



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Electrónica

Ingeniería de Software:
Introducción
Proceso y Metodologías de Desarrollo
Desarrollo Iterativo e Incremental y Ágil

ELO329: Diseño y Programación Orientados a Objetos

Departamento de Electrónica

Universidad Técnica Federico Santa María

Motivación

Problema: Imagina que tienes un **equipo** de construcción que se enfrenta a diferentes **proyectos** de construcción de estructuras, desde **casas para perros hasta edificios rascacielos**. Sin embargo, han surgido desafíos en la planificación, seguimiento y coordinación de estos proyectos debido a la falta de organización.

Casa para perros



Casa de familia acomodada



La puede construir una sola persona

Requerimientos:

- ☐ Modelado mínimo.
- ☐ Proceso simple.
- ☐ Herramientas simples.

Construida eficientemente (tiempo razonable) por un equipo

Requerimientos:

- ☐ Modelado.
- ☐ Proceso bien definido.
- ☐ Herramientas más sofisticadas.

Rascacielos



Motivación

!No es lo mismo 😲 !

Hacer una Tarea en
Programación de 1er. año

≠

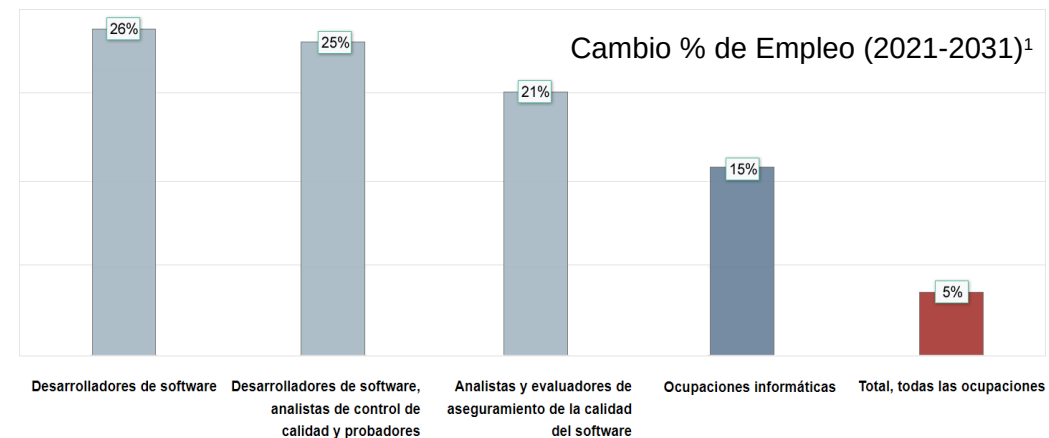
Desarrollar un Producto
de Software



¿Cómo debe cambiar el proceso de desarrollo?

Ingeniería de Software. Profesión

			
Vacantes ofrecidas 1,622,200 (2021)	Nivel de Educación Ingeniería/Bachelor	Salario Promedio US\$109.020 /anual	Crecimiento 25% (2021-2031)

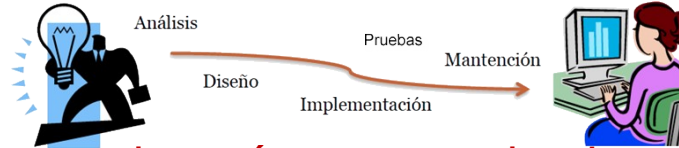


¹ Fuente: <https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm>

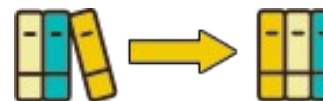
Ingeniería de Software. Definición

Definición

- ❑ Aplicación de un **enfoque sistemático** basado en principios científicos, disciplinado y cuantificable para el diseño y desarrollo de software (ISO/IEC/IEEE 24765:2010²).



- ❑ Pero HOY, **software es mucho más que un algoritmo**, que una tarea o un programa sencillo. El resultado debe responde a una o más necesidades:
 - ❑ Dimensiones.
 - ❑ Interacción entre componentes/objetos.
 - ❑ Equipamiento para su desarrollo.
 - ❑ Usuario final (*stackholders*) requiere versiones intermedias (antes del software completo).
- ❑ Se busca desarrollar software, que resuelva el problema, en **tiempos y a costos controlados**.
- ❑ **Organiza** el desarrollo de software mediante el uso de metodologías y prácticas estructuradas.



Ingeniería de Software. Objetivos

Definiciones

Objetivos	Calidad	<ul style="list-style-type: none">❑ El producto de software cumple con los criterios de aceptación especificados en los requerimientos.<ul style="list-style-type: none">▪ Confiable: Se espera que el producto pueda cumplir en cualquier instante de tiempo
	Reusabilidad	<ul style="list-style-type: none">❑ Un producto de software debe garantizar que los módulos se puedan reutilizar para desarrollar otros nuevos.
	Testeable	<ul style="list-style-type: none">❑ El software debe estar encapsulado de tal forma que cada componente sea susceptible a ser testada para garantizar calidad.
	Adaptable	<ul style="list-style-type: none">❑ El producto debe poder evolucionar según los cambios que vayan surgiendo en los requerimientos.
	Portable	<ul style="list-style-type: none">❑ El software debería ser fácilmente transferible desde un sistema computacional a otro.

Ingeniería de Software. Términos

Ingeniero de software vs. Desarrollador de software

¡A veces los términos se utilizan indistintamente!



Ingeniero de software (*software engineering*)

- ☐ También desarrollan.
- ☐ Concepción y diseño del software, tomando decisiones técnicas y arquitectónicas clave.
- ☐ Alta visión y comprensión del panorama general, base de conocimientos.
- ☐ Proceso de gestión del ciclo de vida de desarrollo del software (*software development life cycle - SDLC*).

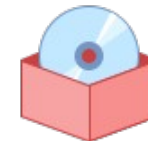


Desarrollador de software (*software developer*)

- ☐ Más limitado en alcance que el Ingeniero de Software.
- ☐ Creatividad lógica/programación.
- ☐ Tarea de codificación para implementar características y funcionalidades específicas y resolver problemas específicos (requerimientos).

Resumen

- Proporcionan un enfoque científico para el desarrollo de software.
- Guían el proceso de desarrollo de software.
- Identifican los pasos necesarios para desarrollar software.





UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Electrónica

Proceso de Desarrollo de Software (Software Development Life Cycle - SDLC)

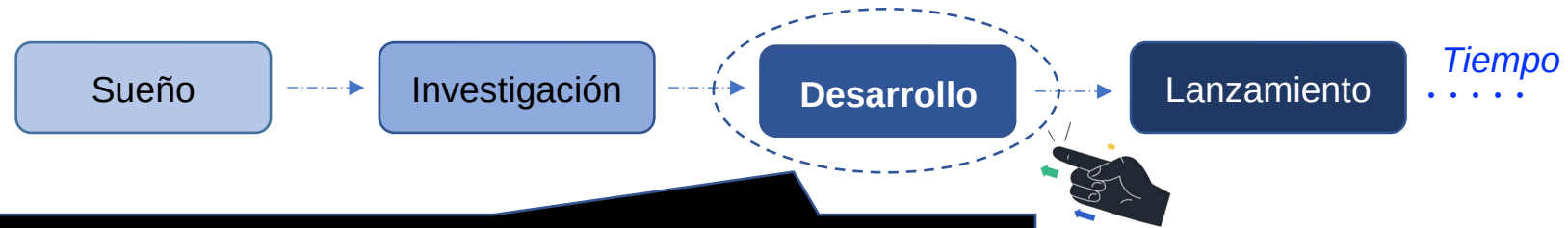
ELO329: Diseño y Programación Orientados a Objetos

Departamento de Electrónica

Universidad Técnica Federico Santa María

Proceso de Desarrollo – SDLC (1/2)

¿Dónde se ubica el Proceso de Desarrollo de Software (SDLC)?

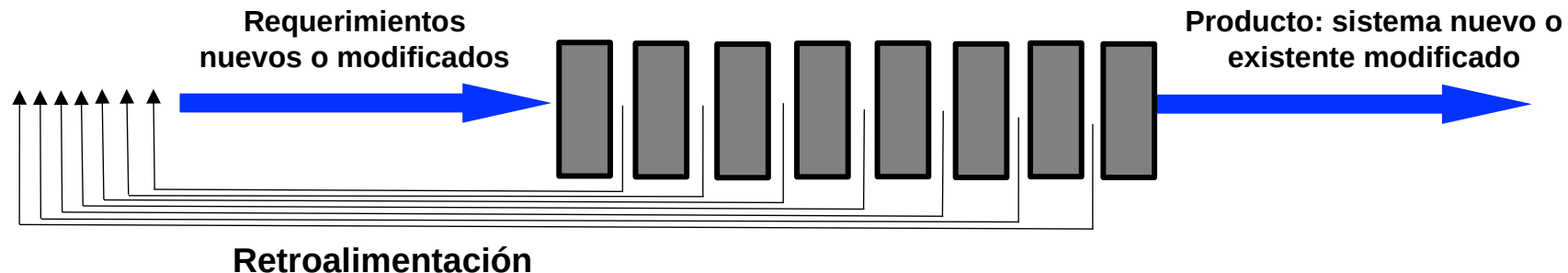


Define **Quién** debe hacer **Qué**, **Cuándo** y **Cómo** debe hacerlo

- ❑ El proceso de desarrollo a la antigua, como “caja negra”:



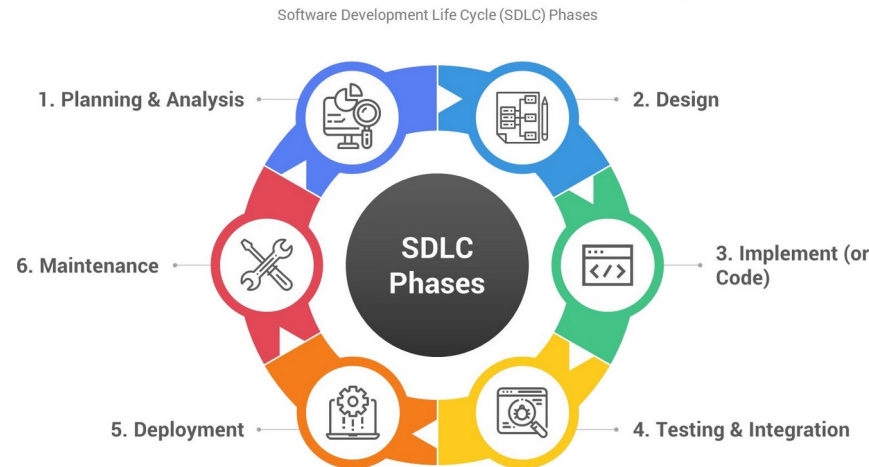
- ❑ El proceso de desarrollo como se concibe ahora:



Proceso de Desarrollo – SDLC (2/2)

- ❑ El original SDLC fue concebido como un método de cascada (*Waterfall method*) tradicional donde las fases eran **lineales** (caso caja negra).
- ❑ Estas fueron adaptadas para introducir **iteraciones** (acomodar requerimientos de cambios).

Software Development Life Cycle (SDLC) Phases



Ventajas

- ❑ Responder a los requisitos cambiantes.
- ❑ Mejora la eficiencia y reduce los riesgos.
- ❑ Facilita la comunicación entre las partes interesadas.
- ❑ Los miembros del equipo conocen en qué deberían estar trabajando y cuándo.
- ❑ Los miembros del equipo saben cuándo el desarrollo puede pasar a la siguiente fase.

Proceso de Desarrollo. Etapas (1/6)

Etapa 1: Planificación (Planning)

- ❑ **Requerimientos:** se recopilan, analizan, documentan y priorizan (entendimiento inicial del problema a resolver). Estos son registrados en un documento llamado **SRS** (*software requirements specification*).

¡SRS necesita aprobación de  usuarios finales!

- ❑ Al planificar una solución de software, se deben considerar los siguientes **factores**:

- | | |
|---|---|
| <ul style="list-style-type: none">▪ Usuarios de la solución (reuniones con clientes)▪ El propósito general de la solución▪ Entradas y salidas de datos▪ Cumplimiento legal y regulatorio | <ul style="list-style-type: none">▪ Identificación de riesgo▪ Requisitos de aseguramiento de la calidad▪ Asignación de recursos humanos y financieros▪ Programación de proyectos |
|---|---|

- ❑ Como parte del proceso de planificación, los **costos de mano de obra (esfuerzo) y materiales** se **estiman** y se comparan con las **limitaciones de tiempo**.

- ❑ Identificación de los **equipos de proyecto** y se proponen los **roles** de cada miembro del equipo.

Proceso de Desarrollo. Etapas (2/6)

Etapa 2: Diseño (Design)



- ☐ Los requerimientos se obtienen del **SRS** para desarrollar la **Arquitectura de Software**.
- ☐ Los requerimientos son transformados en código.



- ☐ Varios miembros del equipo **trabajan juntos** en esta etapa para diseñar la arquitectura.
- ☐ La **arquitectura es revisada** por las partes interesadas y el equipo. Durante esta fase, se pueden diseñar prototipos.
 - Un **prototipo** es un croquis preliminar del sistema, o partes del sistema, que se utiliza con fines de demostración.
 - Diseño de interfaces gráficas (WEB, GUI, comandos, voz, entre otros)
 - Realización de diagramas (UML)
- ☐ **Comunicación de reglas de negocio y lógica de aplicación.**



- ☐ El documento creado en esta fase se denomina documento de diseño **SDD** (*software design document*) y los desarrolladores lo utilizan durante la siguiente fase, que es la fase de desarrollo.

Proceso de Desarrollo. Etapas (3/6)

Etapa 3: Implementación (Implementation or Coding)



- ☐ Una vez que se completa el **SDD**, los planificadores del proyecto utilizan el **SDD** para determinar y **asignar tareas** de codificación.



- ☐ Los desarrolladores comienzan el **proceso de codificación**.
- ☐ Esta fase a menudo requiere el uso de **herramientas de programación**, diferentes lenguajes de programación y *stacks* de software.
 - Stacks de software: colección de componentes independientes que trabajan unidos para respaldar la ejecución de una aplicación.
- ☐ Comentar el código fuente **/** */** en forma entendible y razonable.
- ☐ La **calidad del código** debe cumplir con los requerimientos previstos del software.
- ☐ Codificación **limpia y consistente, fácil de leer y mantener**
- ☐ Sigue **estándares** de codificación.



- ☐ Las organizaciones también pueden tener **estándares o pautas propias que deben seguirse**.

Proceso de Desarrollo. Etapas (4/6)

Etapa 4: Pruebas (Testing)



- ❑ Una vez la etapa de codificación está completa, entonces comienza el proceso de pruebas (**testing**).



- ❑ Algunos proyectos grandes tienen **equipos de prueba dedicados**.
- ❑ El código se prueba para garantizar la estabilidad, la seguridad y que cumpla con los requisitos del **SRS**. Las pruebas pueden ser manuales, automatizadas o ambas.



- ❑ Se identifican errores (**BUGs**) o **requerimientos faltantes**, se corrigen y se vuelven a probar hasta que el software tenga **estabilidad**.



- ❑ Algunos niveles comunes de prueba incluyen **pruebas unitarias (unit testing)** o **pruebas de integración (integration testing)**, **pruebas del sistema (system testing)** y **pruebas de aceptación (acceptance testing)**.
 - Unit testing: se prueban módulos individualmente.
 - Intregation testing: se prueban módulos trabajando en conjunto.
 - System testing: verifican que una aplicación realice tareas según lo diseñado.
 - Acceptance testing: control de calidad **QA** (*quality assurance*) que determina hasta qué punto una aplicación cumple con la aprobación de los usuarios finales **UAT** (*user acceptance testing*).

Proceso de Desarrollo. Etapas (5/6)

Etapa 5: Despliegue (Deployment)



- ❑ La fase de **puesta en producción** es donde la aplicación se lanza al entorno de producción y se pone a disposición de los usuarios.



- ❑ Esto también puede ocurrir por pasos, primero, se lanza a una plataforma de prueba para la aceptación del usuario, también llamado **UAT**, y una vez que el cliente firma (está de acuerdo) la funcionalidad, se lanza a producción.



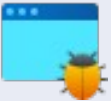
- ❑ Este enfoque se puede utilizar para hacer que el software esté disponible en un sitio WEB, una tienda de aplicaciones para dispositivos móviles o un servidor de distribución de software en una red corporativa (e.j., Apple App Store, Google Play Store, Amazon App Store, entre otras).

Proceso de Desarrollo. Etapas (6/6)

Etapa 6: Mantenimiento (Maintenance)



- ☐ La fase de **mantenimiento** ocurre una vez que el código se ha implementado en un entorno de producción.



- ☐ También **ayuda a encontrar errores**, identificar problemas de interfaz de usuario (UI) para identificar otros requisitos que pueden no haber sido enumerados en el **SRS**.






- ☐ **Mejoras de código** también se pueden identificar en esta etapa.



- ☐ Si se descubren errores en esta fase que se pasaron por alto durante la fase de testing, **es posible que estos errores deban corregirse** (resolución de incidencias) para **problemas de alta prioridad** o incorporarse a los requerimientos como parte de una **próxima versión del software** y el proceso puede comenzar de nuevo.

Proceso de Desarrollo. Lanzamiento


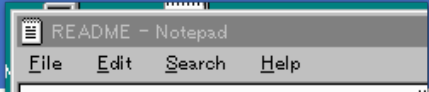
¿Cómo se distribuye la versión más reciente de un software?
“Release”

 Alpha	 Beta	 Disponibilidad General
<ul style="list-style-type: none">❑ Primera versión funcional del sistema <u>lanzada</u> a un <u>grupo selecto</u> de partes interesadas.❑ Puede <u>contener errores</u>. Contiene la mayor parte de la funcionalidad deseada.❑ <u>Cambios de diseño</u> pueden ocurrir.	<ul style="list-style-type: none">❑ Versión limitada, se entrega a las <u>partes interesadas fuera de la organización en desarrollo</u>.❑ Probar el software en <u>condiciones reales</u>.❑ La versión beta debe <u>cumplir con todos los requisitos funcionales</u>.	<ul style="list-style-type: none">❑ Después de <u>β</u>, se acuerdan, realizan y prueban los cambios, y se lanza una <u>versión estable (GA)</u>.❑ La audiencia para el lanzamiento de <u>GA</u> son <u>todos los usuarios</u>.❑ Ejemplo: IntelliJ Release repository³.

³Ejemplo: IntelliJ Release repository (<https://www.jetbrains.com/intellij-repository/releases>)

Proceso de Desarrollo. Documentación

El software debe ser documentado para ser proporcionado a usuarios finales no técnicos y usuarios técnicos

 Documen tación	Sistema	<ul style="list-style-type: none">❑ Explicación detallada del funcionamiento del software o cómo usarlo.❑ Consta de archivos README, comentarios en línea, documentos de arquitectura y diseño, información de verificación y guías de mantenimiento. 
	Usuario	<ul style="list-style-type: none">❑ La documentación del usuario se proporciona a los usuarios finales no técnicos para ayudarlos en el uso del producto.❑ Se proporciona en forma de guías de usuario, videos y manuales de instrucciones y ayuda en línea.

Metáfora del columpio. Incertidumbre



Como el cliente lo explicó



Como el líder del proyecto lo entendió



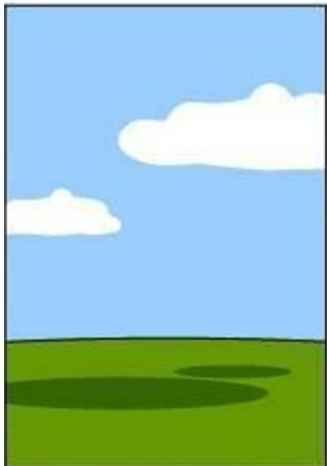
Como el analista lo diseñó



Como el programador lo escribió



Como el vendedor lo describió



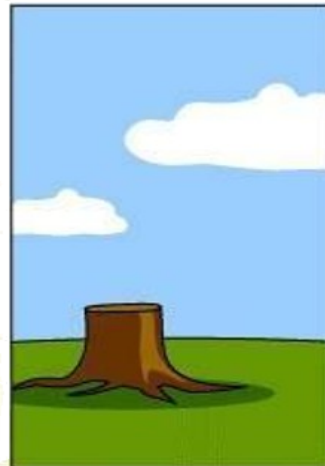
Como fue documentado el proyecto



Que aplicaciones se instalaron



Como le fue facturado al cliente



Como se le dio soporte



Lo que el cliente realmente necesitaba

¿Podré cumplir con los plazos?

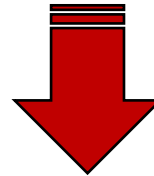
¿Estaré dentro de lo presupuestado?

¿Quedará satisfecho el cliente?

¿METODOLOGÍA DE DESARROLLO?

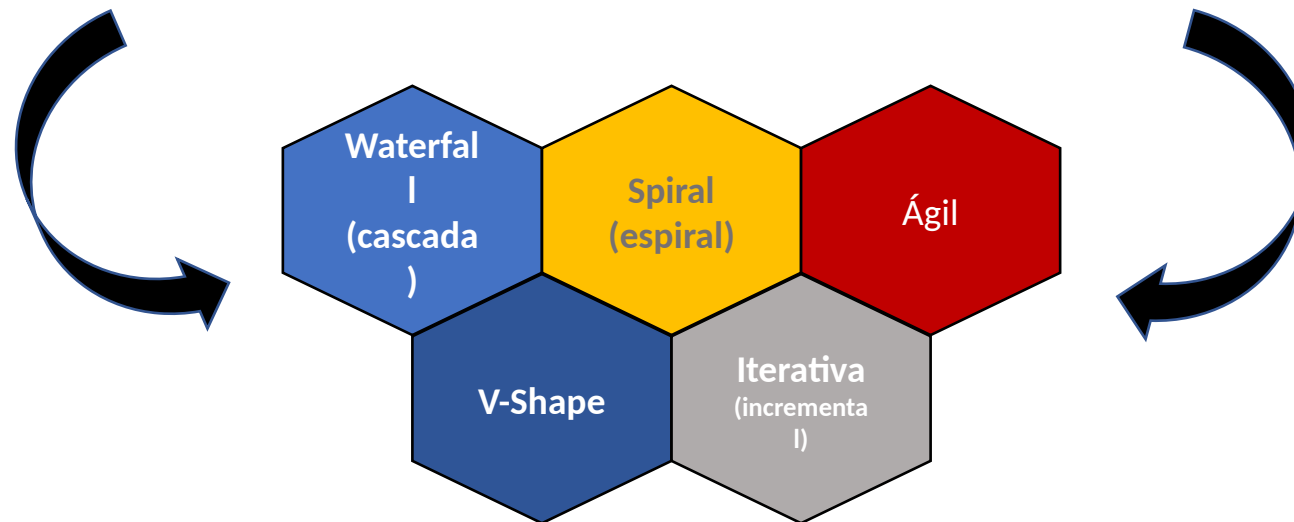
Proceso de Desarrollo. Metodología de Desarrollo (1/5)

¿Cómo lograr la comunicación y facilitar el intercambio de información entre los miembros del equipo de desarrollo?



□ Una **metodología** es el **conjunto de procedimientos** que imponen un proceso disciplinado sobre el desarrollo de software **con el fin** de hacerlo más **predecible** y **eficiente**.

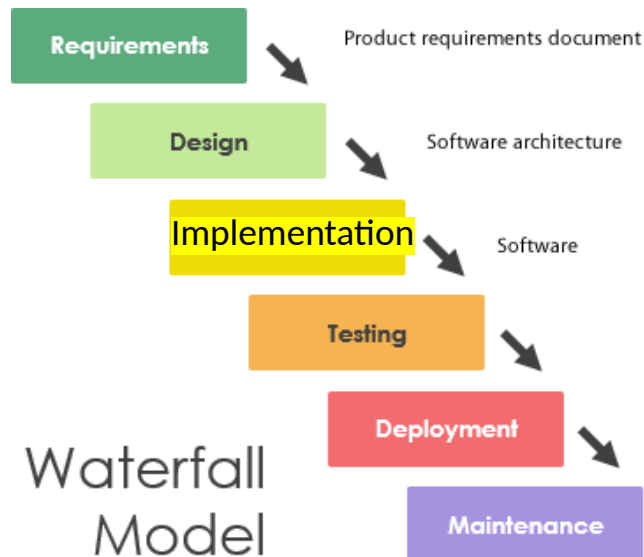
Hay varias metodologías de desarrollo



Proceso de Desarrollo. Metodología de Desarrollo (2/5)

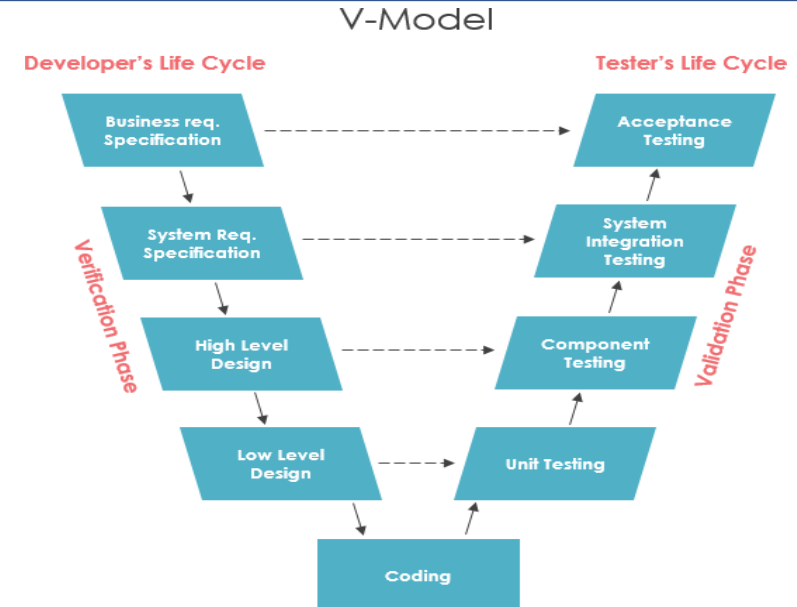
Modelo cascada

- ❑ Método **secuencial** de desarrollo de software donde la **salida** de una fase es la **entrada** de la siguiente fase.
- ❑ La **planificación** (e.j., definición de requisitos, diseño arquitectónico) se **realiza por adelantado**.
- ❑ Genera **largos intervalos** (años), entre lanzamientos de versiones.



Modelo V

- ❑ Es como una cascada, \therefore también es **secuencial**.
- ❑ Fases en forma de V. Las fases (**izquierdas**) son de “**verificación**”, fases (**derechas**) son de “**validación**”.
- ❑ Las **pruebas** se realizan durante las fases de verificación y se ejecutan en las fases de validación.

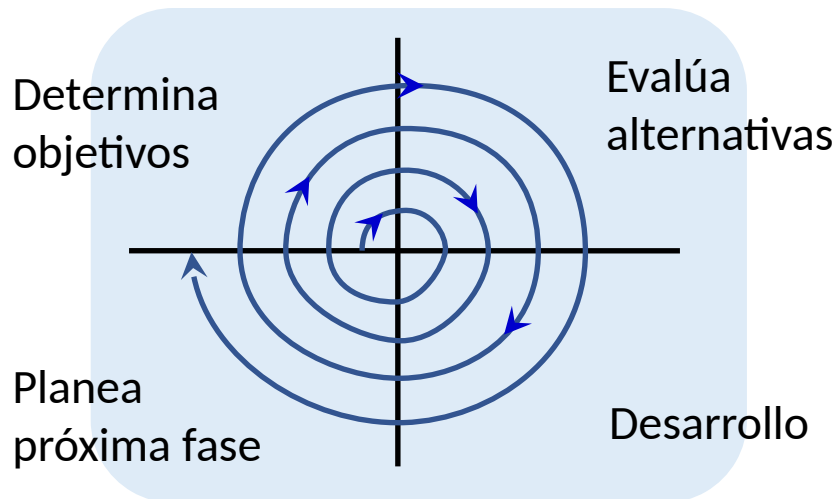


⁴ Fuente: What is a Software Process Model? (<https://www.visual-paradigm.com/guide/software-development-process/what-is-a-software-process-model/>)

Proceso de Desarrollo. Metodología de Desarrollo (3/5)

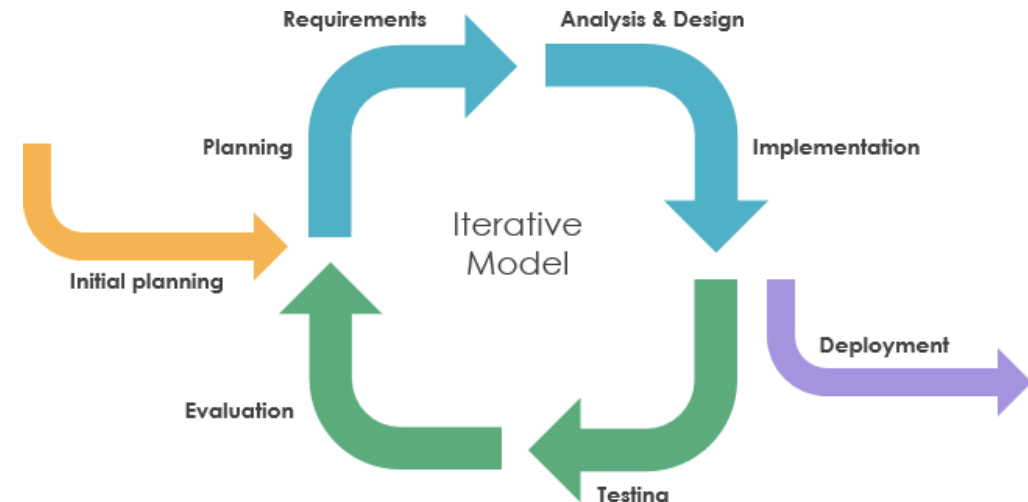
Modelo Espiral

- ❑ Se introdujo para **abordar las deficiencias** (riesgos) del modelo en cascada tradicional.
- ❑ Consta de varias **fases que se repiten en ciclos**, formando una espiral ascendente.
- ❑ Cada ciclo (fase de **SDLC**) representa una **versión mejorada del software**.
- ❑ Admite el **manejo de riesgos**.



Modelo Iterativo

- ❑ Repetición de ciclos (iteraciones) para obtener **resultados progresivos y refinados en el tiempo**.
- ❑ Permiten **retroalimentación continua** y la **adaptación** del software en desarrollo vs. modelos tradicionales que siguen un flujo de trabajo secuencial y lineal.
- ❑ Se **divide el proyecto en pequeñas iteraciones**, cada iteración tiene un **conjunto de objetivos y tareas**.
- ❑ Cada iteración incluye **análisis de requisitos, diseño, implementación y pruebas**.



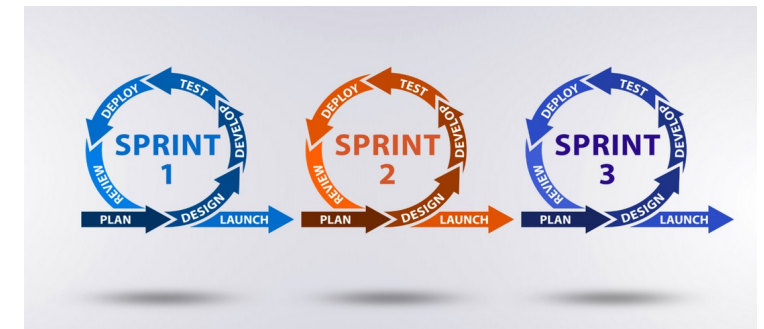
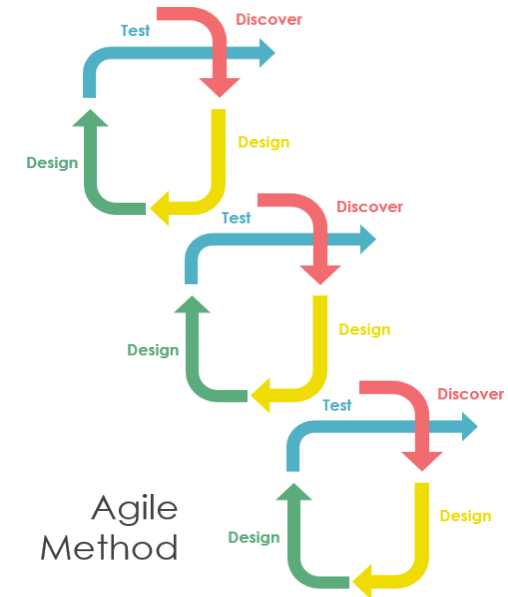
⁴ Fuente: What is a Software Process Model? (<https://www.visual-paradigm.com/guide/software-development-process/what-is-a-software-process-model/>)

Proceso de Desarrollo. Metodología de Desarrollo (4/5)

Método Ágil

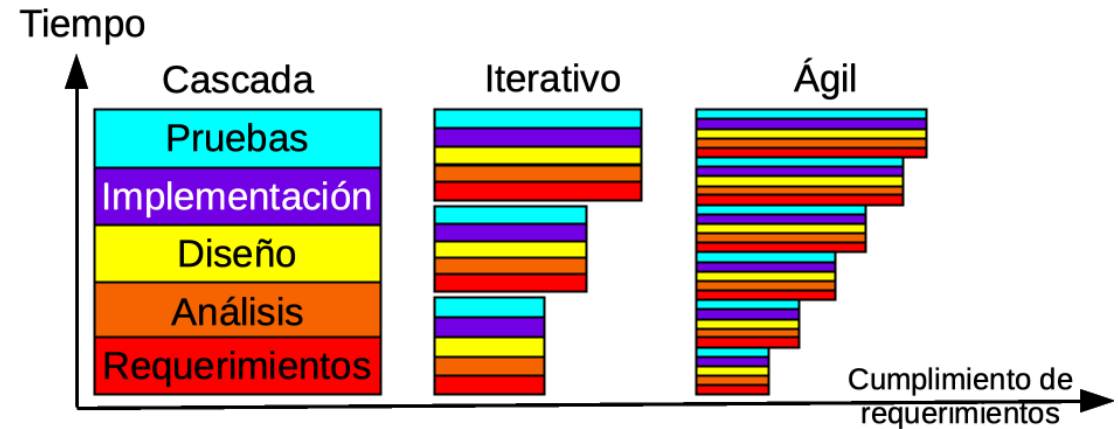
- ❑ Enfoque iterativo para el desarrollo de software (aún se alinea con el SDLC). Cada fase es breve.
- ❑ Es probablemente la metodología más popular utilizada en el desarrollo de software moderno.
- ❑ Los equipos trabajan en ciclos, o *sprints*, que duran de 1 a 4 semanas.
- ❑ Las pruebas unitarias (*unit testing*) ocurren en cada sprint para reducir el riesgo de falla.
- ❑ Al final de cada sprint, se publica una parte del código de trabajo en una reunión llamada “demostración de sprint” donde las partes interesadas pueden ver la nueva funcionalidad y dar feedback.
- ❑ Se centra en ráfagas de desarrollo rápidas y breves, vs. métodos tradicionales, como *Waterfall* y *V-Model* los cuales se centran en todo el producto antes de solicitar los comentarios de los clientes.

¡ÁGIL: no significa rápido, las iteraciones son cortas!



Proceso de Desarrollo. Metodología de Desarrollo (5/5)

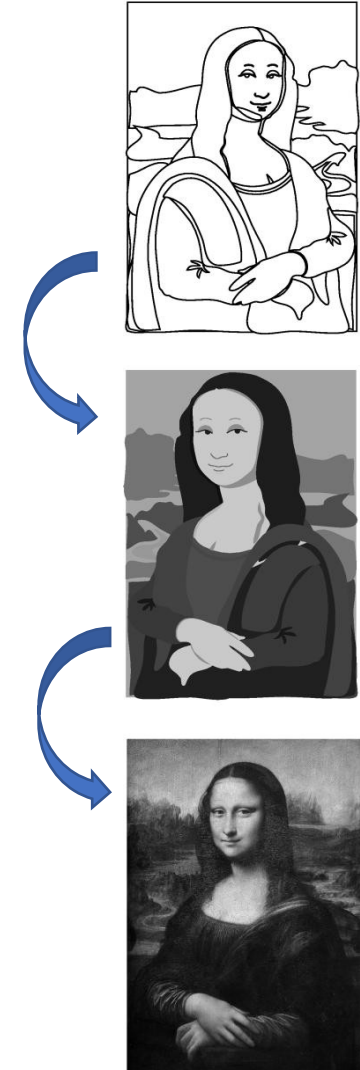
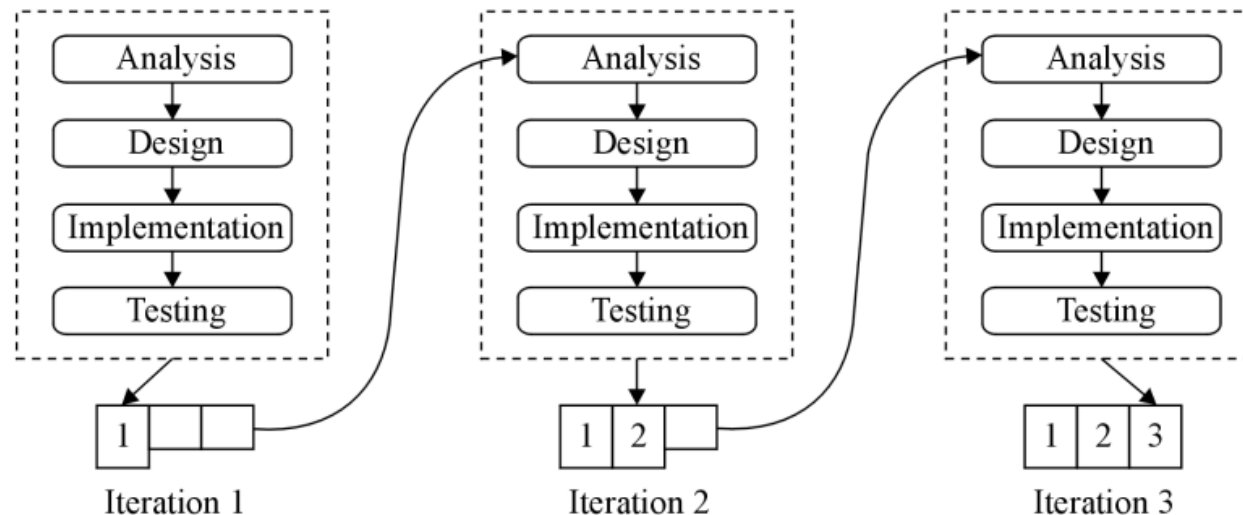
¿Cuál utilizar?
¿Cuál es mejor?



- ☐ Depende del tipo de producto de software a construir.
- ☐ Si el producto tiene un objetivo muy específico, requerimientos bien definidos y muy poca tendencia al cambio, en donde la gestión se pueda predecir, entonces, es podemos usar *Waterfall*.
- ☐ Si el producto tiene alta incertidumbre, en donde constantemente se están realizando prototipos y se necesita entrega continua de valor, lo mejor es usar metodologías ágiles de desarrollo.
- ❖ Ejemplo de uso metodología **Ágil**: en **emprendimientos**; proyectos con **requisitos cambiantes**; desarrollo de software en **ciclos cortos** (e.j., Proyecto de ELO329); equipos colaborativos y autoorganizados; proyectos con **alta incertidumbre**.
- ❖ Ejemplo de uso metodología **Iterativa**: **ciclos de desarrollo cortos** y **entregas incrementales**; desarrollo de proyectos complejos o de gran escala.
- ❖ Ejemplo de uso metodologías **Cascada**: **requisitos bien definidos**, proyectos con **restricciones de tiempo y presupuesto** estrictas; **documentación y trazabilidad exhaustivas**; **entornos regulados** o con **requisitos legales** específicos; no hay ciclos, cada paso debe ser seguro (e.j., viaje a la Luna).

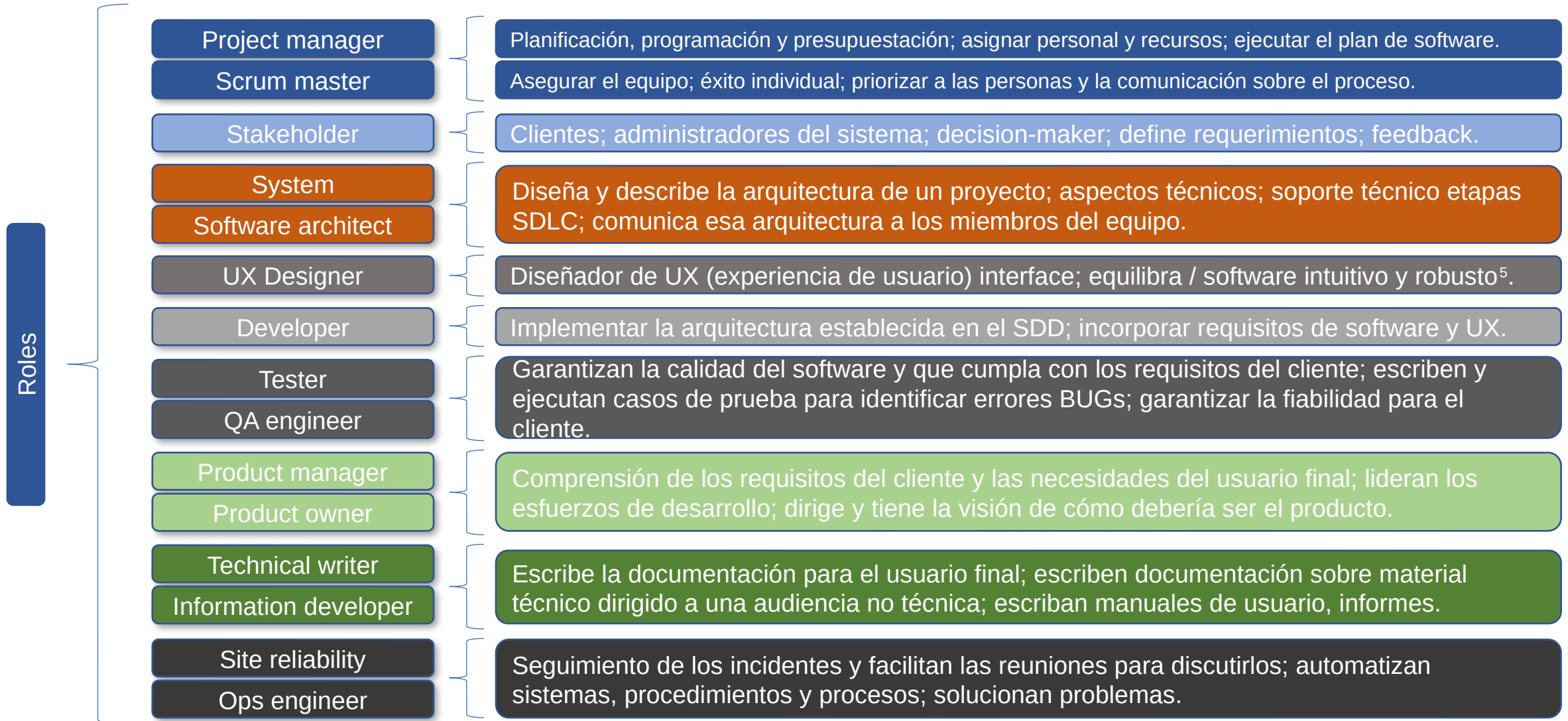
Desarrollo Iterativo Incremental

- ❑ Esta idea es la base de varios métodos de desarrollo de software.
- ❑ La idea es desarrollar el sistema siguiendo etapas incrementales caracterizadas por generación de sucesivas versiones que van abarcando requerimientos hasta completar el sistema.
- ❑ **Iterativo:** cada vez re-visitamos las etapas del modelo en cascada.
- ❑ **Incremental:** con frecuencia integramos los avances para generar una **versión con sentido para el cliente**.



^{5,6}Fuentes: An iterative approach in development of the student information system: Lessons learned (<https://doi.org/10.1109/ICAT.2009.5348414>)
A customisable and responsive design online booking system (<http://dx.doi.org/10.5121/ijcsit.2017.9506>)

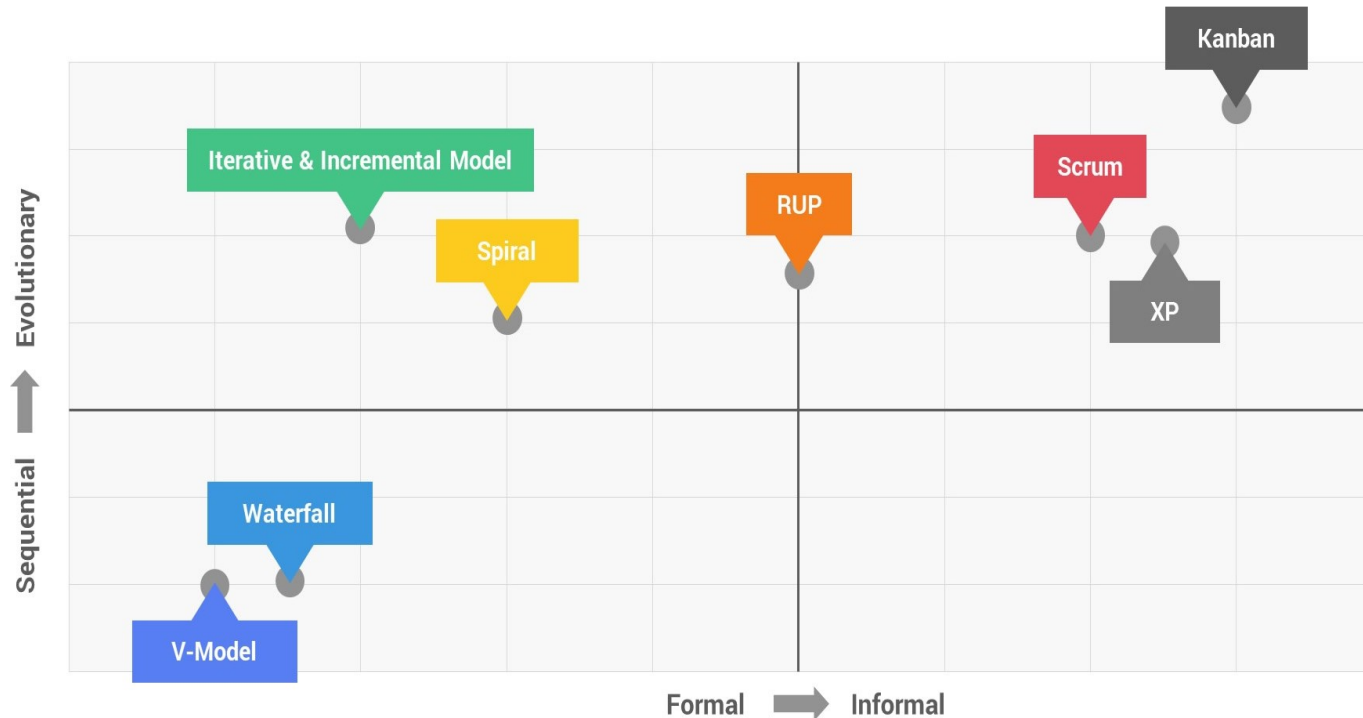
Roles en Proyectos de Ingeniería de Software



⁵ Ejemplo: Laboratorio UX, USM (<https://labux.inf.utfsm.cl/>)

Modelos Populares - Proceso de Desarrollo (SDLC)

Types of Popular SDLC Models⁶



Hoy > 70% empresas usan enfoque Ágil en sus proyectos de TI⁷



Tradicionales

- ☐ Cascada
- ☐ V-Model
- ☐ Iterativa-Incremental
- ☐ Espiral
- ☐ RUP (Rational Unified Process)



Grupo Ágil

- ☐ Scrum
- ☐ XP (Xtreme Programming)
- ☐ Kanban
- ☐ Lean
- ☐ SAFe

⁶ Fuente: ScienceSoft. The outline of popular SDLC models (<https://www.scnsoft.com/blog/software-development-models>)

⁷ Fuente: Project Management Institute. Success Rates Rise (<https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf>)

Desarrollo Ágil de Software (1/3)

70%

Organizaciones que usan ágil⁸

25%

Empresas Manufactureras⁸

77%

**Organizaciones adoptaron Scrum
(ref. Totales usan Ágiles)⁸**

28%

**Más de éxito que métodos
tradicionales⁹**

47%

**Tasa de falla de transformaciones
de empresas a prácticas Ágiles¹⁰**

- 1. Levantamiento de requerimientos y especificaciones:**
 - Reconocer que evolucionarán.
 - Importante: pensar qué áreas son más probable de cambiar.
 - 2. Diseño:**
 - Diseñar considerando que habrán cambios.
 - 3. Implementación:**
 - Lo más crítico primero.
 - No hacer optimizaciones, la implementación probablemente cambiará.
- ☐ **Iterar 1,2,3:**
- Mostrar el prototipo al usuario o cliente.
 - Actualizar los requerimientos, diseño, etc.

⁸ Fuente: Project Management Institute. Success Rates Rise (<https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf>)

⁹ Fuente: Price Waterhouse Coopers. Agile Project Delivery Confidence (<https://www.pwc.com/gx/en/actuarial-insurance-services/assets/agile-project-delivery-confidence.pdf>)

¹⁰ Fuente: Forbes. Five Aspects Of A Successful Agile Transformation For Your Enterprise (<https://www.forbes.com/sites/forbestechcouncil/2022/06/03/>)

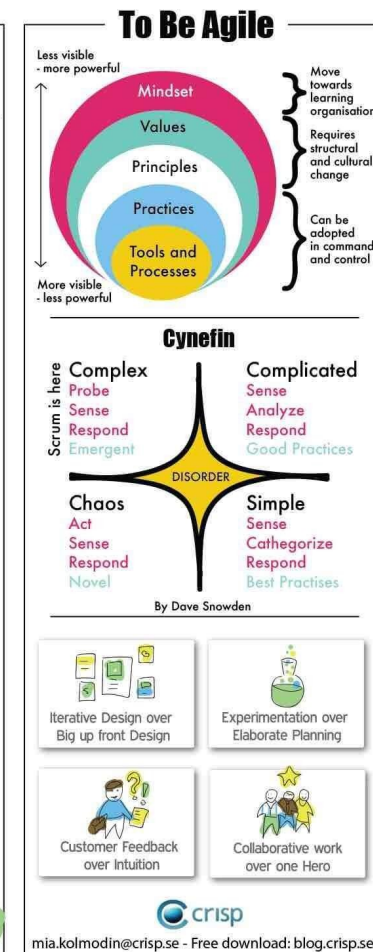
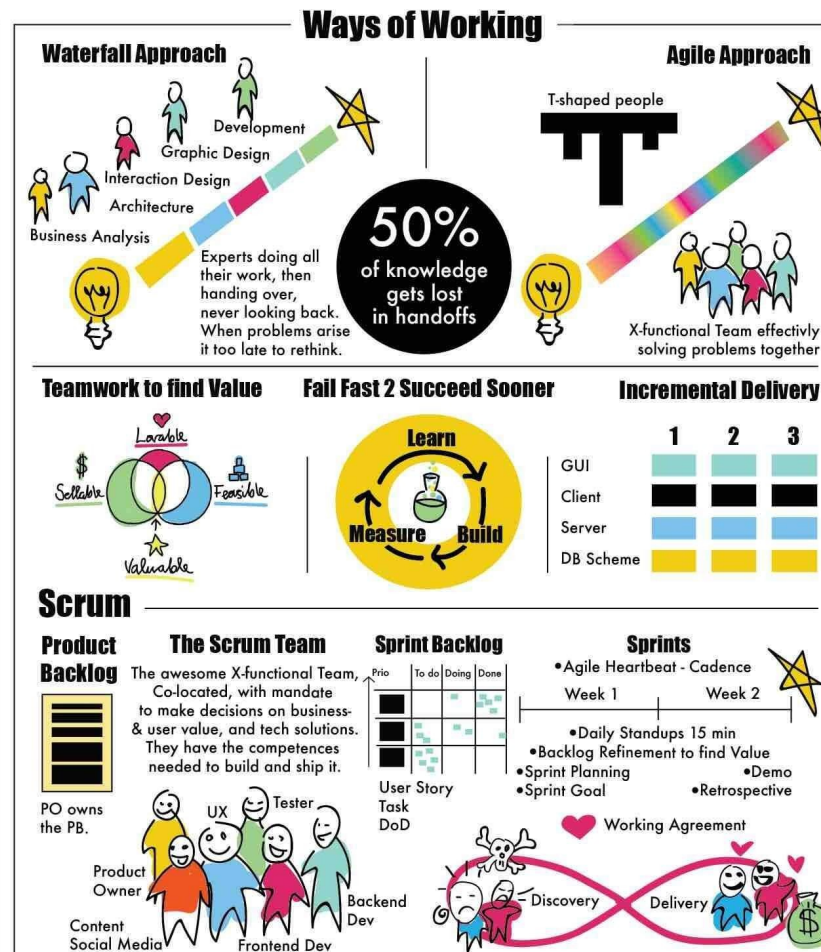
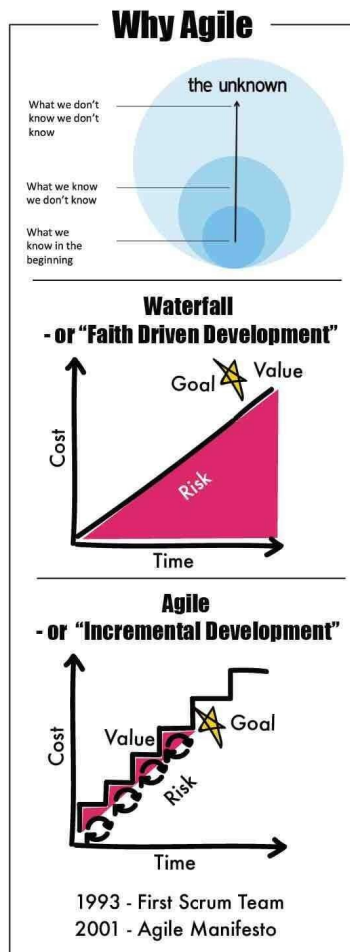
Desarrollo Ágil de Software (2/3)

Agile in a Nutshell with a spice of Lean

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.



Desarrollo Ágil de Software: Manifiesto (3/3)

Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

- ☐ **Individuos e interacciones** sobre procesos y herramientas
- ☐ **Software funcionando** sobre documentación extensiva
- ☐ **Colaboración con el cliente** sobre negociación contractual
- ☐ **Respuesta ante el cambio** sobre seguir un plan

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.”

Agile Manifiesto

Resumen



- ❑ Definición de Ingeniería de Software. Responsabilidades de un ingeniero de software.
- ❑ El uso del SDLC (Proceso de Desarrollo de Software – Life Cycle).
- ❑ Procesos comunes de Ingeniería de Software: requerimientos, diseño, codificación, pruebas, lanzamiento y documentación.



- ❑ Un SRS (software requirement specifications) es un documento que almacena las funcionalidades que debe realizar el software.



- ❑ Diferentes metodologías de Software: Waterfall, V-Model, Ágil, otras, capaces de implementar el ciclo de vida de desarrollo de software.
- ❑ Roles involucrados en un proyecto de Ingeniería de Software.

Lección Importante



- ❑ El tiempo es independiente del contexto: “ahorrar una semana la comienzo de un proyecto es tan bueno como ahorrarla al final”. “Una semana es una semana”.
- ❑ Es mucho más fácil ahorrar tiempo al inicio del proyecto (cuando los “entregables” son menos claros).



¿Cómo usted aplica esto al proyecto de la asignatura?

¿Cuánto desearía tener algunos días “libres” antes del plazo?