Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

SOFTWARE E INGENIERÍA DE SOTWARE

Sem - 1

Ing. Ena Mirella Cacho Chávez

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

Logro de la unidad

Al concluir la unidad, el estudiante describe los conceptos y principios de la ingeniería del software demostrando dominio del tema con claridad y precisión.

Logro de la sesión

Al finalizar la sesión, el estudiante estará en condiciones de:

- Identificar la naturaleza, proceso, dominios y principios de software
- Identificar las actividades que forman parte de la Práctica de ing. De software.

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

Tabla de Contenidos

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

1. EVALUACIÓN DE MATERIAL ASINCRÓNICO

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

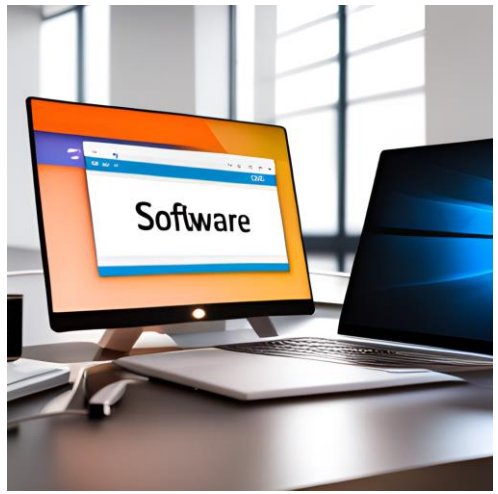
1.1. Actividad asincrónica

- Antes de la sesión sincrónica, el estudiante debe revisar: Lectura del Capítulo I : “Software e Ingeniería de Software” (Pressman & Maxim, 2020).

1.2. Evaluación de material asincrónico

- Estimado estudiante,

Luego de revisar el material brindado, responder a la interrogante:
¿Qué es lo que no se podía predecir hace sesenta años sobre el Software y que hoy es una realidad?



“Software a la carta”

1.2. Evaluación de material asincrónico

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- Hace sesenta años nadie podía predecir que el software se convertiría en:
- Una tecnología indispensable para los negocios, la ciencia y la ingeniería;
- Que el software permitiría la creación de nuevas tecnologías (por ejemplo, la ingeniería genética y la nanotecnología),
- La ampliación de tecnologías existentes (por ejemplo, las telecomunicaciones) y el cambio radical de tecnologías más antiguas (por ejemplo, los medios de comunicación)
- Que el software sería la fuerza motriz de la revolución de los ordenadores personales
- Que los consumidores comprarían aplicaciones de software utilizando sus dispositivos móviles
- Que el software evolucionará lentamente de un producto a un servicio a medida que las empresas de software "a la carta" ofrecieran funcionalidad justo a tiempo a través de un navegador web
- Que una empresa de software se hiciera más grande y más influyente que todas las empresas de la era industrial
- O que una vasta red impulsada por software evolucionara y cambiara todo, desde la investigación en bibliotecas a las compras de los consumidores, pasando por el discurso político y los hábitos de citas de los adultos jóvenes (y no tan jóvenes).

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

2. DESARROLLO DE CONTENIDOS

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

2.1. La naturaleza del software

Visualizar un corto video que lo puedes encontrar en “Material Complementario”

2.1. La naturaleza del software

El software tiene un doble papel.

- ***Es un producto:***
  - *El software **aprovecha** la capacidad informática de **hardware** (dispositivos móviles, ordenadores, nubes o máquinas autónomas) para **transformar, gestionar, adquirir y transmitir** información. Puede manejar **datos simples**, como un solo bit, o complejas representaciones de **realidad aumentada** generadas a partir de múltiples fuentes independientes superpuestas en el mundo real.*
- ***Es un vehículo para suministrar un producto:***
  - *El software actúa como base para el **control del ordenador** (sistemas operativos), la **comunicación** de información (redes) y la creación y **control de otros programas** (herramientas y entornos de software).*

2.1. La naturaleza del software

- ***Ventajas del software***
  - *Proporciona el **producto más importante** de nuestro tiempo: la información.*
  - *Transforma los **datos personales** (por ejemplo, las transacciones financieras de un individuo) para que puedan ser más útiles en un contexto local.*
  - *Gestiona la **información empresarial** para mejorar la competitividad*
  - *Proporciona una **puerta de acceso** a las redes mundiales de información (por ejemplo, la Inter- net) y*
  - *Proporciona **los medios** para adquirir información en todas sus formas.*

- ***Desventajas del software***
  - *Constituye un vehículo que puede amenazar la **intimidad personal** y*
  - *Una puerta que permite a los malintencionados cometer actos delictivos.*

2.1. La naturaleza del software

- *La **función** de los **programas informáticos** ha experimentado **cambios** significativos en los últimos 60 años.*

  - *Rendimiento del hardware*
  - *Arquitecturas de computación*
  - *Aumento de la memoria y la capacidad de almacenamiento*
  - *Gran variedad de opciones exóticas de entrada y salida han dado lugar a sistemas informáticos más sofisticados y **complejos.***
  - *La sofisticación y la **complejidad** pueden producir **resultados deslumbrantes** cuando un sistema tiene éxito, pero también pueden plantear enormes **problemas** a quienes **deben construir y proteger sistemas complejos.***

2.1. La naturaleza del software

- *Una enorme industria de software se ha convertido en un* factor dominante *en las* ***economías del mundo industrializado****.*
- *Los* ***equipos de especialistas*** *en software, cada uno centrado en una parte de la tecnología necesaria para ofrecer una aplicación compleja*
- *Han* ***sustituido al programador solitario*** *de una época anterior. Y, sin embargo, las preguntas que se hacían al programador solitario son las mismas que se plantean cuando se construyen sistemas informáticos modernos:*
  - *¿Por qué se* ***tarda*** *tanto en terminar un programa informático?*
  - *¿Por qué son tan* ***altos los costes*** *de desarrollo?*
  - *¿Por qué no podemos* ***encontrar todos los errores*** *antes de entregar el software a nuestros clientes?*
  - *¿Por qué dedicamos* ***tanto tiempo y esfuerzo*** *a* ***mantener los programas existentes****?*
  - *¿Por qué seguimos teniendo dificultades para* ***medir el progreso*** *a medida que se desarrolla y mantiene el software?*

Preguntas *son una manifestación de la preocupación por el software y su desarrollo, llevado a* ***la adopción de la práctica de la ingeniería del software***

2.2. Definición de Software

- *El software consiste en:*

(1) ***Instrucciones*** *(programas informáticos) que, cuando se ejecutan, proporcionan las características, funciones y rendimiento deseados*

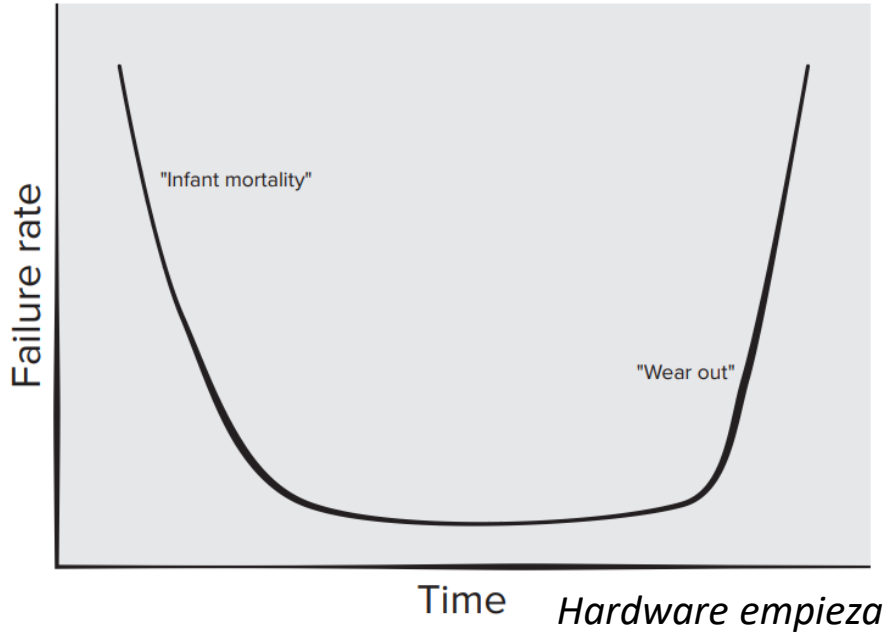
(2) ***Estructuras de datos*** *que permiten a los programas manipular información de forma adecuada; y*

(3) ***Información descriptiva*** *en formato impreso y virtual que describe el* ***funcionamiento y uso de los programas.***

- ***Características del software***
  - *Elemento lógico más que físico del sistema.*
  - *El software no se "desgasta".*

2.2. Definición de Software

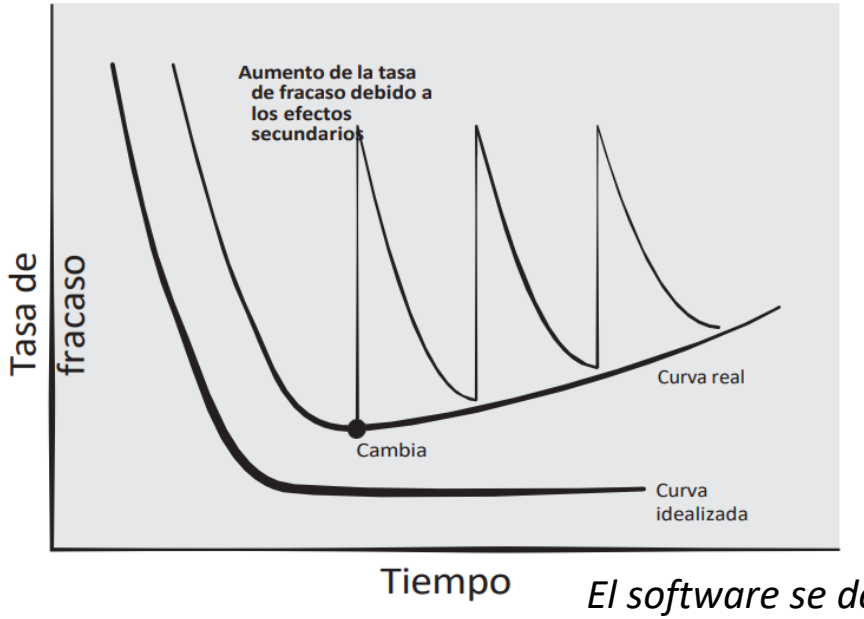
*Curva de fallos del **Hardware**, llamada también curva de la bañera.*



Hardware empieza a desgastarse

Si el hardware se desgasta, se sustituye por una pieza de recambio

*Curva de fallos del **Software***



El software se deteriora debido a los cambios.

*Cada fallo de software indica un error en el diseño o en el proceso a través del cual el diseño se tradujo en código ejecutable por la máquina, las tareas de **mantenimiento** del software implican una **complejidad** considerablemente mayor que el mantenimiento del hardware.*

2.3. Dominios de aplicación de software

- ***Siete*** *grandes categorías de programas informáticos plantean continuos retos a los ingenieros de software:*

1. Software de sistema.
2. Software de aplicación.
3. Software científico y de ingeniería.
4. Software integrado.
5. Software de línea de productos
6. Aplicaciones web/móvil.
7. Software de inteligencia artificial.

2.3. Dominios de aplicación de software

- **1. Software de sistema**
  - Conjunto de programas escritos para dar **servicio a otros programas** (por ejemplo, compiladores, editores y utilidades de administración de archivos) procesan estructuras de información complejas, pero **determinadas.**
  - Otras aplicaciones del sistema (por ejemplo, componentes del sistema operativo, controladores, software de red, procesadores de telecomunicaciones) procesan **datos en gran medida indeterminados**.
- Un programa informático es:
  - **Determinado**: si el orden y el momento de las entradas, el procesamiento y las salidas son **predecibles**.
  - **Indeterminado:** si el orden y el momento de las entradas, el procesamiento y las salidas **no pueden predecirse** con antelación.

2.3. Dominios de aplicación de software

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **2. Software de aplicación.**
  - Programas independientes que **resuelven una necesidad** empresarial **específica**. Las aplicaciones de esta área procesan datos empresariales o técnicos de forma que facilitan las operaciones empresariales o la toma de decisiones técnicas o de gestión.

- **3. Software científico y de ingeniería**.
  - Una amplia gama de programas de "cálculo numérico" o ciencia de datos que van desde la **astronomía** a la **vulcanología**, desde el análisis de **tensiones en automóviles** a la **dinámica orbital**, desde el **diseño asistido por ordenador** a los hábitos de **gasto de los consumidores**, y desde el **análisis genético** a la **meteorología**.

2.3. Dominios de aplicación de software

- **4. Software integrado.**
  - Reside dentro de un **producto** o **sistema** y se utiliza para **implementar y controlar características** y funciones para el usuario final y para el propio sistema.
  - El software integrado puede realizar:
    - **Funciones limitadas y esotéricas** (por ejemplo, el control del teclado de un horno de microondas) o
    - Proporcionar una **función** y una capacidad de **control significativas** (por ejemplo, funciones digitales en un automóvil, como el control del combustible, las pantallas del tablero y sistemas de frenado).

- **5. Software de línea de productos.**
  - Compuesto por **componentes reutilizables** y diseñado para proporcionar capacidades específicas para su **uso por muchos clientes** diferentes. Puede centrarse en un **mercado limitado** y esotérico (por ejemplo, productos de control de inventario) o intentar dirigirse al mercado de **consumo masivo.**

2.3. Dominios de aplicación de software

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **6. Aplicaciones web/móvil.**
- Esta categoría de software centrado en la red abarca una amplia gama de aplicaciones y engloba aplicaciones basadas en navegador, computación en la nube, computación basada en servicios y software que reside en dispositivos móviles.

- **7. Software de inteligencia artificial.**
- Hace uso de la **heurística** para resolver problemas complejos que no son susceptibles de cálculo regular o análisis directo. Las aplicaciones en este campo incluyen la **robótica**, los sistemas de toma de decisiones, el reconocimiento de patrones (imagen y voz), el aprendizaje automático, la demostración de teoremas y los juegos.

2.3. Dominios de aplicación de software

- Millones de **ingenieros de software** de todo el mundo **trabajan** duro en proyectos de software de una o varias de estas **categorías.**
  - En algunos casos:
    - Se construyen **nuevos** sistemas, pero en muchos otros
    - Se **corrigen, adaptan y mejoran** aplicaciones ya existentes.
- No es raro que una **joven ingeniera** de software trabaje en un **programa más antiguo que ella.**
- Las generaciones anteriores de personas dedicadas al software han **dejado un legado** en cada una de las categorías de las que hemos hablado.
- Esperemos que el **legado que dejará esta generación** alivie la carga de los **futuros ingenieros de software.**

2.4. Software Heredado

- Los **programas antiguos** a menudo denominados software heredado, han sido objeto de atención y preocupación continuas desde la **década de 1960.**
- Dayani-Fard y sus colegas [Day99] describen el software heredado de la siguiente manera:
  - Los sistemas de software heredados... se desarrollaron hace décadas y se han **modificado** continuamente para **adaptarse a los cambios** en los requisitos empresariales y las plataformas informáticas.
  - La proliferación de estos sistemas está causando quebraderos de cabeza a las grandes organizaciones, que los consideran **costosos de mantener y arriesgados de evolucionar.**

2.4. Software Heredado

- **Efecto secundario** presente en el software heredado, a consecuencia de los cambios:
- La mala calidad.
  - Diseños inextensibles
  - Código enrevesado
  - Documentación deficiente o inexistente,
  - Casos de prueba y resultados que nunca se archivaron, y
  - Un historial de cambios mal gestionado.

- Estos sistemas a menudo soportan "**funciones básicas** y son indispensables para el negocio". ¿Qué hacer? La única respuesta razonable puede ser: **No hacer nada**, al menos hasta que el sistema heredado deba sufrir algún **cambio significativo**.

2.4. Software Heredado

- **Si** el software heredado **satisface las necesidades** de sus usuarios y funciona con fiabilidad, no está roto y **no es necesario arreglarlo.**
- Sin embargo, con el paso del tiempo, los **sistemas heredados suelen evolucionar** por una o varias de las siguientes razones: ·
  - El software debe **adaptarse** a las necesidades de los **nuevos entornos informáticos** o tecnologías.
  - El software debe **mejorarse** para aplicar **nuevos requisitos empresariales**.
  - El software debe **ampliarse** para que **funcione con otros sistemas** o bases de datos más modernos.
  - Hay que **rediseñar** el software para hacerlo viable en un **entorno informático en evolución**.

2.4. Software Heredado

- El objetivo de la **ingeniería de software moderna**
  - “Idear metodologías que se basen en la **noción de evolución**; es decir, la noción de que los sistemas de software cambian continuamente, pueden construirse nuevos sistemas de software a partir de los antiguos , y ... todos deben interactuar y cooperar entre sí " [Day99].

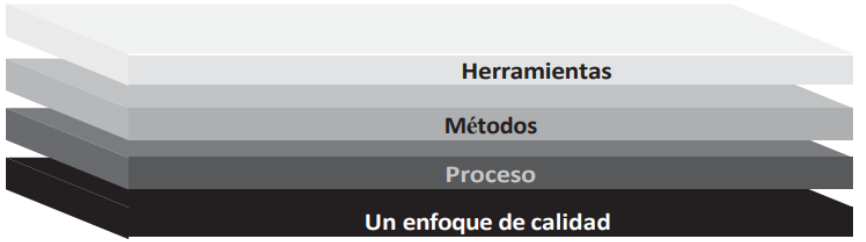
2.5. Definiendo la disciplina

- Según el IEEE
  - **Ingeniería del software:**
    - La aplicación de un **enfoque sistemático, disciplinado y cuantificable** al desarrollo, funcionamiento y mantenimiento del software; es decir, la aplicación de la ingeniería al software.
    - Sin embargo, un **enfoque "sistemático, disciplinado y cuantificable**" aplicado por un equipo de software puede **resultar pesado** para otro. Necesitamos **disciplina,** pero también **adaptabilidad y agilidad.**
    - En este caso, **la calidad se juzga** sobre la base del pensamiento de la **ingeniería de software moderna**, un criterio algo **injusto**, ya que algunos conceptos y principios de la ingeniería de software moderna pueden no haberse entendido bien en el momento en que se desarrolló el **software heredado.**

2.5. Definiendo la disciplina

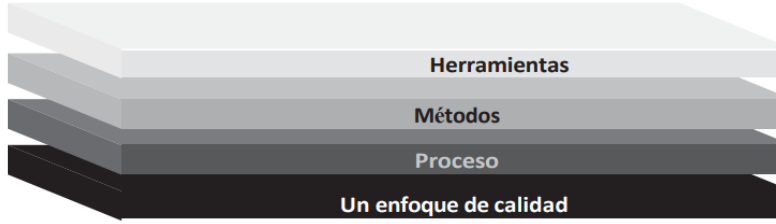
- La ingeniería de software es una tecnología **por capas**.
  - Cualquier enfoque de ingeniería (incluida la ingeniería de software) debe basarse en un **compromiso organizativo con la calidad.**
  - Es posible que haya oído hablar de la gestión de calidad total (TQM) o Six Sigma, y de filosofías similares que fomentan una cultura de mejora continua de los procesos. Es esta cultura la que, en última instancia, conduce a enfoques más eficaces de la ingeniería de software. **La base** sobre la que se sustenta la ingeniería de software es un **enfoque de calidad.**

Capas de ingeniería de software



2.5. Definiendo la disciplina

- La base de la ingeniería de software es la capa de proceso.
- El **proceso**
  - Define un **marco** que debe establecerse para la **entrega eficaz de tecnología** de ingeniería de software. El proceso de software constituye la **base del control de gestión de los proyectos** de software y establece el **contexto** en el que se aplican los métodos técnicos, se **producen los productos** de trabajo (modelos, documentos, datos, informes, formularios, etc.), se establecen los hitos, se garantiza la calidad y se gestionan adecuadamente los **cambios.**
- Los **métodos**
  - Proporcionan los **procedimientos técnicos** para crear software. Abarcan un amplio abanico de tareas que incluyen la **comunicación, el análisis de requisitos, el modelado del diseño, la construcción de programas, las pruebas y el soporte**. Los métodos de ingeniería de software se basan en un **conjunto de principios básicos** que rigen cada área de la tecnología e incluyen actividades de **modelado** y otras técnicas descriptivas.
- Las **herramientas**
  - Proporcionan un **soporte automatizado o semiautomatizado** para el proceso y los métodos. Cuando las herramientas se integran para que la información creada por una herramienta pueda ser utilizada por otra, se establece un sistema de apoyo al desarrollo de software, denominado **ingeniería de software asistida por ordenador.**



2.6. Proceso de Software

- **Un proceso**
  - Es un conjunto de actividades, acciones y tareas que se llevan a cabo cuando se va a crear algún producto de trabajo.
- **Una actividad**
  - Trata de alcanzar un objetivo general (por ejemplo, la comunicación con las partes interesadas) y se aplica independientemente del ámbito de aplicación, el tamaño del proyecto, la complejidad del esfuerzo o el grado de rigor con el que se realiza.
- **La ingeniería de software.**
  - Una acción (por ejemplo, el diseño arquitectónico) engloba un conjunto de tareas que generan un producto de trabajo importante (por ejemplo, un modelo arquitectónico).
- **Una tarea**
  - Se centra en un **objetivo pequeño,** pero bien definido (por ejemplo, realizar una prueba unitaria) que produce un resultado tangible.

2.6. Proceso de Software

- En el contexto de la ingeniería de software,
  - Un **proceso** no es una receta rígida sobre cómo crear programas informáticos. Se trata más bien de un **enfoque adaptable** que permite a las personas que realizan el trabajo (el equipo de software) elegir el conjunto adecuado de acciones y tareas. La **intención** es siempre entregar el **software a tiempo** y con la **calidad** suficiente para satisfacer a quienes han patrocinado su creación y a quienes lo utilizarán.

2.6. Proceso de Software

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **I. El marco del proceso**
  - Un marco de procesos **establece los cimientos** de un proceso completo de ingeniería del software **identificando** un pequeño número de **actividades marco** que son aplicables a todos los proyectos de software, independientemente de su tamaño o complejidad.
  - Además, el marco de procesos abarca un conjunto de **actividades paraguas** que son **aplicables a todo el proceso de software**.
  - Un marco de procesos genérico para la ingeniería de software engloba cinco **actividades:**
    - Comunicación
    - Planificación
    - Modelado
    - Construcción
    - Despliegue

2.6. Proceso de Software

- **1. Comunicación.**
  - **Comprender** los objetivos del proyecto de las partes interesadas y **reunir los requisitos** que ayuden a definir las características y funciones del software.
- **2. Planificación.**
  - Cualquier viaje complicado puede simplificarse si existe un mapa. El mapa llamado **plan de proyecto** de software **define el trabajo de ingeniería** de software describiendo las tareas técnicas que deben realizarse, los riesgos probables, los recursos necesarios, los productos que deben obtenerse y un calendario de trabajo.

2.6. Proceso de Software

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **3. Modelado.**
  - Tanto si eres paisajista, constructor de puentes, ingeniero aeronáutico, carpintero o arquitecto, trabajas con **modelos** todos los días. Creas un "**boceto**" de la cosa para entender el panorama general: qué aspecto tendrá desde el punto de vista arquitectónico, cómo encajan las partes constituyentes y muchas otras características. Si es necesario, se perfecciona el boceto con más y más detalles para comprender mejor el problema y cómo resolverlo.
  - Un ingeniero de software hace lo mismo creando **modelos** para **comprender mejor los requisitos** del software y el **diseño** que cumplirá esos requisitos.
- **4. Construcción.**
  - Lo que se diseña debe construirse. Esta actividad combina la generación de **código** (manual o automatizada) y las **pruebas** necesarias para detectar errores en el código.
- **5. Despliegue.**
  - El software (como una entidad completa o como un incremento parcialmente completado) se **entrega** al cliente, que **evalúa el producto** entregado y proporciona **comentarios basados en la evaluación**.

2.6. Proceso de Software

- **II. Actividades paraguas**
  - Se aplican a lo **largo de un proyecto** de software y ayudan a un equipo de software **a gestionar** y controlar:
    - El progreso
    - La calidad
    - El cambio y
    - El riesgo.
  - Incluyen:
    - Seguimiento y control del proyecto de software.
    - Gestión de riesgos.
    - Garantía de calidad del software.
    - Revisiones técnicas.
    - Medición.
    - Gestión de la configuración del software.
    - Gestión de la reutilización.
    - Preparación y producción de productos de trabajo.

2.6. Proceso de Software

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **III. Adaptación del proceso**
  - Señalamos que el proceso de ingeniería de software debe ser **ágil y adaptable** (al problema, al proyecto, al equipo, a la organización y cultura).
  - Por lo tanto, un **proceso adoptado** para **un proyecto** puede ser significativamente **diferente** de un proceso adoptado **para otro proyecto**. Entre **las diferencias** se encuentran:
    - Flujo global de actividades, acciones y tareas y las interdependencias entre ellas.
    - Grado de definición de las acciones y tareas dentro de cada actividad marco
    - Grado de identificación y exigencia de los productos de trabajo
    - Forma en que se aplican las actividades de garantía de calidad
    - Manera en que se aplican las actividades de seguimiento y control del proyecto
    - Grado general de detalle y rigor con el que se describe el proceso.
    - Grado de implicación del cliente y otras partes interesadas en el proyecto
    - Grado de autonomía del equipo informático
    - Grado en que se prescriben la organización y las funciones del equipo.

2.7. Práctica de Ingeniería de Software

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- Las **actividades genéricas** del marco de trabajo y las **actividades paraguas**, establecen un **esqueleto** de **arquitectura para el trabajo de ingeniería del software.**

- La esencia de la práctica de la ingeniería de software:
    1. Comprender el problema (comunicación y análisis).
    2. Planificar una solución (modelado y diseño de software).
    3. Llevar a cabo el plan (generación de código).
    4. Examinar la exactitud del resultado (pruebas y garantía de calidad).

2.7. Práctica de Ingeniería de Software

- **1. Entender el problema.**
  - A veces es difícil admitirlo, pero la mayoría de nosotros sufrimos de arrogancia cuando se nos presenta un problema. Escuchamos durante unos segundos y luego pensamos: "Ah, sí, lo entiendo, vamos a resolver esto". Por desgracia, comprender no siempre es tan fácil.

  - Merece la pena dedicar un poco de tiempo a responder algunas preguntas sencillas:
    - ¿Quién tiene interés en la solución del problema? Es decir, ¿quiénes son los **interesados**?
    - ¿Cuáles son las incógnitas? ¿Qué datos**, funciones** y características se necesitan para resolver correctamente el problema?
    - ¿Se puede compartimentar el problema? ¿Es posible representar problemas más pequeños que puedan ser más fáciles de entender?
    - ¿Puede representarse gráficamente el problema? ¿Puede crearse un modelo de análisis?

2.7. Práctica de Ingeniería de Software

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **2. Planificar la solución.**
  - Ya conoces el problema (o eso crees) y estás impaciente por empezar a programar. Antes de hacerlo, frena un poco y haz un pequeño **diseño**:

    - ¿Ha visto problemas similares antes? ¿Existen patrones reconocibles en una posible solución? ¿Existen programas informáticos que implementen los datos, funciones y características necesarios?
    - ¿Se ha resuelto un problema similar? En caso afirmativo, ¿son reutilizables los elementos de la solución?
    - ¿Pueden definirse subproblemas? En caso afirmativo, ¿son fáciles de encontrar soluciones para los subproblemas?
    - ¿Puede representar una solución de manera que conduzca a una aplicación eficaz? ¿Se puede crear un modelo de diseño?

2.7. Práctica de Ingeniería de Software

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **3. Llevar a cabo el plan.**
  - El diseño que has creado sirve de **hoja de ruta** para el sistema que quieres construir. Puede **haber desvíos** inesperados y es posible que descubras una ruta aún mejor sobre la marcha, pero el "plan" te permitirá avanzar sin perderte.
    - ¿Se ajusta la solución al plan? ¿Es el código fuente trazable al modelo de diseño?
    - ¿Es cada componente de la solución demostrablemente correcto? ¿Se han revisado el diseño y el código o, mejor aún, se han aplicado pruebas de corrección al algoritmo?

2.7. Práctica de Ingeniería de Software

- **4. Examine el resultado.**
  - No puedes estar seguro de que tu solución sea perfecta, pero sí de que has diseñado un número suficiente de pruebas para descubrir tantos errores como sea posible.
    - ¿Es posible probar cada componente de la solución? ¿Se ha aplicado una estrategia de pruebas razonable?
    - ¿Produce la solución resultados conformes a los datos, funciones y características requeridos? ¿Se ha validado el software con respecto a todos los requisitos de las partes interesadas?
  - No debería sorprenderle que gran parte de este enfoque sea de sentido común. De hecho, es razonable afirmar que un enfoque de sentido común de la ingeniería de software nunca te llevará por mal camino.

2.8. Principios generales

- **Principio**
  - “Una importante ley subyacente o suposición requerida en un sistema de pensamiento".

  - Algunos se centran en **la ingeniería de software** en su conjunto
  - Otros consideran **una actividad** específica del marco genérico (por ejemplo, la comunicación), y
  - Otros se centran en **acciones** de ingeniería de software (por ejemplo, el diseño arquitectónico) o tareas técnicas (por ejemplo, la creación de un escenario de uso).
- **Tenemos 7 principios:**
  - El primer principio: la razón de ser de todo

2.8. Principios generales

- **El primer principio: la razón de ser de todo**
  - Un sistema informático existe por una razón: **aportar valor** a sus usuarios. Todas las decisiones deben tomarse teniendo esto en cuenta.
  - Antes de especificar un requisito del sistema, antes de anotar una funcionalidad del sistema, antes de determinar las plataformas de hardware o los procesos de desarrollo, hágase preguntas como:
  - "¿**Añade esto valor** real al sistema?". Si la respuesta es negativa, no lo haga. Todos los demás principios respaldan éste.

2.8. Principios generales

- **El segundo principio: KISS (Keep It Simple, Stupid!)**
  - Hay muchos factores a tener en cuenta en cualquier esfuerzo de diseño. Todo **diseño debe ser lo más sencillo** posible, pero no más simple. Esto facilita tener un sistema más fácil de entender y de mantener.
  - Esto no quiere decir que haya que descartar funciones en nombre de la sencillez. De hecho, los diseños más elegantes suelen ser los más sencillos. **Sencillo no significa "rápido y sucio".** A menudo, simplificar el diseño requiere mucha reflexión y trabajo a lo largo de múltiples iteraciones. El resultado es un software más fácil de mantener y menos propenso a errores.

2.8. Principios generales

- **El Tercer Principio: Mantener la visión**
  - Una visión clara es esencial para el éxito de un proyecto de software. Sin integridad conceptual, un sistema amenaza con convertirse en un mosaico de diseños incompatibles, sujetos por tornillos equivocados... . Comprometer la visión arquitectónica de un sistema de software debilita y acaba rompiendo incluso los sistemas bien diseñados. Contar con un **arquitecto capacitado** que pueda **mantener la visión e imponer el cumplimiento** ayuda a garantizar el éxito de un proyecto de software.

2.8. Principios generales

- **El Cuarto Principio: Lo que tú produces, otros lo consumirán**
  - Especifique, diseñe, documente e implemente siempre sabiendo que **otra persona tendrá que entender lo que está haciendo.** La audiencia de cualquier producto de desarrollo de software es potencialmente grande. Especifica pensando en los usuarios. Diseña teniendo en cuenta a los implementadores. Codifique teniendo en cuenta a quienes deben mantener y ampliar el sistema. Puede que alguien tenga que depurar el código que escribes, y eso le convierte en usuario de tu código. **Facilitarle el trabajo añade valor al sistema.**

2.8. Principios generales

- **Quinto principio: apertura al futuro**
  - En los entornos informáticos actuales, donde las especificaciones cambian de un momento a otro y las plataformas de hardware se quedan obsoletas a los pocos meses, la vida útil del software suele medirse en meses y no en años. Sin embargo, los sistemas de software verdaderamente "industriales" deben durar mucho más. Para ello , los sistemas deben estar preparados para adaptarse a estos y otros cambios. Los sistemas que lo hacen con éxito se han diseñado así desde el principio. No se arrinconen nunca. Pregúntese siempre "¿y si...?" y prepárese para todas las respuestas posibles creando sistemas que **resuelvan el problema general, no sólo el específico.**

2.8. Principios generales

- **Sexto principio: Planificar la reutilización**
  - La reutilización **ahorra tiempo y esfuerzo**. Alcanzar un alto nivel de reutilización es sin duda el objetivo más difícil de lograr en el desarrollo de un sistema de software. La reutilización de código y diseños se ha proclamado como una de las principales ventajas del uso de tecnologías orientadas a objetos. Sin embargo, la rentabilidad de esta inversión no es automática. Planificar con antelación la reutilización reduce el coste y aumenta el valor tanto de los componentes reutilizables como de los sistemas en los que se incorporan.

2.8. Principios generales

- **Séptimo principio: ¡Piensa!**
  - Este último principio es probablemente el más olvidado. Pensar de forma clara y completa antes de actuar casi siempre produce mejores resultados. Cuando piensas en algo, es más probable que lo hagas bien. También adquieres conocimientos sobre cómo volver a hacerlo bien. **Si piensas en algo y aun así lo haces mal, se convierte en una experiencia valiosa**. Un efecto secundario de pensar es aprender a reconocer cuándo no se sabe algo, momento en el que se puede investigar la respuesta. Cuando en un sistema se ha pensado con claridad, el valor sale a la luz. La aplicación de los seis primeros principios requiere una intensa reflexión, pero la recompensa potencial es enorme. Si todos los ingenieros y equipos de software siguieran los siete principios de Hooker, se eliminarían muchas de las dificultades que experimentamos al construir sistemas informáticos complejos

2.9. Cómo iniciar un proyecto

- Todo proyecto de software se precipita por alguna **necesidad** empresarial:

  - Corregir un defecto en una aplicación existente
  - Adaptar un "sistema heredado" a un entorno empresarial cambiante
  - Ampliar las funciones y características de una aplicación existente o
  - Crear un nuevo producto, servicio o sistema.

2.9. Cómo iniciar un proyecto

- Al principio de un proyecto de software, la **necesidad** empresarial suele **expresarse de manera informal** en el marco de una conversación sencilla. La conversación presentada en la barra lateral es típica.
- Apenas se mencionó el software como parte de la conversación. Y, sin embargo, el software hará o deshará la línea de productos SafeHome.

Cómo se inicia un proyecto

El escenario: Sala de reuniones de CPI Corporation, una (ficticia) empresa que fabrica productos de consumo para uso doméstico y comercial.

Los protagonistas: Mal Golden, director de desarrollo de productos; Lisa Perez, directora de marketing; Lee Warren, director de ingeniería; Joe Camalleri, vicepresidente ejecutivo de desarrollo de negocio.

La conversación:

Joe: Vale, Lee, ¿qué es eso que he oído de que tu gente está desarrollando un qué? ¿Una caja inalámbrica universal genérica?

Lee: E s muy chulo. . . del tamaño de una pequeña caja de cerillas. . . podemos conectarlo a sensores de todo tipo, una cámara digital, casi cualquier cosa. Utiliza el protocolo inalámbrico 802.11n. Nos permite acceder a la salida del dispositivo sin cables. Creemos que dará lugar a toda una nueva generación de productos.

Joe: ¿Estás de acuerdo, Mal?

Sí. De hecho, con unas ventas tan bajas como las de este año, necesitamos algo nuevo. Lisa y yo hemos estado haciendo un pequeño estudio de mercado, y creemos que tenemos una línea de productos que podría ser grande.

Joe: ¿Cómo de grande... de grande?

Mal (evitando un compromiso directo): Háblale de nuestra idea, Lisa.

Lisa: E s toda una nueva generación de lo que llamamos "productos de gestión del hogar". Los llamamos *SafeHome*. Utilizan la nueva interfaz inalámbrica y proporcionan a los propietarios o pequeños empresarios un sistema controlado desde el PC: seguridad doméstica, vigilancia del hogar, control de electrodomésticos y dispositivos... ya sabes, bajar el aire acondicionado de casa mientras conduces, ese tipo de cosas.

Lee (interviniendo): Ingeniería ha hecho un estudio de viabilidad técnica de esta idea, Joe. Es factible a bajo coste de fabricación. La mayoría del hardware está d i s p o n i b l e . El software es un problema, pero no es nada que no podamos hacer.

Joe: Interesante. Ahora, le pregunté acerca de la línea de fondo.

Mal: PCs y tabletas han penetrado más del 70 por ciento de todos los hogares en los EE.UU.. Si pudiéramos ponerle el precio adecuado, podría ser una aplicación revolucionaria. Nadie más tiene nuestra caja inalámbrica ... es propietaria. Tendremos una ventaja de 2 años sobre el competencia. ¿Ingresos? Tal vez de 30 a 40 millones de dólares en el segundo año.

Joe (sonriendo): Vamos a llevar esto al siguiente nivel. Estoy interesado.

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

3. TRABAJO COLABORATIVO

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

Trabajo colaborativo

- *En grupos de 3 integrantes, elaborar un organizador gráfico sobre el tema.*
- *MIRO*

Receso

- *Para que todos los participantes pueden despejarse y regresen a la sesión con mayor predisposición para continuar, se brinda un pequeño receso de (10 minutos)*

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

4. EVALUACIÓN DE LA SESIÓN

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

Evaluación de la sesión

- *Estimado estudiante, vamos a ingresar a una herramienta de gamificación para responder a 10 preguntas que servirán para reforzar lo aprendido en la sesión.*

slido

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

5. CONCLUSIONES

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

5.1. Conclusiones

- *El **software** es el elemento clave en la **evolución de los sistemas y productos** informáticos y una de las tecnologías más importantes del panorama mundial. En los últimos 60 años, el software ha pasado de ser una herramienta especializada en la resolución de problemas y el análisis de la información a convertirse en una industria en sí misma.*
- *El **software heredado** sigue planteando **retos especiales** a **quienes deben mantenerlo.***
- ***La ingeniería de software engloba procesos**, métodos y herramientas que permiten construir sistemas informáticos complejos a tiempo y con calidad.*
- *El **proceso de software** incorpora cinco **actividades** marco: **comunicación, planificación, modelado, construcción y despliegue**; que son aplicables a todos los proyectos de software.*
- *La **práctica de la ingeniería del software** es una actividad de resolución de problemas que sigue un conjunto de principios básicos. A medida que aprenda más sobre la ingeniería de software, empezará a entender por qué estos principios deben tenerse en cuenta al iniciar cualquier proyecto de software.*

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

6. AVISO O ANUNCIO

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

6.1. Aviso o Anuncios

- *Para la siguiente sesión, debes revisar "Modelos de proceso"*

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

7. REFUERZO MI APRENDIZAJE

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

Cuestionario

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

QUIZIZZ

EXAMEN

UNC SOFTWARE E INGENIERÍA DE SOTWARE

10 Preguntas · Desarrollo profesional · Business

Ena Mirella Ca... · 2 minutos hace

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

7. REFERENCIAS BIBLIOGRÁFICAS

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

Referencias Bibliográficas

- *Pressman, R., & Maxim, B. (2020). Software engineering: A practitioner's approach (Ninth edition). McGraw-Hill Education.*

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

8. MATERIAL COMPLEMENTARIO

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

Material complementario

- *Software e Ingeniería de software (Pressman & Maxim, 2020)*

GRACIAS

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Ingeniería de Sistemas
Universidad Nacional de Cajamarca