de bloque, aleatorización).
REPLICACIÓN	Investigadores intentan reproducir la investigación lo más cerca posible. Si los resultados de la replicación son consistentes con la investigación original, se ha aumentado la confianza en la hipótesis que el estudio original apoyó.

Fuente: Elaboración Propia.

- **Planificación**

Selección del contexto: Se debe definir el contexto de estudio, que puede incluir aspectos como el grupo objetivo (estudiantes o profesionales de la industria) y la naturaleza de los problemas a investigar (problemas reales frente a problemas de juguete).

Formulación de hipótesis:

→ H0: Hipótesis nula. No existe una diferencia significativa en la productividad al comparar TDD (Desarrollo Guiado por Pruebas) con ITL (Integración de Pruebas de Línea).
→ H1: Hipótesis alternativa. La productividad de TDD es superior a la de ITL.
→ H2: Hipótesis alternativa 2. La productividad de ITL es superior a la de TDD.

Es importante considerar el error tipo I (rechazar la hipótesis nula cuando es verdadera) y el error tipo II (no rechazar la hipótesis nula cuando es falsa).

Selección de variables:

→ **Variable independiente (factor):** Son las variables que se modifican para estudiar el efecto que producen esos cambios. Por ejemplo, la técnica de desarrollo utilizada (TDD o ITL).
→ **Variable dependiente (resultado):** Son las variables que se analizan para comprobar el impacto de los cambios en las variables independientes. Ejemplos de estas incluyen la calidad y la productividad.
→ **Tratamientos o niveles:** Se refiere a las técnicas de desarrollo empleadas, que en este caso son TDD o ITL.

Selección de sujetos: Los sujetos son las personas que aplican los tratamientos del experimento, como los desarrolladores. La población seleccionada debe ser representativa para asegurar la validez de los resultados.

- **Elección del diseño**

→ **Aleatorización:** Consiste en la asignación aleatoria de tratamientos a los sujetos, así como en la selección aleatoria de los sujetos mismos. Esto ayuda a reducir sesgos y aumentar la validez del experimento.
→ **Bloqueo:** En ocasiones, existen factores que pueden influir en la variable dependiente, pero cuyo efecto no es de interés en el estudio. El bloqueo se utiliza para eliminar estos efectos no deseados. Por ejemplo, se puede dividir a los sujetos en grupos equilibrados en función de la experiencia.

→ **Diseños Experimentales:** Está compuestos por: 1) Diseños de un factor, 2) Diseños de bloque, 3) Diseño factorial, 4) Diseño anidado, 5) Diseño fraccional, 6) Diseño factorial con bloque.

Proyecto	1	2
Técnica aplicada	A	B

Proyecto	Equipo 1	Equipo 2
1	A	B
2	B	A

→ **Instrumentación**

Objetos experimentales: Incluyen especificaciones de requisitos, diseños y otros elementos relevantes para el experimento.

Guías: Listas de comprobación y descripciones de procesos que facilitan la ejecución del estudio.

Instrumentos de medición: Formularios y herramientas que permiten recoger datos de manera estructurada.

→ **Evaluación de la Validez**

La evaluación de la validez se centra en determinar cuán confiables son los resultados obtenidos y en identificar posibles amenazas a la validez.

Validez interna: Refleja el grado de confianza en la relación de causa y efecto, así como la capacidad para extraer conclusiones. Esto incluye considerar cómo se seleccionan y agrupan los sujetos.

Validez externa: Indica hasta qué punto los resultados del estudio pueden generalizarse. Una validez externa baja puede deberse a la selección inadecuada de sujetos o a la realización del experimento en un entorno no adecuado.

Validez de constructo: Evalúa si las variables realmente miden los constructos teóricos que se intentan evaluar. Por ejemplo, utilizar el número de capacitaciones en un tema como medida de experiencia puede ser poco significativo.

Validez de la conclusión: Se refiere a la validez estadística de las conclusiones. Puede verse afectada por un bajo poder estadístico, el incumplimiento de las suposiciones de los tests estadísticos o la falta de fiabilidad de las medidas.

- **Operación**
  - → **Preparación:** Contar con los sujetos experimentales. Además, se debe contar con el consentimiento informado y la confidencialidad. Desvelar los detalles del experimento siempre que no puedan sesgar los resultados.
  - → **Ejecución:** Lo ideal es que los sujetos realicen el experimento en el mismo lugar.
  - → **Validación de datos:** Comprobar que se han recogido los datos correctamente.

- **Análisis e interpretación**
  - → **Estadísticos descriptivos:** Media, desviación estándar, máximo, mínimo, Q1, Q2 (mediana), Q3.
  - → Reducción de datos (Exclusión de valores atípicos).
  - → **Contraste de hipótesis:** Selección del test estadístico adecuado en función del diseño experimental seleccionado y la distribución de los datos.

- **Test Estadísticos**

Los test estadísticos para contraste de hipótesis según el diseño experimental seleccionado (Ver Tabla 2).

Tabla 2

Testa estadísticos según la hipótesis

Tipo de diseño	Test paramétrico	Test no paramétrico
Un factor, dos tratamientos	Test t (t de Student) Test F (Fisher) Test t emparejado	Mann - Whitney Chi 2 Wilcoxon

Un factor, más de dos tratamientos	ANOVA	Kruskal – Walis Chi 2
Más de un factor	ANOVA	

Fuente: Elaboración Propia.

- **Presentación y empaquetamiento de hallazgos**

  → Presentar los hallazgos de los experimentos en un congreso o revista académica.
  → Reporte de experimentos (Documentación homogénea).

Ejercicio 5:

Ejercicio en clases

Revisar un artículo de un experimento en TDD (Erdogmus) y contestar las siguientes preguntas:

¿Cuál es el objetivo del experimento?

¿Cuál es el factor?

¿Cuáles son los tratamientos?

¿Cuántos sujetos son y a qué curso pertenecían?

¿Cuál fue la tarea experimental?

¿Cuántos casos de prueba se desarrollaron con que técnica se implementaron?

¿Qué cuestionarios llenaron los sujetos?

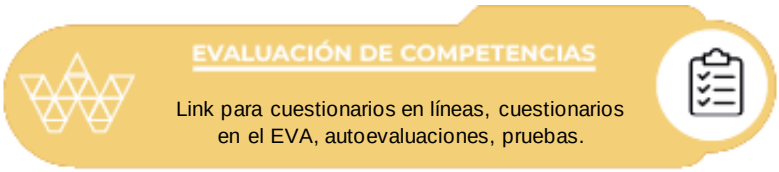
¿Cómo se obtiene el valor de la métrica de Productividad?

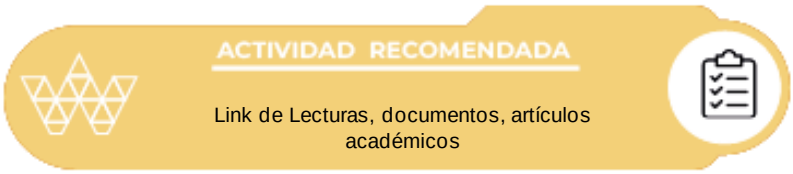
¿Cómo se obtiene el valor de la métrica de Calidad?

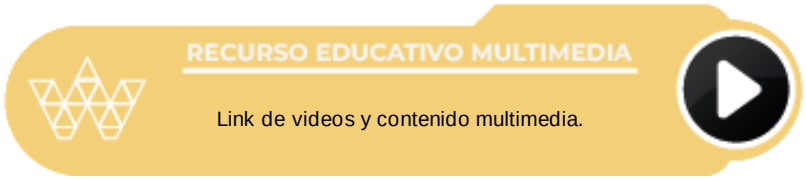
¿Cuál es el test estadístico?

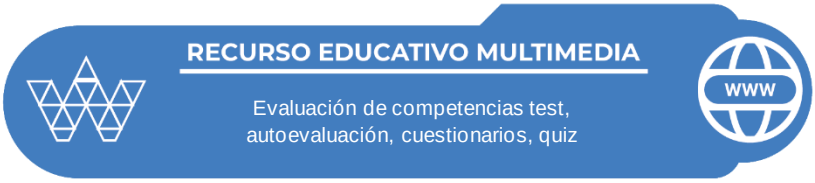
¿Cuál es el resultado del experimento respecto a la calidad y productividad?

1. Anexos


EVALUACIÓN DE COMPETENCIAS
Link para cuestionarios en líneas, cuestionarios en el EVA, autoevaluaciones, pruebas.


ACTIVIDAD RECOMENDADA
Link de Lecturas, documentos, artículos académicos


RECURSO EDUCATIVO MULTIMEDIA
Link de videos y contenido multimedia.


RECURSO EDUCATIVO MULTIMEDIA
Evaluación de competencias test, autoevaluación, cuestionarios, quiz
WWW

2. Bibliografía:

Ajit Singh. (2021). *Agile & Scrum*. Babelcube Inc.

Mall, R. (2014). *Fundamentals of software engineering*. PHI Learning.

Pressman, R. S. (2010). *Ingeniería del software: Un enfoque práctico* (7ª ed.). McGraw-Hill.

Pressman, R. S. (2010). *Ingeniería de software: Un enfoque práctico*. McGraw-Hill.

SCRUMstudy. (2013). *Una guía para el conocimiento de SCRUM (SBOK)*.

Velthuis, M. G. P. (2015). *Calidad de sistemas de información* (3ª ed., ampliada y actualizada). RA-MA.

Ziauddin, S., Kamal Tipu, S., & Zia, S. (2012). *Técnica para la estimación ágil de proyectos de software*. Ministerio de Hacienda y Administraciones Públicas.

España. (2012). *Metodología de gestión de riesgos MAGERIT*. Ministerio de Hacienda y Administraciones Públicas.