UT01. DESARROLLO DEL SOFTWARE

Entornos de Desarrollo

1DAW – C.I.F.P. Carlos III - Cartagena

Índice

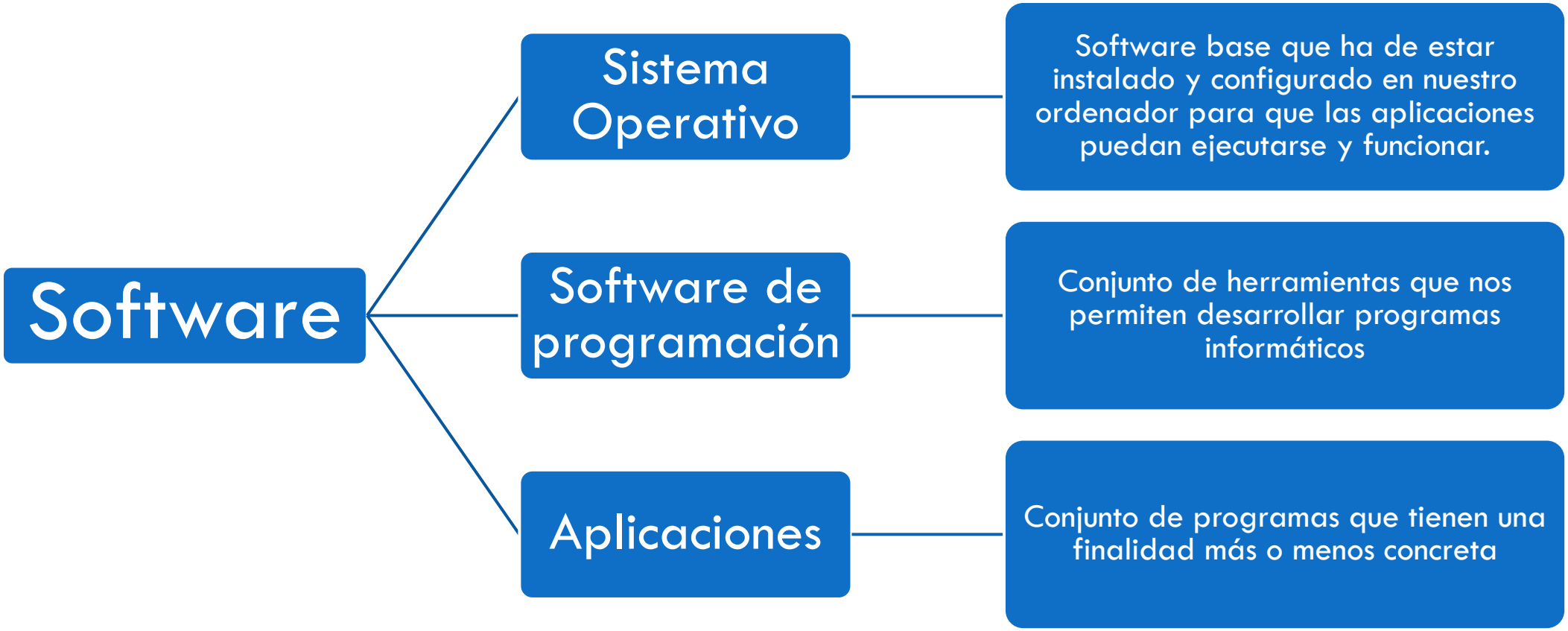
1. Software de ordenador

Software

- El ordenador se compone de dos partes bien diferenciadas: hardware y software.
- "*Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora*" (RAE)
- "*Conjunto de programas de cómputo de, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación*" (estándar 729 IEEE)

Tipos de software. Según el tipo de tarea

- El ordenador se compone de dos partes bien diferenciadas: **Hardware** y **Software**



Tipos de software. Según el tipo de tarea

- **Programa** es un conjunto de instrucciones escritas en un lenguaje de programación
- Tipos de Software

De sistemas

- Librar al usuario de los detalles del hardware que se usa y de su gestión. Proporciona una interfaz de alto nivel, cómoda para el usuario.
- Sistemas operativos. Controladores de dispositivos. Utilidades

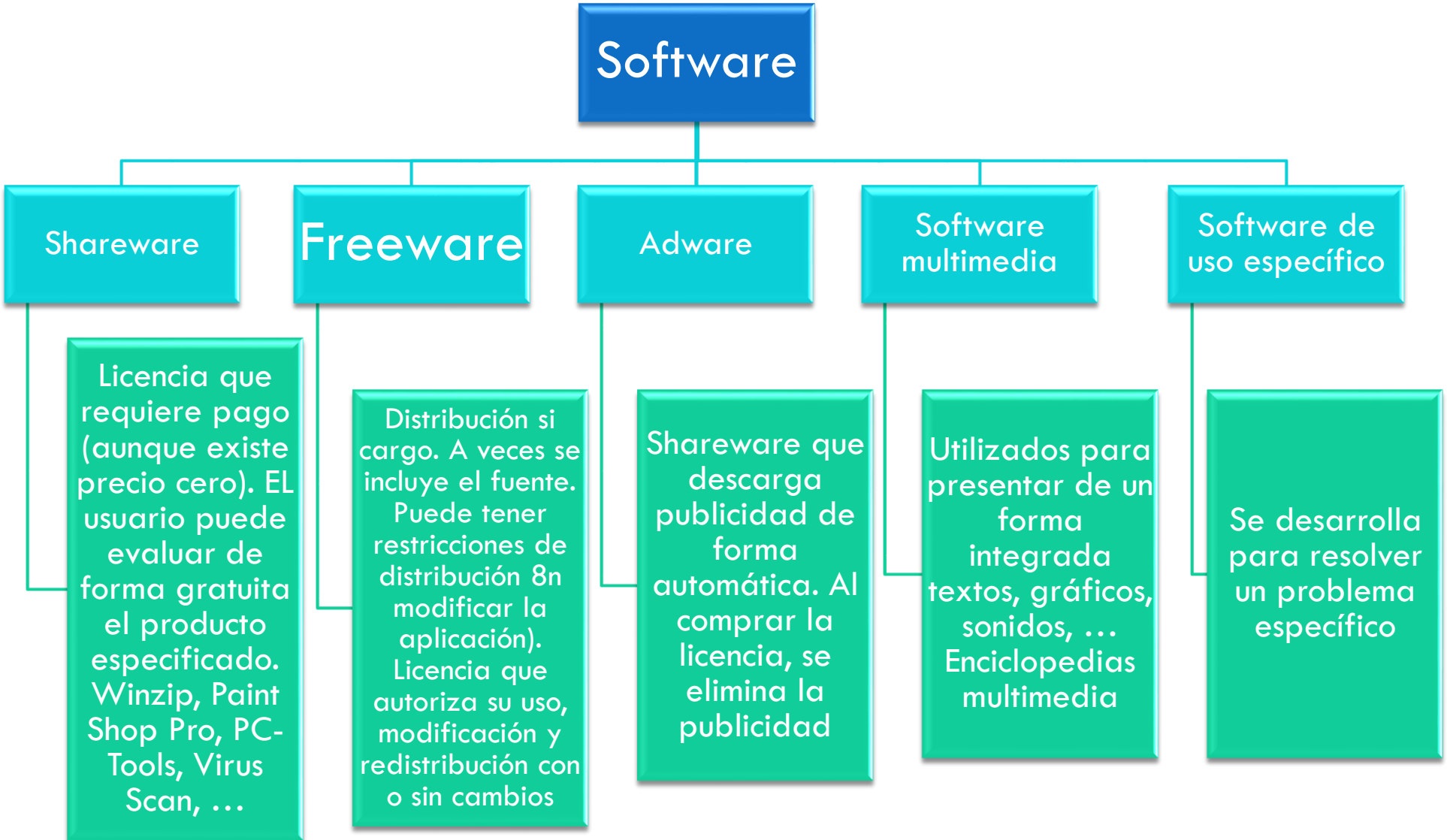
De programación

- Proporcionar herramientas al usuario para el desarrollo de sistemas informáticos
- Editores de texto. Compiladores. Intérpretes. Enlazadores. Depuradores. Entornos integrados (IDE)

De aplicación

- Permitir al usuario realizar una o varias tareas específicas
- Aplicaciones Ofimáticas. Software educativo. Bases de datos. Videojuegos. Software de diseño asistido (CAD)

Tipos de sw. Según el método de distribución



Actividad 1

- Investiga sobre Licencias de Software. Software **libre** y **propietario** y de **dominio público**

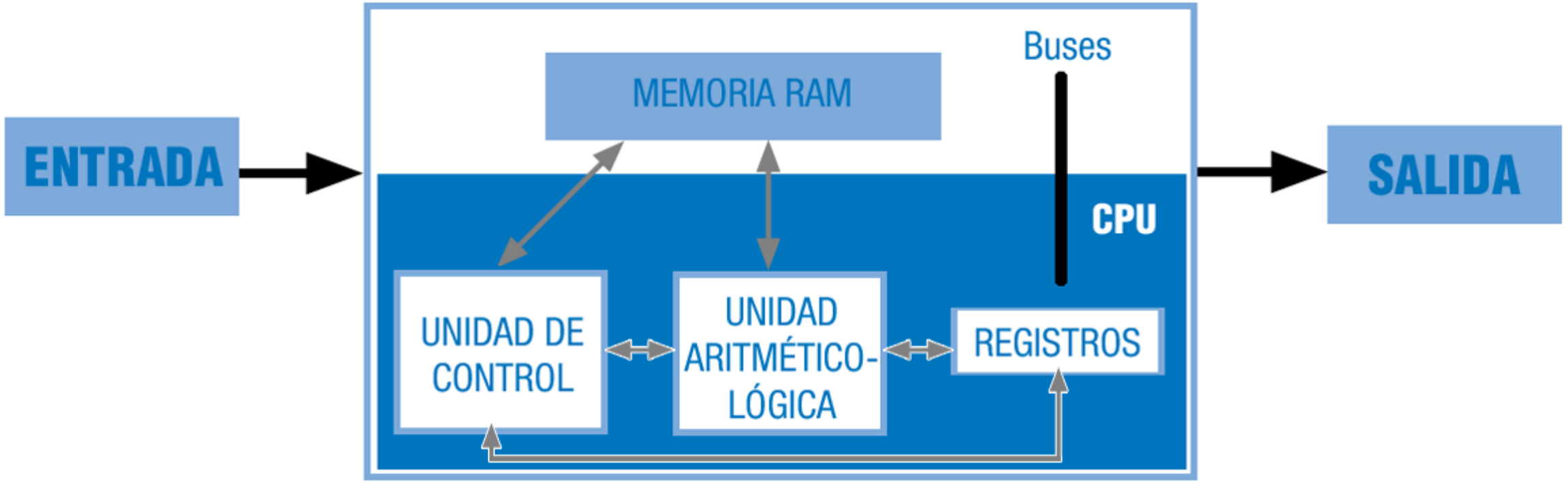
http://www.gnu.org/licenses/license-list.es.html#SoftwareLicenses

2. Relación Hardware – Software

Relación Hardware – Software

- Relación indisoluble entre hardware y el software.
- El software se ejecutará sobre los dispositivos físicos.
- La primera arquitectura hardware con programa almacenado se estableció en 1946 por **John Von Neumann**
- Relación software-hardware desde dos puntos de vista:
  - **Desde el punto de vista del sistema operativo**
    - Encargado de coordinar al hardware
    - Intermediario entre este y las aplicaciones.
    - Encargado de gestionar los recursos hardware durante su ejecución (tiempo de CPU, espacio en memoria RAM, tratamiento de interrupciones, gestión de los dispositivos de Entrada/Salida, etc
  - **Desde el punto de vista de las aplicaciones**
    - Multitud de lenguajes de programación diferentes con sentencias en lenguaje natural
    - El hardware de un ordenador sólo es capaz de interpretar señales eléctricas (ausencias o presencias de tensión) que, en informática, se traducen en secuencias de 0 y 1 (código binario).
    - ¿Cómo será capaz el ordenador de "entender" algo escrito en un lenguaje que no es el suyo?.

Arquitectura Von Neumann



3. Desarrollo de software

Ciclo de vida del software

- **Ciclo de vida del software**
  - **IEEE 1074**: "una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software".
  - **ISO 12207-1**: "actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto software, abarcando la vida del sistema desde al definición de los requisitos hasta la finalización de su uso.
- **Ciclo de desarrollo**
  - Subconjunto del anterior que empieza en el análisis y termina en con la entrega del sistema al usuario

Ciclo de vida del software

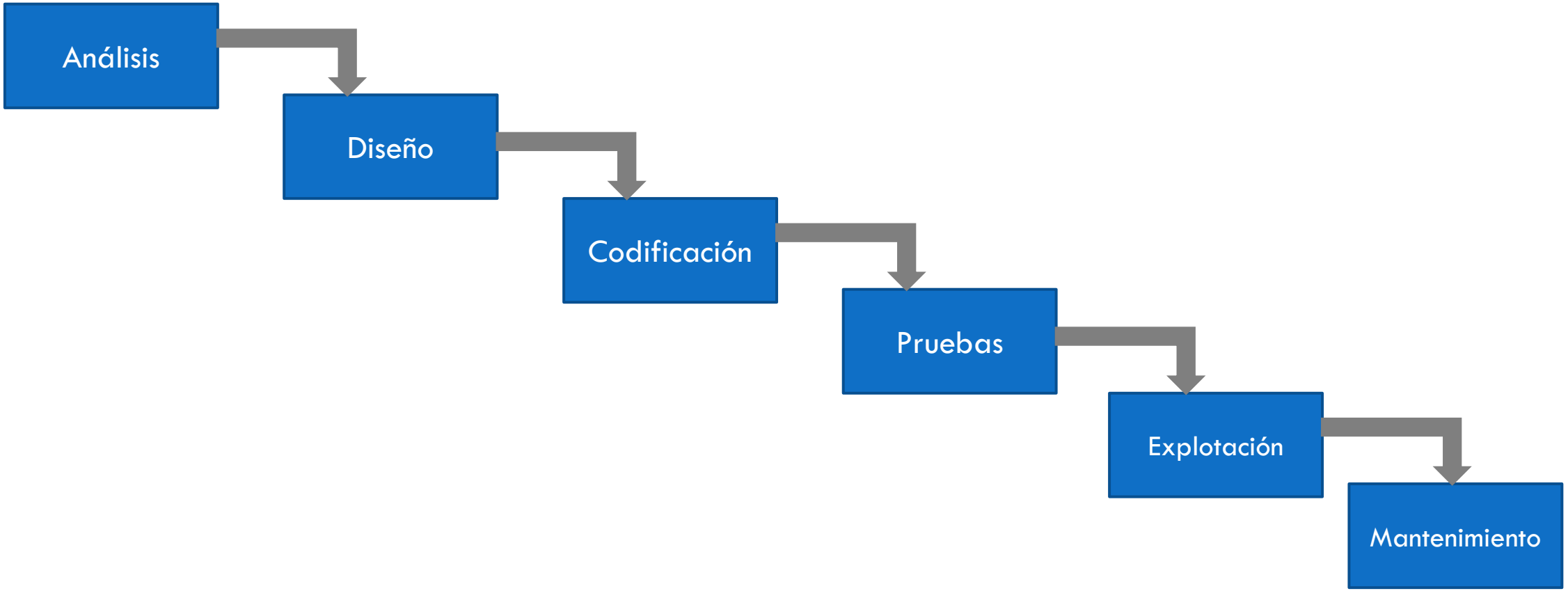
- Etapas.
  - **Análisis**. Construye un modelo de requisitos. Se trata de entender y comprender de forma detallada el problema a resolver. Documentación entendible, completa u fácil de verificar y modificar. Qué hacer
  - **Diseño**. Cómo hay que hacerlo. Estructuras de los datos, arquitectura del software, interfaz de usuario y procedimientos. PE. Seleccionar el lenguaje de programación, SGBD, etc
  - **Codificación**. Traducir lo escrito en el diseño a un forma legible por la máquina. Salida: código ejecutable
  - **Pruebas**. Comprobar que se cumplen criterios de corrección y calidad. Garantizan el correcto funcionamiento del sistema
  - **Mantenimiento**. Después de la entrega del software. Adaptarse a los cambios. Cambios por errores. Adaptación al entorno (pe cambio de SO), mejoras funcionales.

Ciclo de vida del software

- **Documentación.** Cada etapa tiene como entrada los documentos de la etapa anterior y obtiene nuevos documentos.

Modelos de ciclo de vida I

- **Modelo en cascada clásico**
  - Propuesta por Royce [ROYCE, 1970]
  - Variaciones a lo largo del tiempo, Sommerville, Boehm, …
  - Fases variables.



Modelos de ciclo de vida

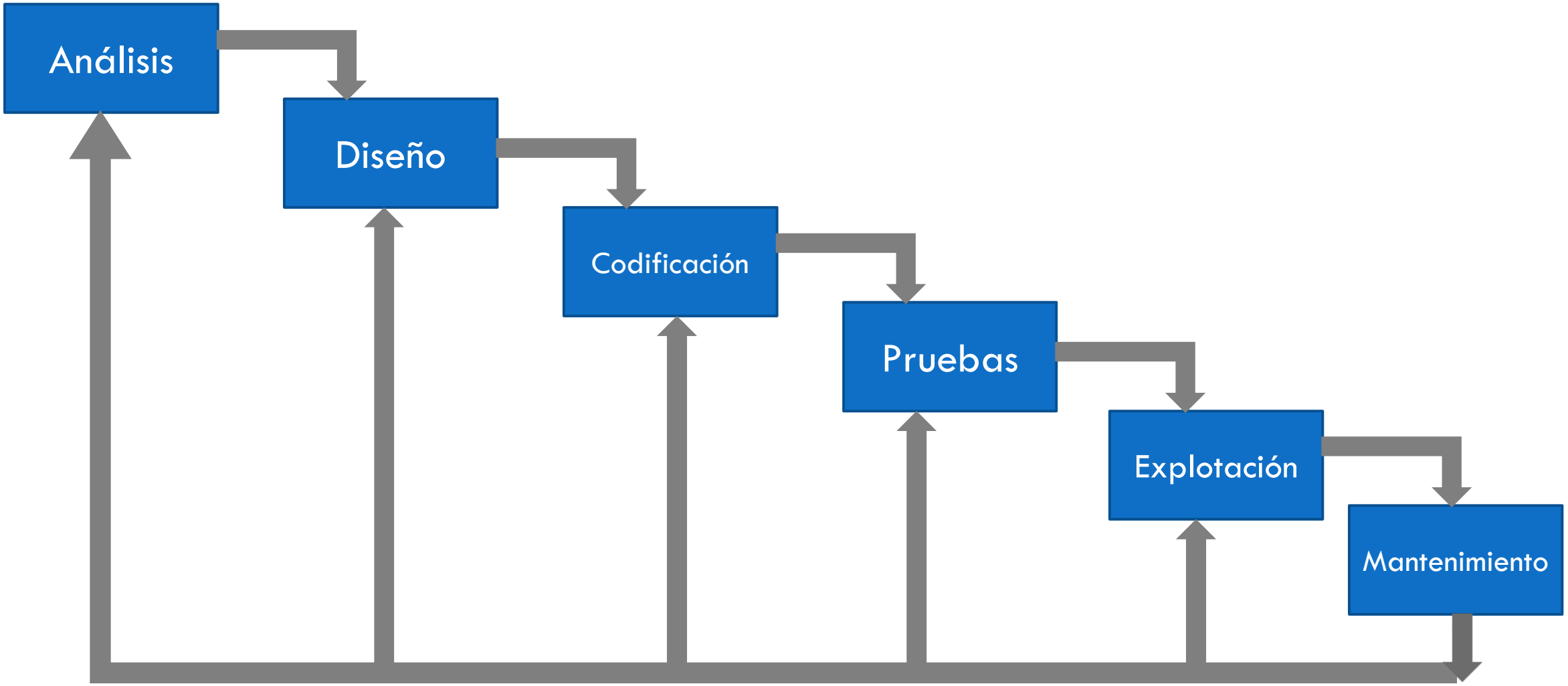
- **Modelo en cascada**
  - **Características**
    - Imposible de utilizar.
    - Requiere conocer de antemano todos los requisitos del sistema
    - Para desarrollos pequeños.
    - No permite retroalimentación (no refleja el mundo real)
    - Se tarda mucho en pasar por todo el ciclo puesto que no se puede pasar a una etapa sin terminar la anterior

Modelos de ciclo de vida

- **Modelo en cascada con retroalimentación**
  - Uno de los modelos más utilizados.
  - Proviene del anterior con retroalimentación entre etapas.
  - Vuelta atrás para corregir, modificar distintos aspectos
  - Modelo adecuado para proyectos con pocos cambios y requisitos claros.

Modelos de ciclo de vida

- **Modelo en cascada con retroalimentación**



Modelos de ciclo de vida

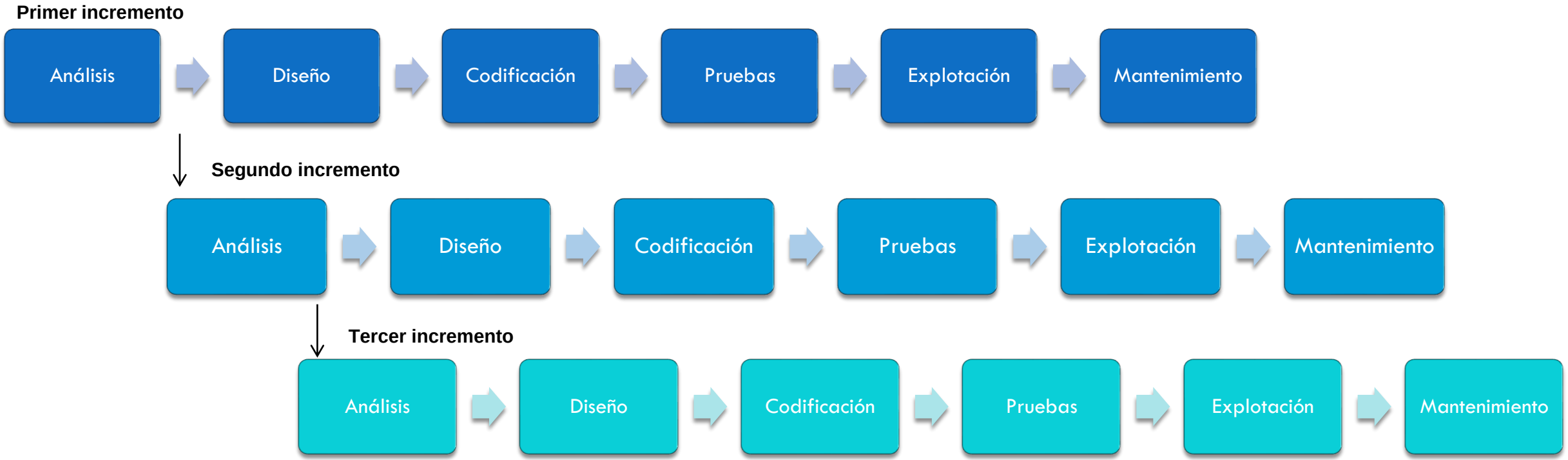
- Modelo en cascada
  - Ventajas
    - Fácil de comprender, planificar y seguir
    - La calidad del producto resultante es alta
    - Permite trabajar con personal poco cualificado
  - Inconvenientes
    - La necesidad de tener todos los requisitos definidos desde el principio
    - Difícil volver atrás si se comenten errores en una etapa
    - Producto no disponible para su uso hasta que no está terminado
  - Se recomienda su uso cuando:
    - El proyecto es similar a alguno que se haya realizado con éxito anteriormente
    - Los requisitos son estables y están bien comprendidos

Modelos de ciclo de vida

- **Modelos evolutivos.**
  - Software que evoluciona con el tiempo.
  - Versiones cada vez más completas hasta el producto final
  - Los más conocidos
    - Modelo iterativo incremental
    - Modelo en espiral

Modelos de ciclo de vida

- **Modelo iterativo incremental.**
  - Basado en el modelo en cascada con realimentación
  - Varios ciclos en cascada que se repiten y refinan en cada *incremento*.
  - Las versiones son cada vez más completas hasta llegar al producto final.
  - P.e.- un procesador de textos. 1 inc: funciones básicas de gestión de archivos y producción de documentos. 2 inc: desarrollo de funciones gramaticales y de corrección ortográfica. 3 inc: desarrollo de funciones avanzadas de paginación

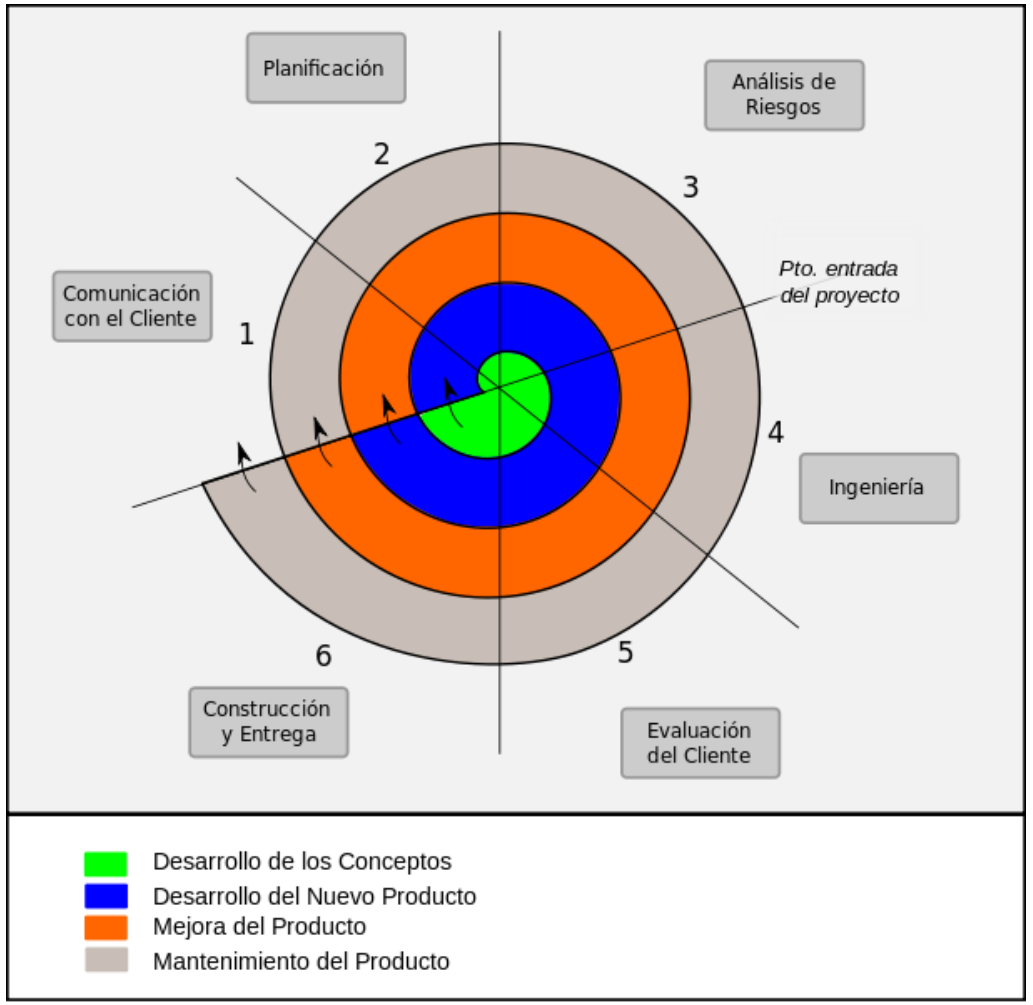


Modelos de ciclo de vida

- Modelo en iterativo incremental
  - Ventajas
    - No se necesitan conocer todos los requisitos al comienzo
    - Permite la entrega temprana al cliente de partes operativas del software
    - Las entregas facilitan la realimentación de los siguientes entregables
  - Inconvenientes
    - Difícil estimación del esfuerzo y el coste final necesario
    - Riesgo de no acabar nunca
    - No recomendable para desarrollo de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido, y/o de alto índice de riesgos
  - Se recomienda su uso cuando:
    - El proyecto es similar a alguno que se haya realizado con éxito anteriormente
    - Los requisitos o el diseño no están completamente definidos y es posible que haya grandes cambios

Modelos de ciclo de vida

- **Modelos evolutivos**
  - **Modelo en espiral.**
    - El software se va construyendo repetidamente en sucesivas versiones que son cada vez mejores por incrementar la funcionalidad de cada versión
    - Reduce riesgos.



Modelos de ciclo de vida

- **Modelos evolutivos. Modelo en espiral**
  - Podría maquetar en papel, modelos de pantalla.
  - En el último ciclo se tendría un prototipo operacional.
  - Fases.
    - Determinar objetivos.
    - Análisis de riesgos. Un riesgo puede ser: requisitos no comprendidos, mal diseño, errores en la implementación
    - Desarrollar y probar
    - Planificación. Revisar y evaluar lo hecho y decidir si se continua, planificando las fases del ciclo siguientes

Modelos de ciclo de vida

- Modelo en espiral
  - Ventajas
    - No se requiere una definición completa de los requisitos al comienzo
    - Análisis de riesgos en todas las etapas
    - Reduce riesgos del proyecto
    - Incorpora requisitos de calidad
  - Inconvenientes
    - Costo del proyecto aumenta a medida que la espiral pasa por sucesivas iteraciones
    - El éxito del proyecto depende de la fase de análisis de riesgos
  - Se recomienda su uso en:
    - Proyectos de gran tamaño y que necesitan constantes cambios
    - Proyectos donde sea importante el factor de riesgo
    - Muy utilizado en sistemas orientados a objetos

Herramientas de apoyo al desarrollo del software

- Automatizar las tareas y ganar fiabilidad y tiempo.
- Centrarnos en los requerimientos del sistema y el análisis del mismo, principales de los fallos del software.
- Herramientas **CASE** – Computer Aided Software Engineering
  - Conjunto de aplicaciones que se utilizan en el desarrollo de software con el objetivo de reducir costes y tiempo del proceso, mejorando por tanto la productividad del proceso.
- **RAD -** Rapid Application Development - Desarrollo rápido de aplicaciones = Desarrollo iterativo + construcción de prototipos + CASE.
- La tecnología CASE = **automatizar** las fases del desarrollo de software
  - Mejorar la planificación del proyecto.
  - Darle agilidad al proceso.
  - Poder reutilizar partes del software en proyectos futuros.
  - Hacer que las aplicaciones respondan a estándares.
  - Mejorar la tarea del mantenimiento de los programas.
  - Mejorar el proceso de desarrollo, al permitir visualizar las fases de forma gráfica.

Herramientas de apoyo al desarrollo del software

- Clasificación
  - **U-CASE-** ofrece ayuda en las fases de planificación y análisis de requisitos.
  - **M-CASE-** ofrece ayuda en análisis y diseño.
  - **L-CASE-** ayuda en la programación del software, detección de errores del código, depuración de programas y pruebas y en la generación de la documentación del proyecto.