INTRODUCCIÓN

CS8105
Ingeniería de Software

02/09/2021

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

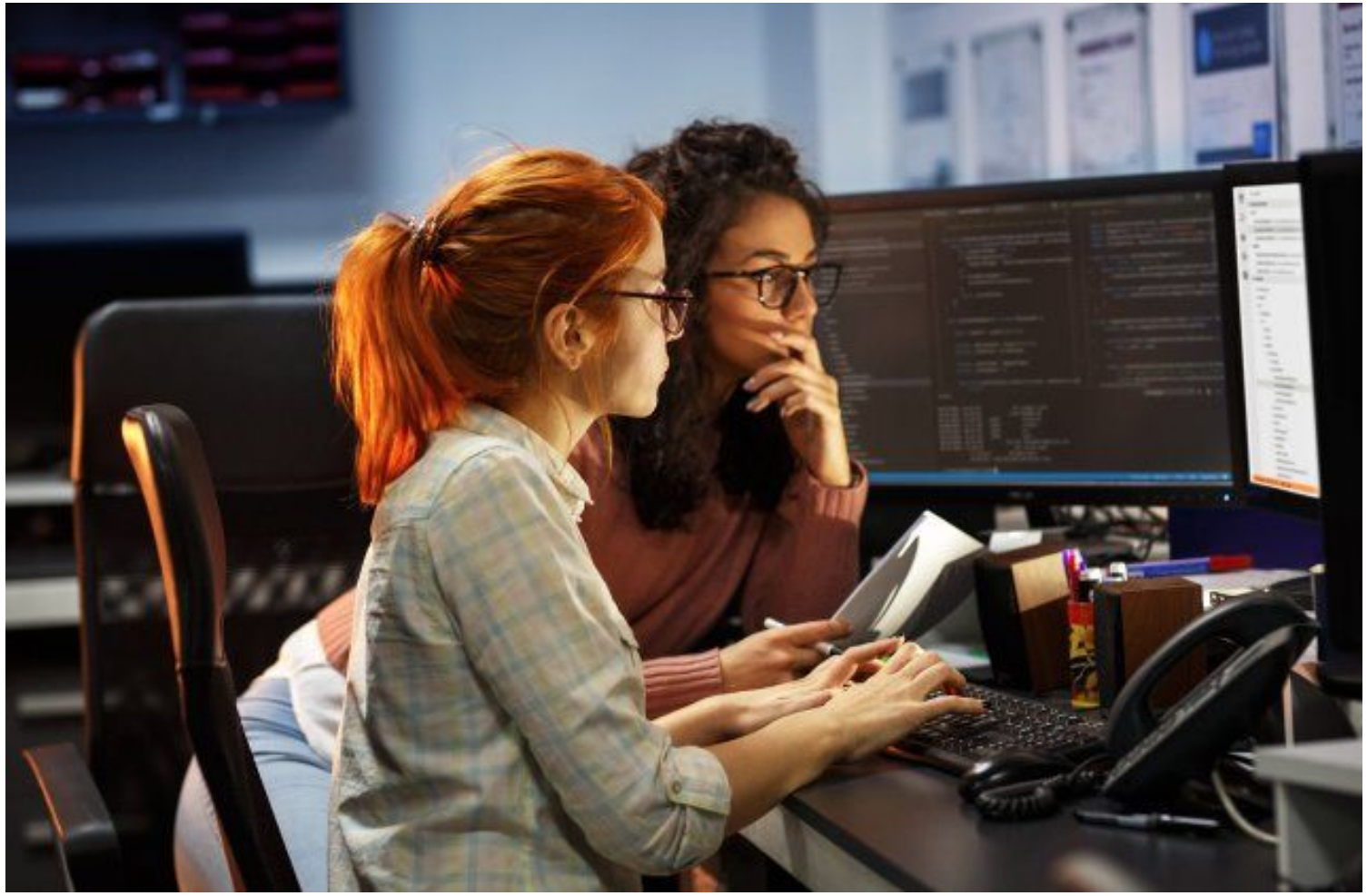
Índice

- Acerca del curso
- Herramientas
- Principales Tópicos
- ¿Qué es Ingeniería de software?

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

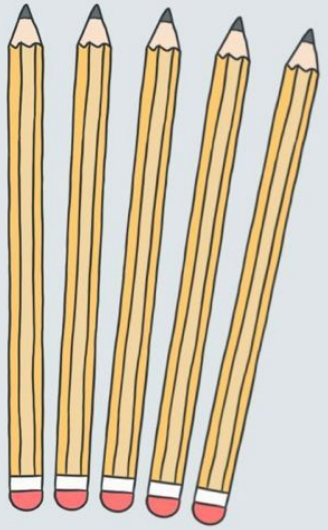
1 ACERCA DEL CURSO

Teoría = 4 horas



SHOW YOUR WORK

NOT YOUR TOOLS



Los resultados del aprendizaje

- Construir software con estándares de calidad que satisface las necesidades de un proyecto de desarrollo avanzado usando herramientas y metodologías adecuadas.
- Familiarizarse con el proceso de construcción y modelado de software mediante el uso de herramientas CASE.
- Los estudiantes deben poder seleccionar arquitecturas y plataformas de tecnología ad-hoc para escenarios de implementación.
- Aplicar modelos basados en componentes para garantizar variables como la calidad, el costo y el tiempo de comercialización en el proceso de desarrollo.
- Proporcione a los estudiantes las mejores prácticas para la verificación y validación de software.
- Planificar el trabajo del equipo de desarrollo de software coordinando con los miembros del equipo y stakeholders.

Sistema de Evaluación

	TEORÍA (T)
EVALUACIÓN *La ponderación de la evaluación se hará si ambas partes están aprobadas	Evaluación Continua (C1) (10%) Evaluación Continua (C2) (10%) Examen Parcial (E1) (20%) Examen Parcial (E2) (20%) Proyecto Parcial (P1) (20%) Proyecto Final (P2) (20%)
	100%

Objetivos del Curso

- Realizar trabajo de **ingeniero**:
  - Resolver **un problema** real a través de un producto de software.
  - Desarrollar **un producto** trabajando en equipo.
  - Asumir **roles** dentro del equipo.
  - Lidiar con el cliente y con los miembros del equipo.
  - Implantar la solución y **evaluar** su impacto.

Metodología

- Los alumnos **trabajarán en equipos** de 5 o 6 personas.
- A cada equipo **se le asignará un proyecto**, que deberá desarrollar durante el semestre.
- Cada equipo **puede negociar** los roles de sus miembros con el profesor.
- Los roles posibles son:
  - Administrador del Proyecto.
  - Analista.
  - Diseñador-Implementador.
  - Tester.
- Los roles de un miembro pueden cambiarse al final de cada fase, pero se **debe tener el OK** del profesor.
- **Al finalizar cada etapa del proyecto** (incremento), cada equipo de trabajo, deberá entregar una **versión implantada** del sistema, ajustándose a los requisitos especificados por el cliente.

Proyecto

- La nota del cliente considera:
  - Calidad / funcionalidad del producto obtenido.
  - Compromiso del equipo para con el proyecto.
  - Proactividad del equipo.
  - Comunicación y coordinación con el cliente.

Reglas de evaluación

- Si más del **50 % de los miembros del equipo** de trabajo está **de acuerdo con expulsar a uno de sus integrantes**, puede hacerlo. Para ello deberá contar con el consentimiento del profesor.
- Si el alumno es expulsado, la responsabilidad del acto será asumida por el profesor.
- El expulsado tendrá un **0** como nota final del proyecto, y reprobará el curso.
- El **administrador** de cada proyecto puede recomendar al profesor, **la expulsión justificada de un miembro del equipo** de trabajo, sin necesidad de que la mayoría de los miembros del equipo estén de acuerdo.

Roles: Administrador del Proyecto

- Es el principal responsable proyecto (Decision maker). Entre sus responsabilidades está:
  - Delimitar el Alcance del Sistema (con los analistas).
  - **Planificar/replanificar** y Administrar el Proyecto. Incluyendo el plan de pruebas e implantación junto con el implementador.
  - **Coordinar** el trabajo de los distintos miembros del equipo.
  - Interactuar con el Cliente.
  - **Velar por el cumplimiento** de los objetivos, plazos y costos comprometidos.
- Este es uno de los roles más críticos dentro de cualquier proyecto de desarrollo de software.

Debe ser organizado, proactivo y tener buenas capacidades de comunicación y coordinación.

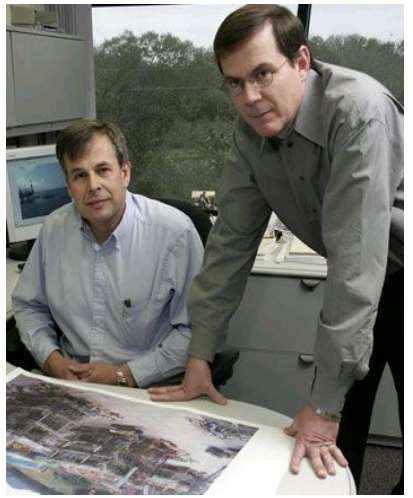
Roles: Analista

- Es el encargado de levantar y especificar los requisitos del sistema a desarrollar. Entre sus tareas está:
  - **Identificar y entrevistar** a clientes y usuarios.
  - **Delimitar el alcance** del sistema (con el AdP).
  - **Desambiguar** los requisitos.
  - **Generar el documento de requisitos** que incluye los requisitos de software y de usuario, dentro de los plazos comprometidos.
  - **Apoyar al Tester** en la especificación de las pruebas de sistema y de usuario.
  - **Velar** porque el **diseño** cumpla con los requisitos (junto con el tester).
  - **Velar** porque el **producto fina**l cumpla con los requisitos (junto con el tester).

- *Debe trabajar rápido y a conciencia.*
- *Debe ser detallista.*

Roles: Diseñador Implementador

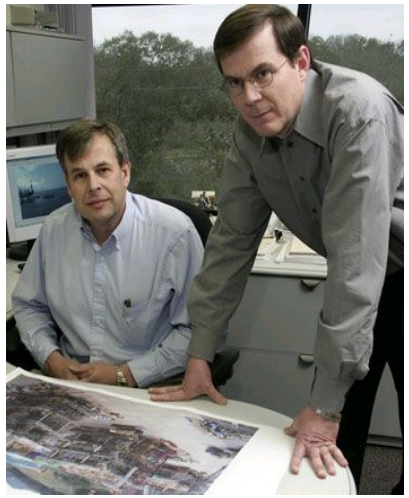
- Es el encargado de generar el diseño del front-end y back-end del sistema. Entre sus funciones está:
  - **Generar el diseño arquitectónico y diseño detallado** del sistema (DD-Documento de Diseño), basándose en los requisitos. El diseño debe ser implementable.
  - **Diseñar e implementar las interface**s del sistema para chequear los requisitos entregados por el cliente.
  - **Implementar el sistema**.
  - **Validar (junto con los testers) los prototipos** con clientes y usuarios pertinentes.



Debe ser creativo (pero realista), inteligente (no trabajar de más), y…. no debe dejar las cosas para último momento.

Roles: Tester

- Es el encargado de asegurar la calidad de cada uno de los productos (documentos, prototipos, etc). Entre sus tareas está:
  - **Validar** la **calidad** de los productos del proyecto, partiendo por las interfaces de usuario.
  - Coordinar las revisiones de los productos del proyecto.
  - **Generar los informes** post-revisión.
  - Realizar un seguimiento de la**s falencias identificadas**.
  - **Velar por la completitud y exactitud** (no ambigüedades) de los documentos.
  - **Velar por la calidad del producto final** (cumplimiento de los requisitos).
  - **Especificar las pruebas** a realizar (con analistas).



Debe ser muy detallista y constructivo …. Ojalá nunca tenga nada que decir.

Es uno de los roles que más contribuye al éxito del proyecto (junto al AdP).

2 HERRAMIENTAS

python™

GitHub

3 PRINCIPALES TÓPICOS

¿Qué es Software?

- Software no son sólo los programas que utilizamos
  - Código fuente NO no es lo único que debemos generar
- Los modelos que describen el problema que resuelven, como fue resuelto, los datos del programa y la documentación para ocupar, instalar o administrar los programas **también** son software.

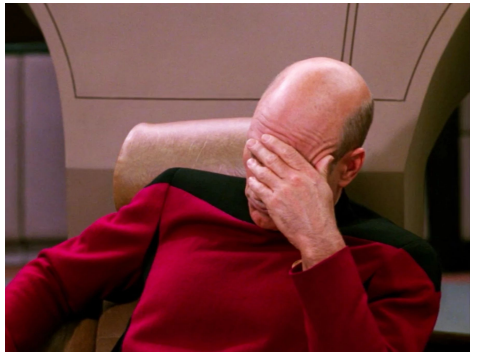
UTEC

Ejemplo Doméstico

Situación cotidiana:

- ❏ El software se desarrolla para satisfacer una necesidad
- ❏ Veamos como un especialista en una materia satisface una necesidad

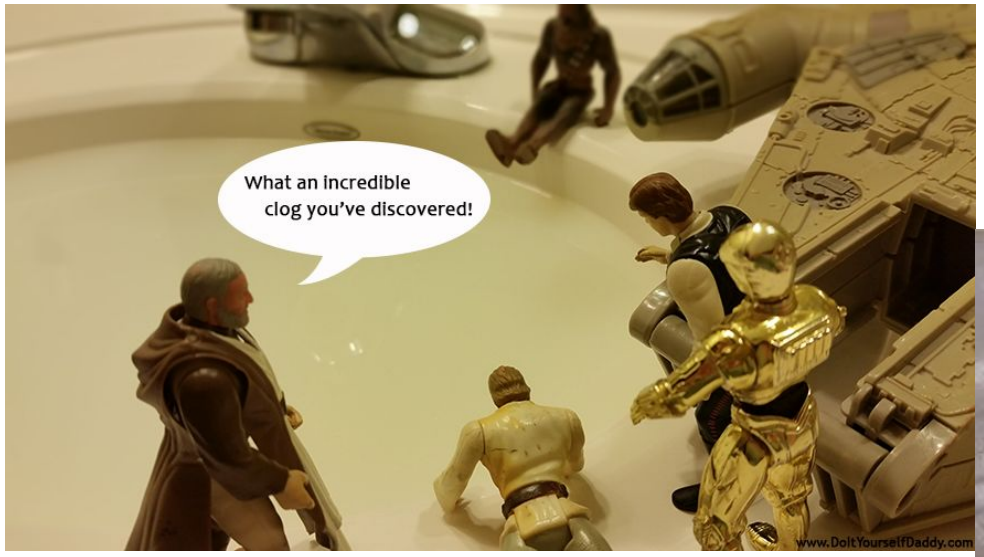
- Juan, Gásfiter
- “Hola Capitán,cuénteme
  - ¿Cuál es su problema?



Picard, Cap. USS Enterprise

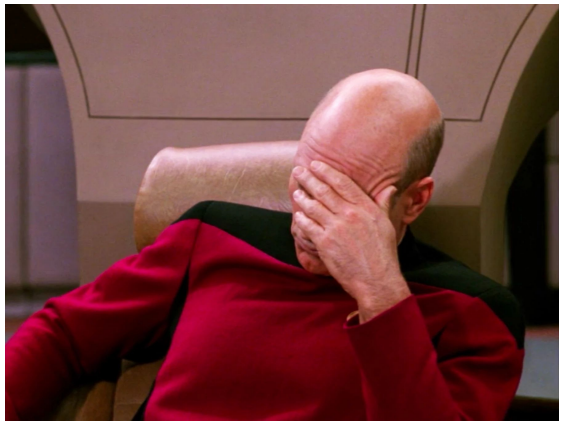
“Data estaba jugando con los monitos en el baño y ahora el agua no se va.”

Ejemplo Doméstico



Ejemplo Doméstico

"Ahh… los 'monitos' son de papel… de plástico"



"De plástico joven, como el Darth Vader ese"

"Comprendo capitán, entonces usted necesita desbloquear la obstrucción del conducto evacuador del baño, que es producido por el elemento sólido"

Ejemplo Doméstico

- ¿Qué ha hecho Juan?

  - Juan ha **escuchado** las necesidades del Capitán Picard

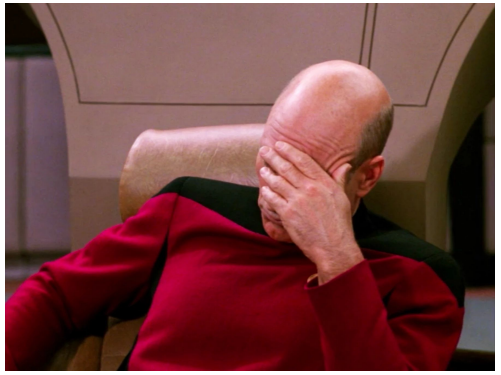
  - Juan ha **preguntado** lo que necesita, para saber la gravedad del problema

  - Juan ha **abstraído** elementos no importantes del problema (no le importan Data ni Darth Vader)

  - Basado en lo anterior, Juan ha **descrito lo que necesita** el Capitán Picard

- Juan ha **analizado el problema** y lo que necesita el Capitán Picard.

Ejemplo Doméstico



“¿Y lo puede arreglar, joven?”

“Por supuesto, Capitán”

“Tendré que hacer un corte en el conducto de 10 cm del baño y otro a 20cm del suelo para sacar el elemento, y luego volveré a poner el mismo tubo, Por último sellaré todo con silicona”

Ejemplo Doméstico

- ¿Qué ha hecho Juan?
  - Juan ha **pensado** en cómo resolver el problema
  - Juan ha pensando en qué puede **reutilizar** en su solución propuesta. (podría haber usado otro tubo y cobrar más, pero él sabe que con el mismo funciona, y Juan es muy ético.)
  - Juan ha d**escrito su solución** en términos de los elementos que interviene en la solución
- Juan ha **diseñado una solución**

Ejemplo Doméstico

“Voy a entrar a picar”

“Juan está **implementando la solución**”

Ejemplo Doméstico

“Listo Capitán, pruebe nomas lo probé y funciona”

“Funciona perfecto, joven”

“Muy bien Capitán, esta es mi tarjeta, cualquier problema con el arreglo me avisa”

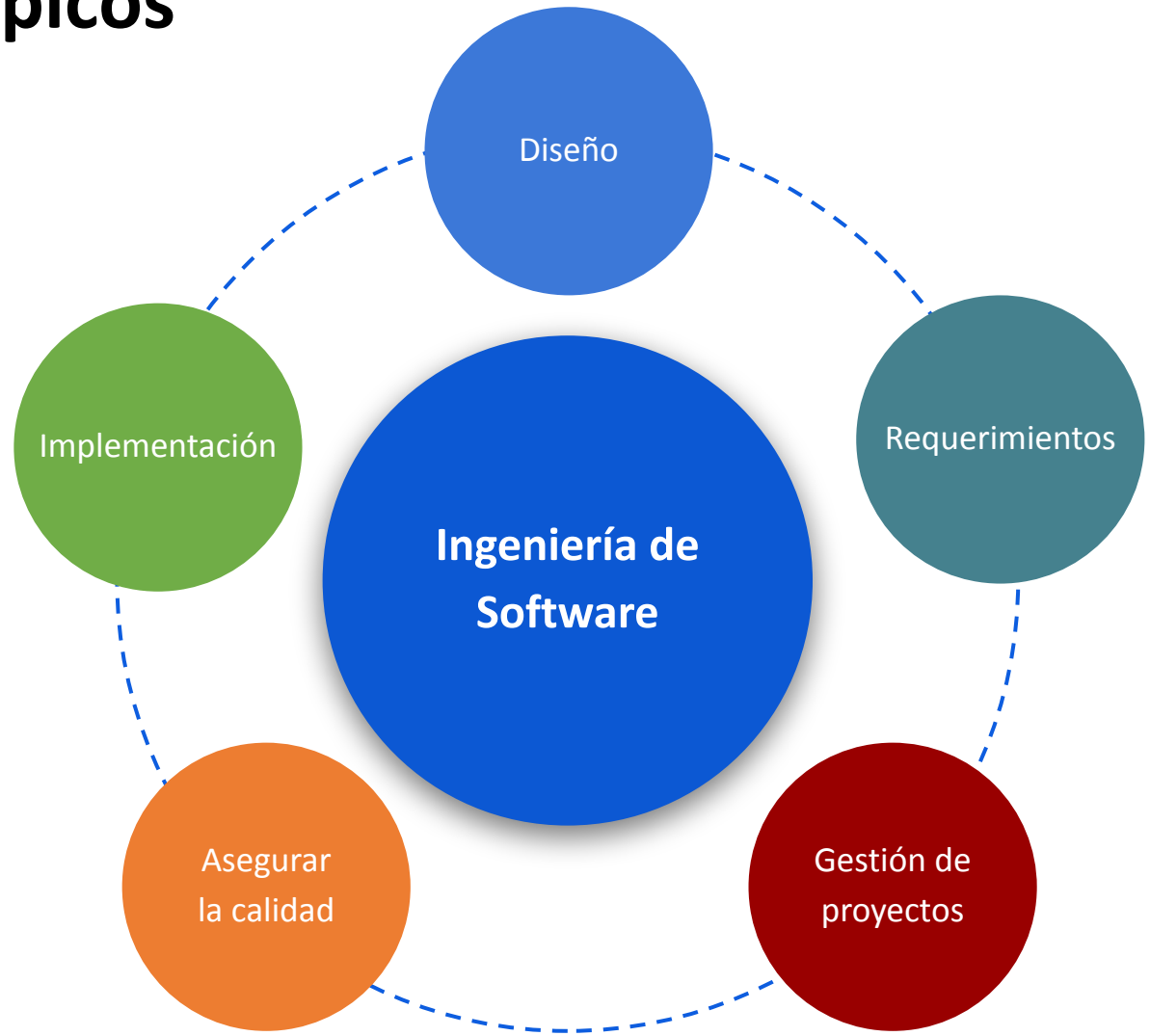
Ejemplo Doméstico

- ¿Qué ha hecho Juan?
  - Juan ha **probado** su solución
  - Juan ha ofrecido sus servicios para una eventual **mantención** de la solución.

Ejemplo Doméstico

- En resumen:
  - Juan **analizó** el problema ⇒ “Requisitos”
  - Juan **diseñó** la solución ⇒ “diseño”
  - Juan **implementó** la solución ⇒ “implementación”
  - Juan **probó** la solución ⇒ “Aseguramiento de la calidad”
  - Juan ofreció realizar la **mantención** de la solución
- El enfoque usado por Juan es adecuado también para resolver un problema de software.
- ¿Alguna crítica hacia Juan?
  - El uso de lenguaje excesivamente técnico
    - tal vez el Capitán no se convenza de que él entendió el problema.
    - tal vez el Capitán no se entiende como va a resolver.

Principales tópicos

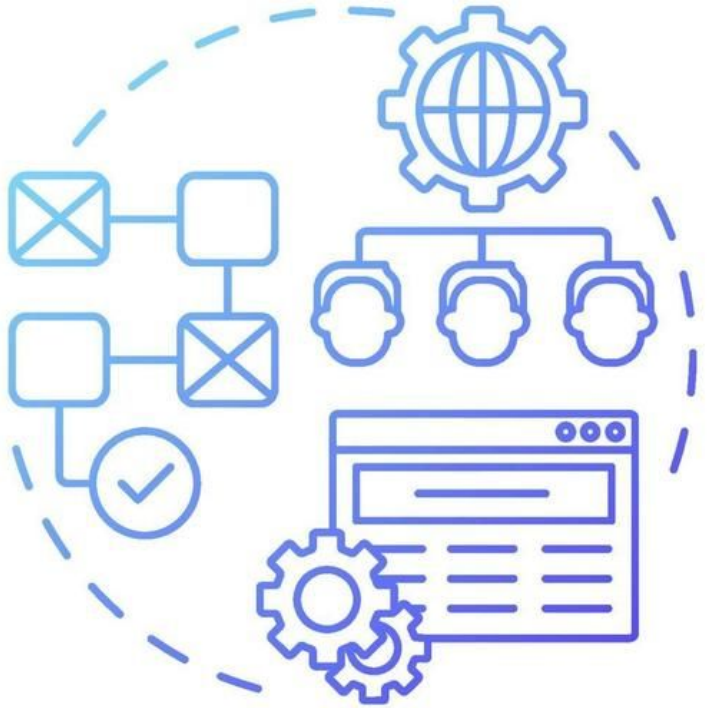


Requerimientos

- Introducción
- Requerimientos no funcionales
- Priorización de requerimientos
- Requerimientos de calidad
- Recopilación de requerimientos
- Especificar requerimientos

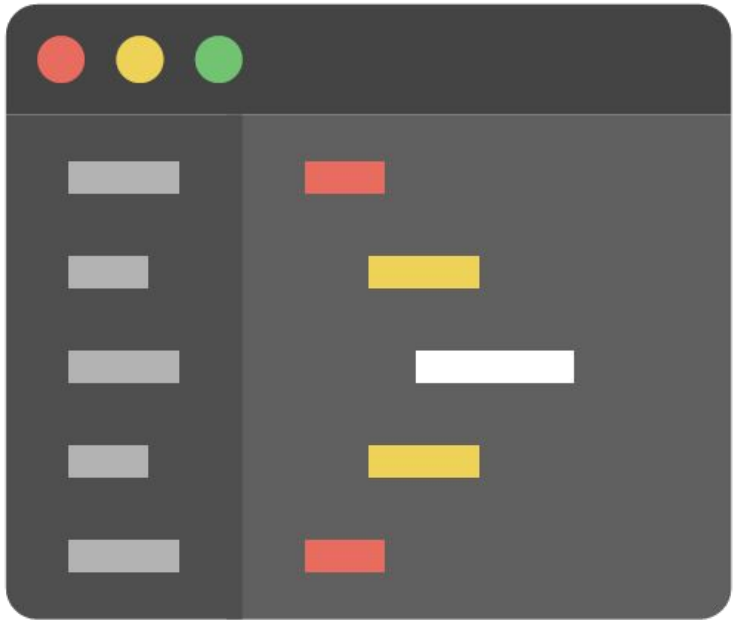
Diseño

- Diseño de Software
- Fundamentos de diseño
- Modelado
- Arquitectura de software
- Patrones de diseño de software



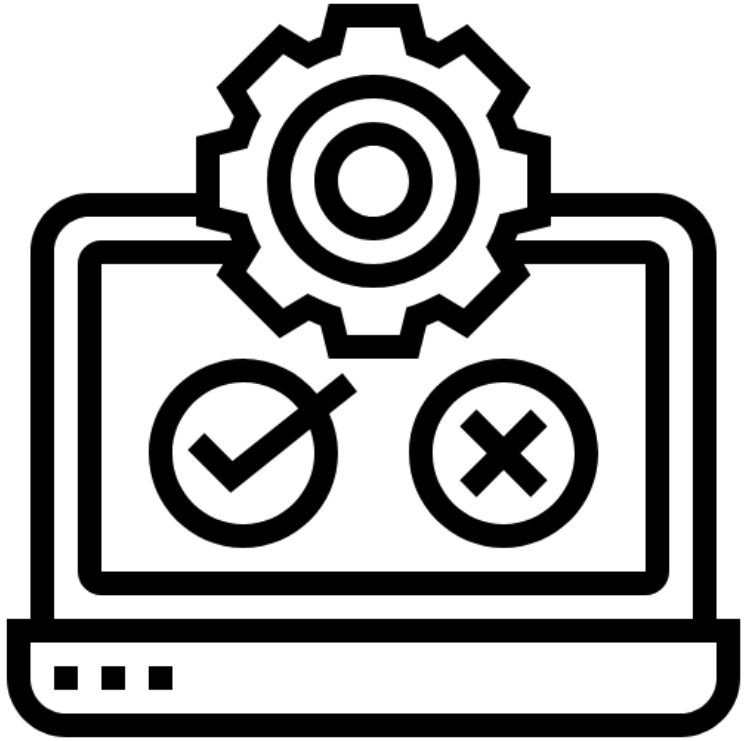
Implementación

- IDEs
- Calidad de código
- Refactorización
- Documentación
- Manejo de errores
- Integración
- Reutilización



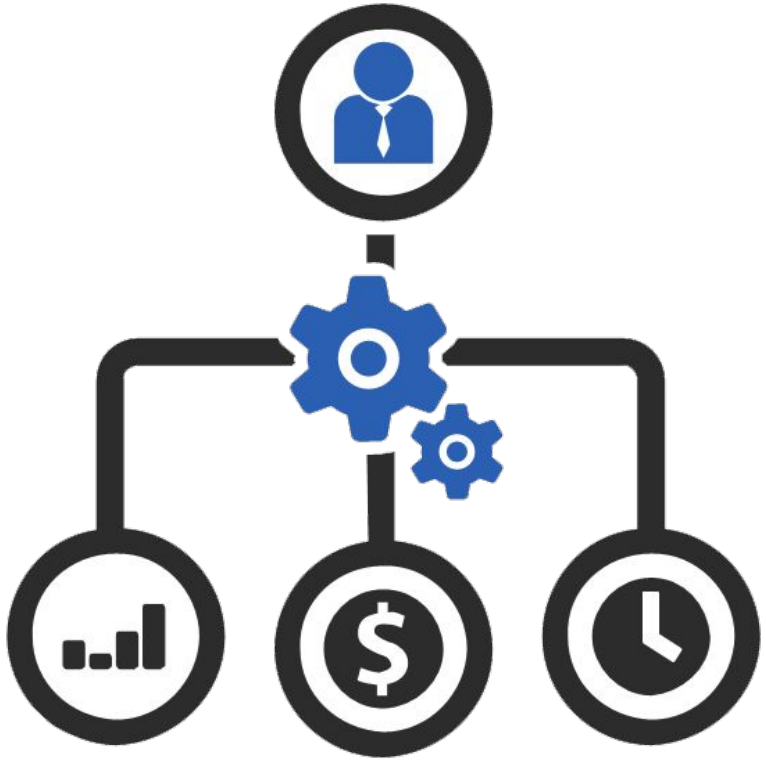
Aseguramiento de la calidad

- QA
- Pruebas
- Tipos de pruebas
- Automatización de pruebas
- Cobertura de prueba
- Inyección de dependencia
- TDD (Test-Driven Development)



Gestión de proyectos

- Control de revisión
- Planificación de proyectos
- SCRUM
- Jira Software
- Confluence



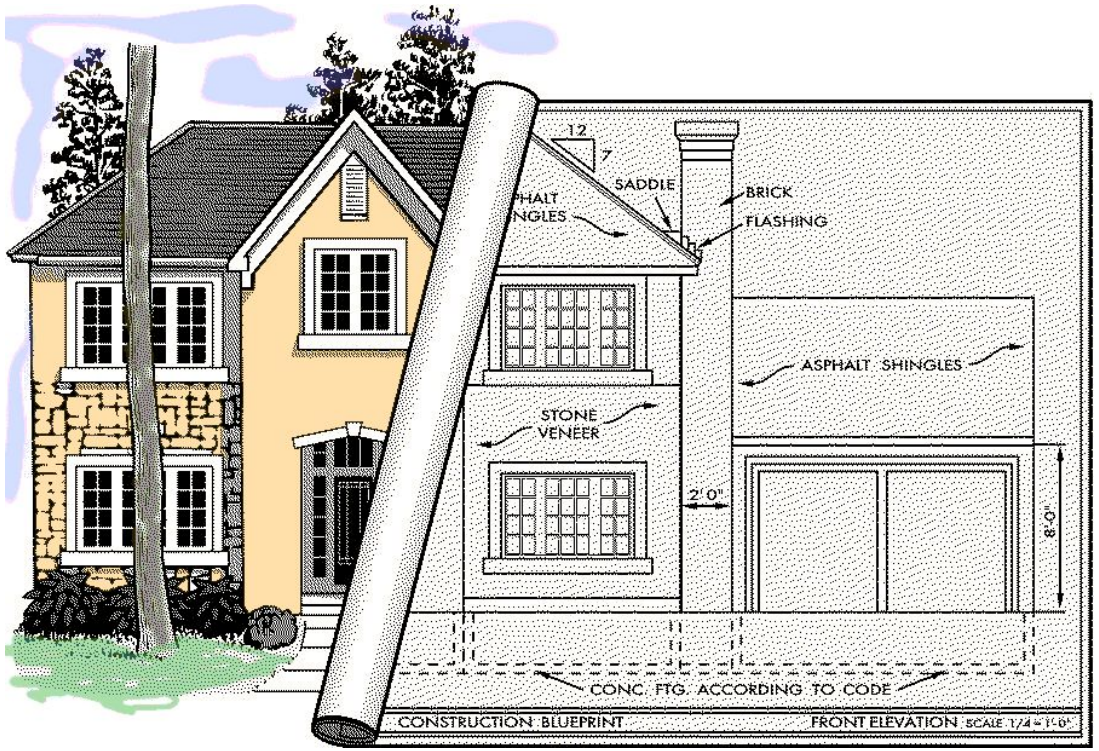
ATLASSIAN

UTEC

4 INGENIERÍA DE SOFTWARE

Los resultados de aprendizaje

- Puede explicar qué es la ingeniería de software

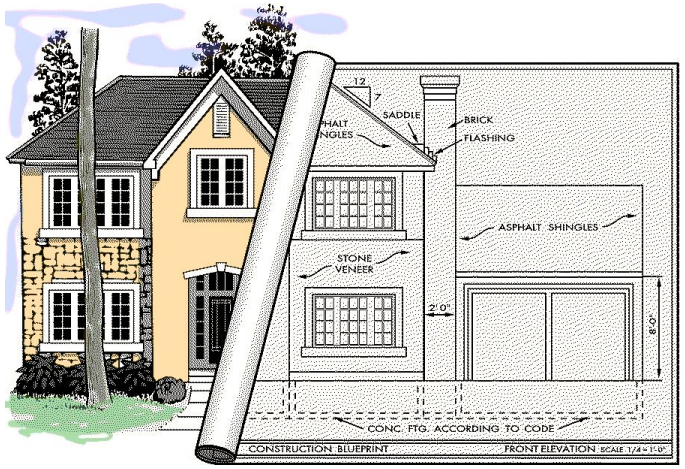


Si usted quiere contratar la construcción de su casa:

¿Qué le exigiría al Producto (casa)?

¿Qué le exigiría al Proyecto?

Ingeniería de Software



-¿El proceso que apoya el ciclo de vida de una casa, es el mismo para cualquier tipo de casa?

Ingeniería Civil no es sinónimo de **Construcción de Casas...**

-**Ingeniería de Software** tampoco es sinónimo de **Desarrollo de Software**....

¿Cuánta ingeniería se requiere?

- Depende del **rango de complejidad** del proyecto y del producto.
- Esto va desde lo muy simple, hasta lo muy complejo.
- **En su versión más simple**, podría ser llevada a cabo por una persona.
- Requiere al menos:
  - Modelamiento mínimo
  - Proceso guía
  - Herramientas simples

Ingeniería Civil VS Ingeniería de Software

Construcción

- Visible, tangible
- Se desgasta con el tiempo
- El cambio está limitado por restricciones físicas
- Crear una copia exacta de un edificio es imposible
- Difícil de mover
- Muchos trabajadores poco calificados siguen procedimientos probados

Software

- Invisible, intangible
- No se desgasta
- El cambio de sistema operativo no está limitado por tales restricciones. Simplemente cambie y vuelva a compilar.
- Cualquier cosa de copias exactas se puede hacer con un costo casi nulo.
- Entregado fácilmente de un lugar a otro.
- No hay trabajadores poco calificados involucrados.

Programas & Software

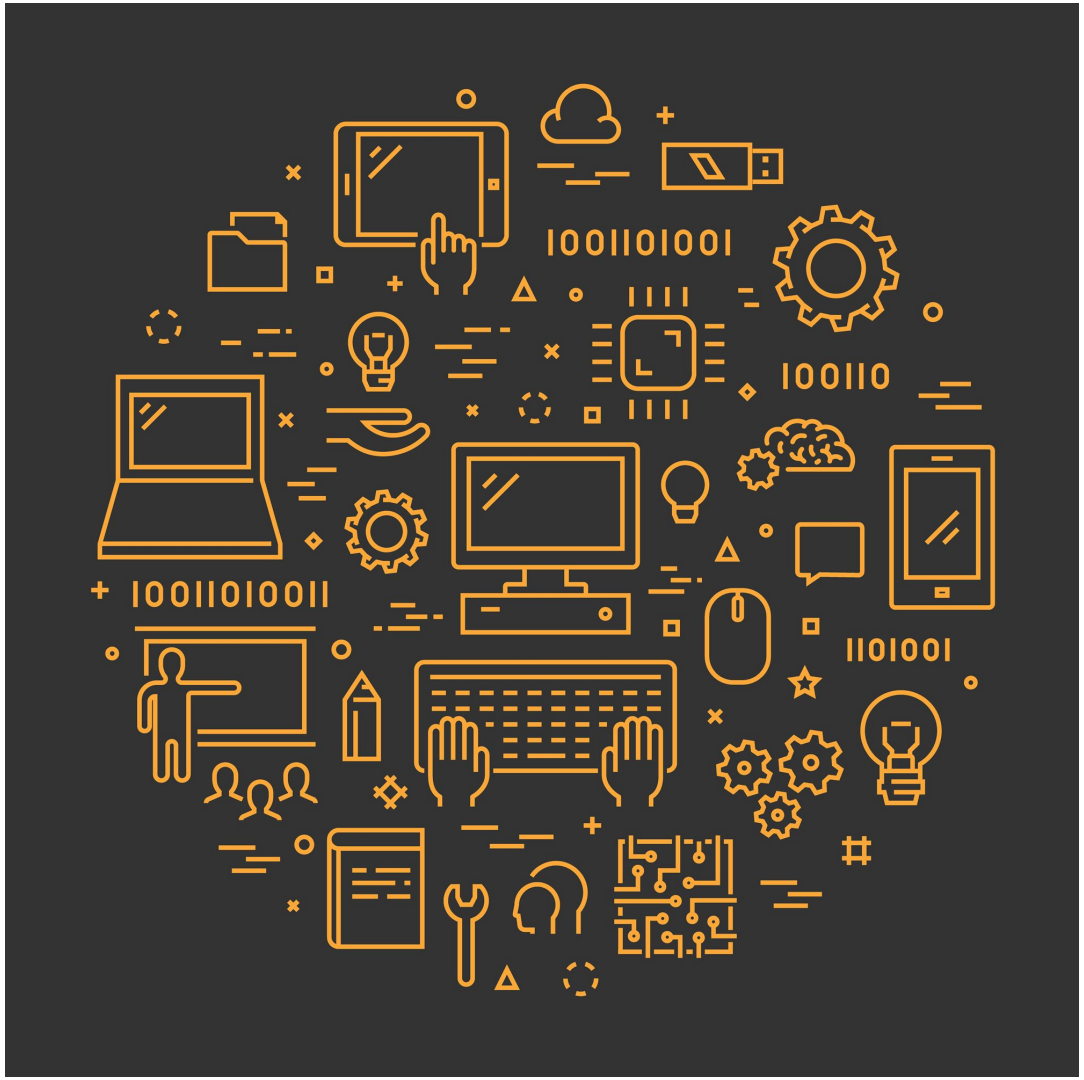
PROGRAMA

- De tamaño pequeño
- El propio autor es alma de usuario
- Desarrollador único
- Adoptar el desarrollo
- Carece de una interfaz adecuada
- Gran documento adecuado

SOFTWARE

- De gran tamaño
- Gran número
- Desarrollado por un **Equipo**
- Desarrollo sistemático
- Interfaz bien definida
- Bien documentada

Software



- El mundo no puede funcionar sin software.
- Las industrias están controladas por sistemas de software, como los sistemas financieros, los laboratorios científicos, las infraestructuras y los servicios públicos, los juegos, el cine, la televisión y la lista continúa.

So, what's a software anyway?

El futuro con el software:
https://www.youtube.com/watch?v=v4cKDzTyOek

Software

- Un **programa** es un conjunto de instrucciones (escritas en forma de código legible por humanos) que realiza una tarea específica.

- Un **software** es un programa informático junto con la documentación asociada y los datos de configuración que hacen que estos programas funcionen correctamente [Som11].

¿Qué es Ingeniería de Software?

- La Ingeniería de Software es el área de las ciencias de la computación que trata con la construcción de sistemas de software, los cuales son **tan grandes** y **complejos** que **se construyen con equipos de ingenieros**" [Ghezzi 91].

- Implica el uso de técnicas y prácticas ingenieriles para alcanzar un resultado previsible, en términos de proyecto y de producto...

¿Qué es Ingeniería de Software?

- Es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software. [Glosario de IEEE]

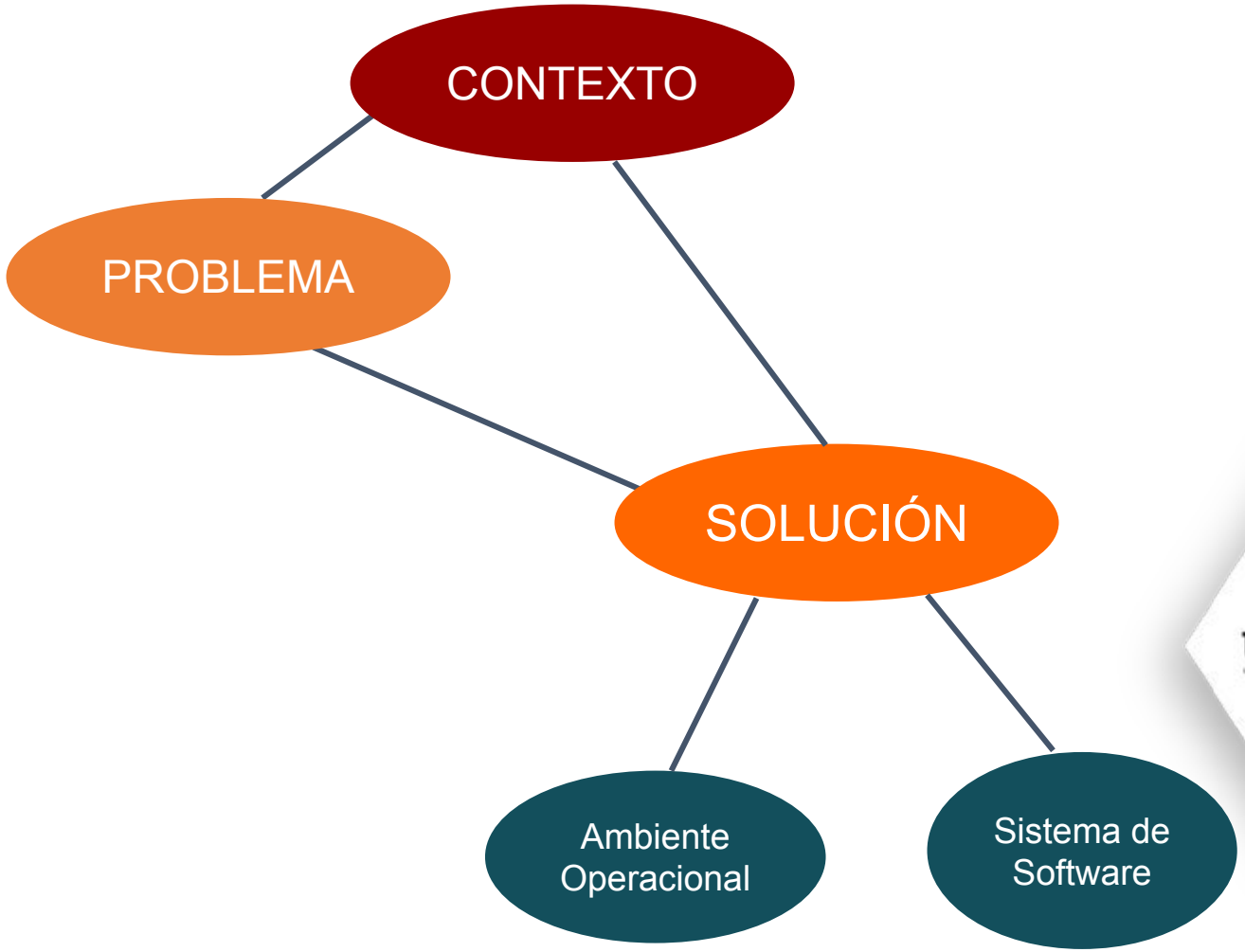
Proceso de Software

"Es un **proceso definido** , que guía la especificación, diseño, implementación, pruebas e implantación de una solución de software, de **modo eficiente y eficaz**"

- Esto requiere que al empezar el proceso se tenga:
  - **Objetivos** claros y explícitos
  - **Planes** para lograr los objetivos
  - **Procedimientos** que implementan los planes
  - **Procedimientos** de monitoreo y control de los planes
  - **Un ambiente** conducente al logro de los objetivos

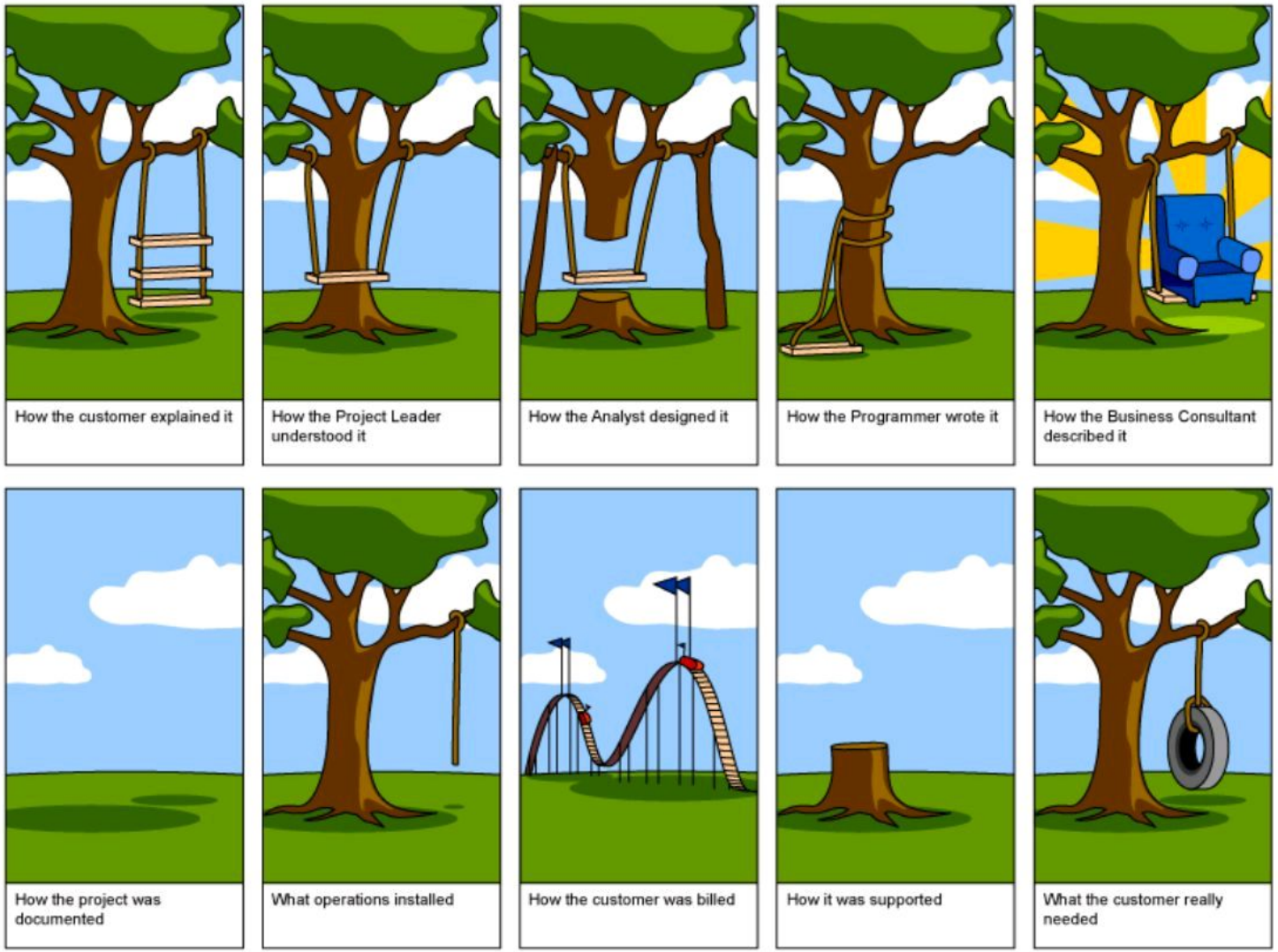
Proyecto de Software

- **Objetivo:** Solucionar un problema de manera eficaz,... y ojala eficientemente (en términos de costo y tiempo) y con bajo riesgo.
- **TODOS** los proyectos de software involucra:



Nuestro principal enemigo

Debemos “ **identificar y enfocarnos en lo importante**”, y debemos “ **ser efectivos** ” en eso.



Proyecto de Software

- Especifiquemos una solución **factible** .. diseñemos algo **implementable** y de bajo riesgo:
  - Con **tecnología** accesible/disponible
  - Con **costos** razonables
  - Con **plazos** adecuados
  - Con una buena relación "**Costo/beneficio**"
- El diseño casi **siempre** requiere de una **contraparte válida** (cliente y usuario) que opine sobre:
  - **Modelo de negocio** que se propone con la solución de software
  - **Factibilidad(técnica/social/económica/operativa)** de ponerlo en producción.

Componente de un producto Software

Métodos, técnicas, Estándares Procedimientos herramientas

Planificación del Proyecto

- Registros de estados
- Planes de pruebas
- Testeos

Planes de Control de calidad

- Revisiones
- Inspecciones
- Resultados de los planes

Objetos Reutilizable

- Procedimientos Almacenados
- Librerías, funciones, mòdulos. etc

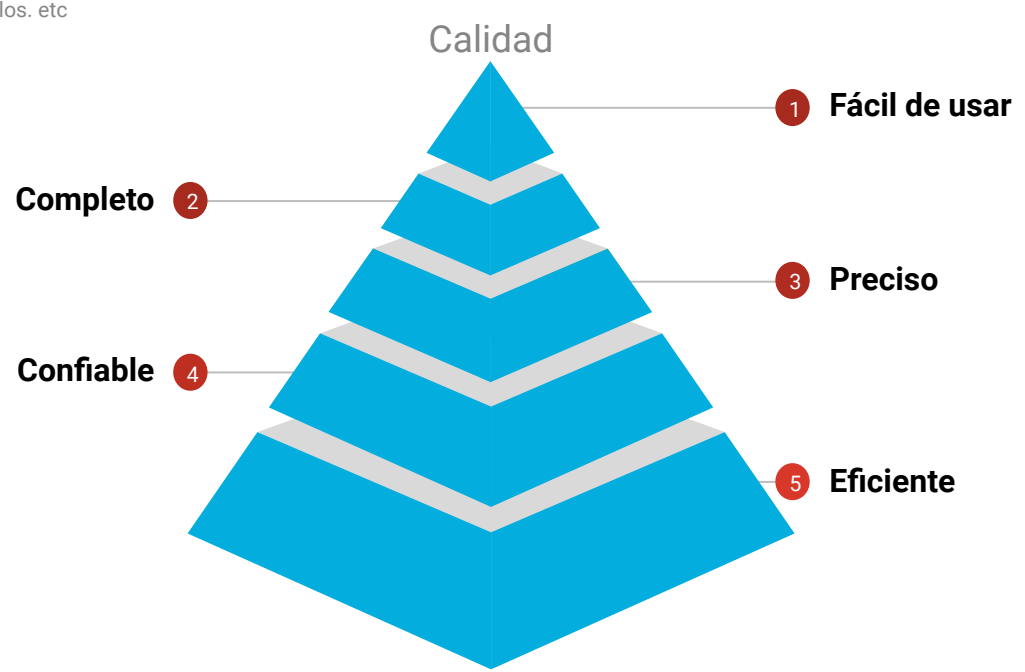
Códigos

- Fuentes
- Ejecutables

Manuales

- De usuario
- De procedimientos

Documentación



Proyecto de Software

- La Ingeniería de Software tiene que ver principalmente con el desarrollo y evolución de sistema grandes o complejos, donde:
  - Software debe ser terminado a tiempo y dentro del presupuesto
  - Software debe tener nivel de desempeño adecuado
  - Software debe ser correcto, confiable, mantenible, escalable, flexible y robusto.
- La parte difícil es lograr todo esto en proyectos grandes o complejos, usualmente donde hay:
  - Rotación de personal (en ambos bandos)
  - Requisitos cambiantes
  - Cambios de tecnología
  - Soluciones que se vuelve obsoleta durante el desarrollo, etc.

Proyecto de Software

- ¿Que es un proyecto Grande?

Kernel de linux tiene más de 10 millones de líneas de código

Alrededor de 50 millones de líneas de código.

Alrededor de 3.5 Millones de líneas de código .

- Comprender y administrar soluciones de esa magnitud requiere de ayuda extra, requiere **Ingeniería de Software**.
- Windows Vista: 1,200 desarrolladores aprox; 2,400 tester aprox.
- Coordinar un equipo de desarrollo de esa magnitud es un esfuerzo y en desafío enorme.

¿Podemos tener problemas?

- Desarrollar software **NO ES FÁCIL**
- En caso extremos (… pero no tan extremos)
  - Quiebra del productor del software
  - Quiebra de los clientes que dependen del producto
  - Pérdidas de vidas humanas

Algunas historia de horror

- El Bank of America proyectó **US$23 M** para un proyecto a 5 años. Se terminó gastando más de **US$60 M** para finalmente abandonar el proyecto.
- El bombardero B1 requirió **US$ 1000 M** más de lo proyectado para mejorar sus sistemas de defensa.
- AllState (seguros) comenzó en 1982 un proyecto de automatización integral de sus operaciones de 5 años de duración y US$8 M de presupuesto. Fue abandonado en 1993 después de gastar US$100 M.
- **Blue Cross** (EsSalud norteamericana) perdió más de US$60 M en pagos incorrectos debido a un error en el software por el que habían pagado más de US$200 M.
- El 4 de Julio de 1996 un cohete Ariadne se desvió de su curso y explotó a 3700 m de altura. Causa: error de software. Parte de las conclusiones de la comisión investigadora afirma .
- Entre 1987 y 1995 a lo menos 6 pacientes fueron severamente heridos o muertos por un error en los sistemas del Therac-25 (acelerador lineal usado para administración de radioterapia).

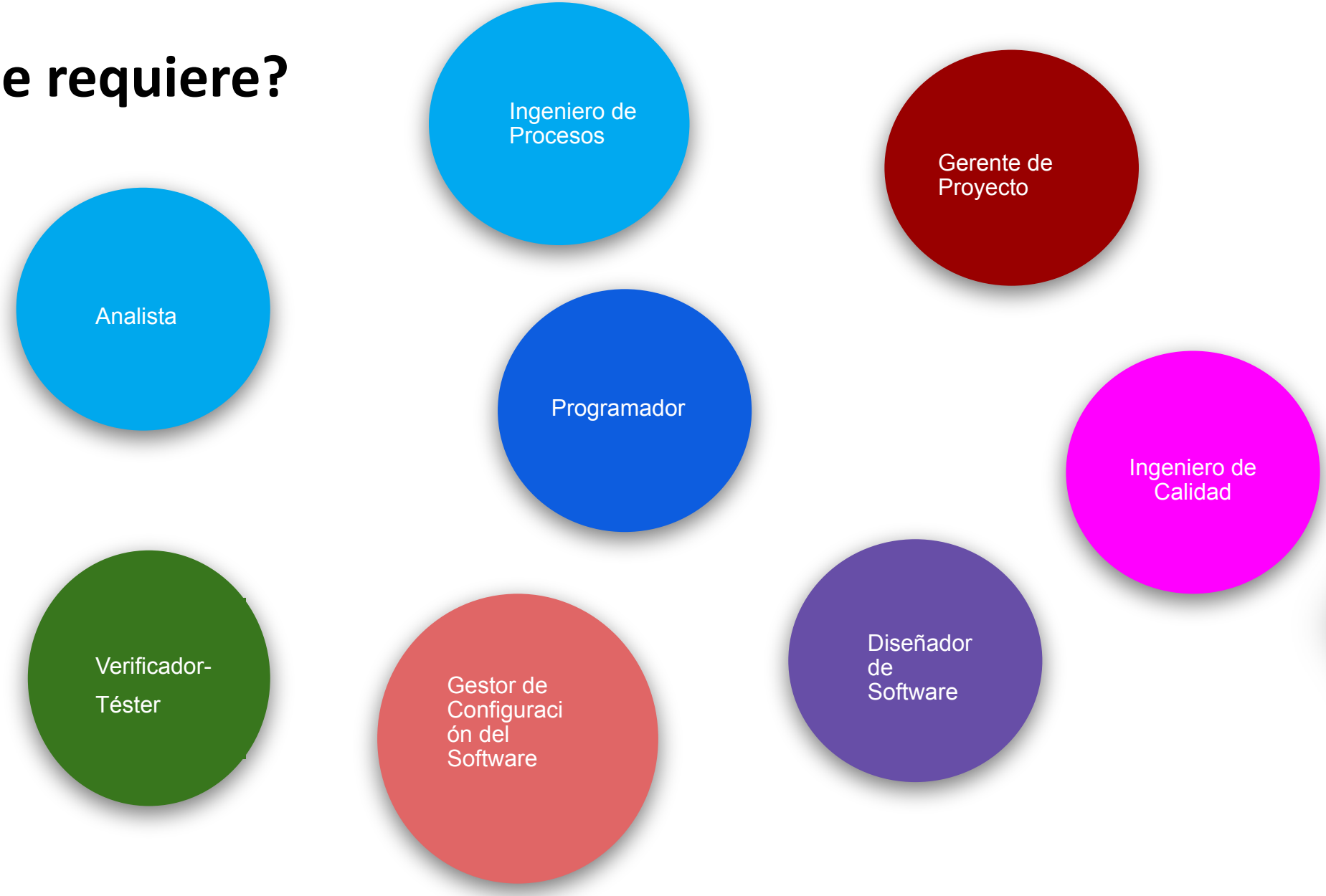
¿Qué se requiere?

- Manejo formal del proceso de desarrollo
- Documentación formal detallada tanto interna como externa
- Trazabilidad ¿de quien este código?¿cuando se agregó esta parte?
- Análisis cuidadoso del problema
- Diseño cuidado usando principios que han demostrado ser útiles.
- Implementación cuidadosa.
- Pruebas rigurosas
- Planificación de largo plazo.

¿Qué se requiere?

- En un **proyecto de software** se genera, además de código muchos documentos formales.
  - Documento de requisitos formales
  - Diseño de alto nivel
  - Diseño detallado
  - Plan y batería de pruebas (tests)
  - Documentación de usuario (Manuales)
  - Documentación de desarrolladores
  - Diccionario de datos (Base de datos)
  - Arquitectura a nivel infraestructura
  - Documentos de despliegue y pase a producción
  - y otros

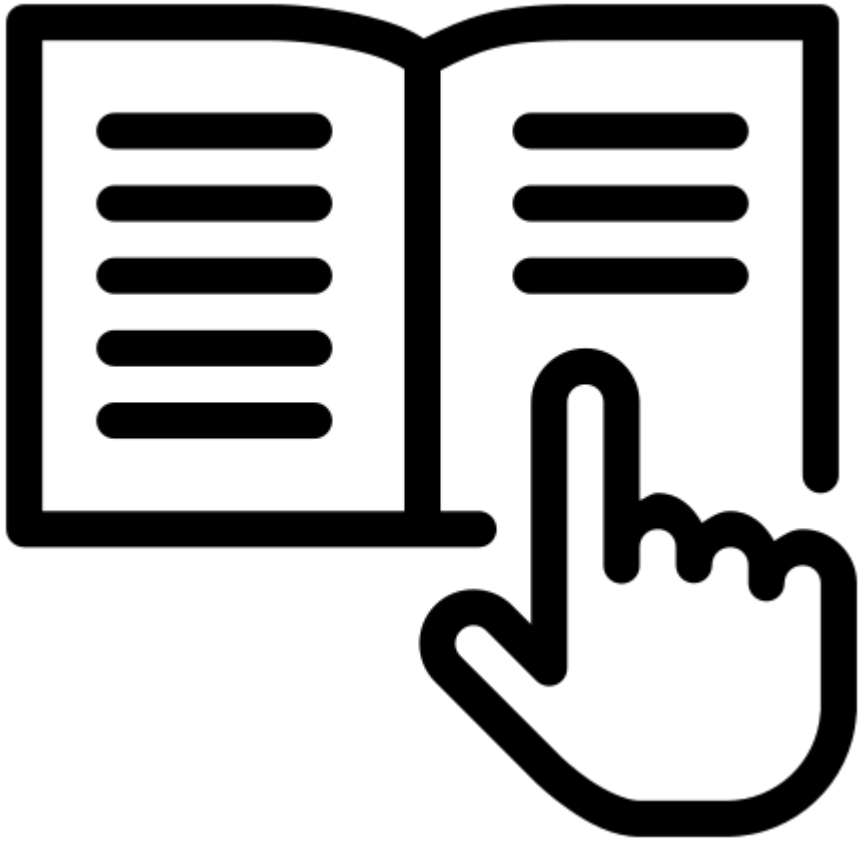
¿Qué se requiere?



Área de la Ingeniería de SW

- ✔ Evaluar - Medir- Controlar - Mantener - Mejorar - Entrenar - Educar

Lectura



- "Joys of the Programming Craft" fue tomado del Capítulo 1 del famoso libro The Mythical Man-Month, de Frederick P. Brooks.

Resumen

- ➔ Un **programa** es un conjunto de instrucciones, el software es un **programa** informático junto con la documentación asociada y los datos de configuración que hacen que estos programas funcionen correctamente.
- ➔ El software lo crea un EQUIPO, no solo usted (no la programación heroica).
- ➔ El software está bien documentado (no solo un boceto o maqueta).
- ➔ El software es de gran tamaño y número de usuarios, por lo que la calidad es muy importante (no cruzar los dedos).
- ➔ El desarrollo de software es sistemático siguiendo una metodología (no solo codificación).
- ➔ La Ingeniería de Software es la aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento de software.

Gracias !!

"Good software like wine takes time" - Joel Spolsky

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA