SOFTWARE

ESCUELA DE INGENIERÍA INFORMÁTICA
PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

- El software de computadora es el producto que construyen los programadores profesionales y al que después le dan mantenimiento.
- Incluye programas que se ejecutan en una computadora de cualquier tamaño y arquitectura.
- La ingeniería de software está formada por un proceso, un conjunto de métodos (prácticas) y un arreglo de herramientas que permite a los profesionales elaborar software de cómputo de alta calidad.

2

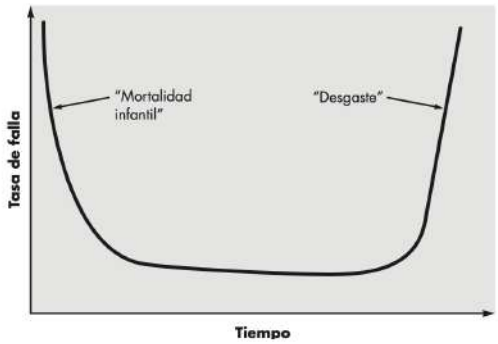
SOFTWARE

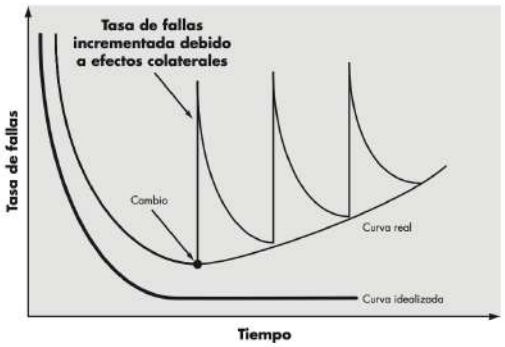
- Es importante examinar las características del software que lo hacen diferente de otros objetos que construyen los seres humanos.
- El software es elemento de un sistema lógico y no de uno físico.
- Tiene características que difieren considerablemente de las del hardware:
  - El software se desarrolla o modifica con intelecto; no se manufactura en el sentido clásico.
  - El software no se "desgasta".
  - Aunque la industria se mueve hacia la construcción basada en componentes, la mayor parte del software se construye para un uso individualizado.

3

SOFTWARE

- Curvas de falla de hardware versus software:





4

PRODUCTOS DE SOFTWARE

01 **Software a la medida**

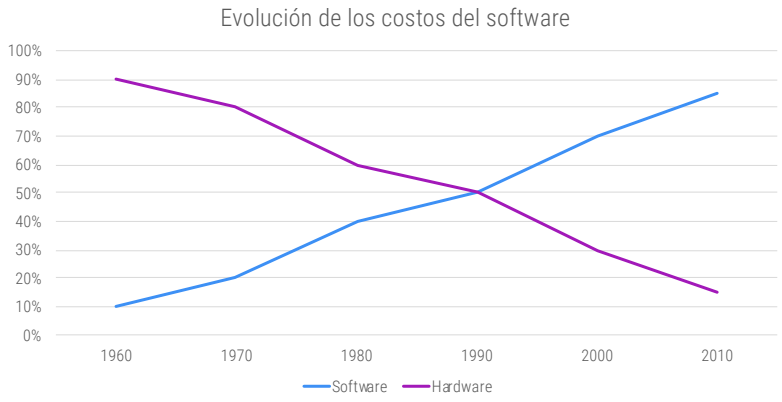
Aquel que se construye a la medida de las necesidades de la organización que contrata su diseño y puesta en marcha.

02 **Software empaquetado**

Aquel que se adquiere de forma genérica y debe existir un cierto grado de aprendizaje/adaptación por parte de la organización.

5

COSTOS DEL SOFTWARE



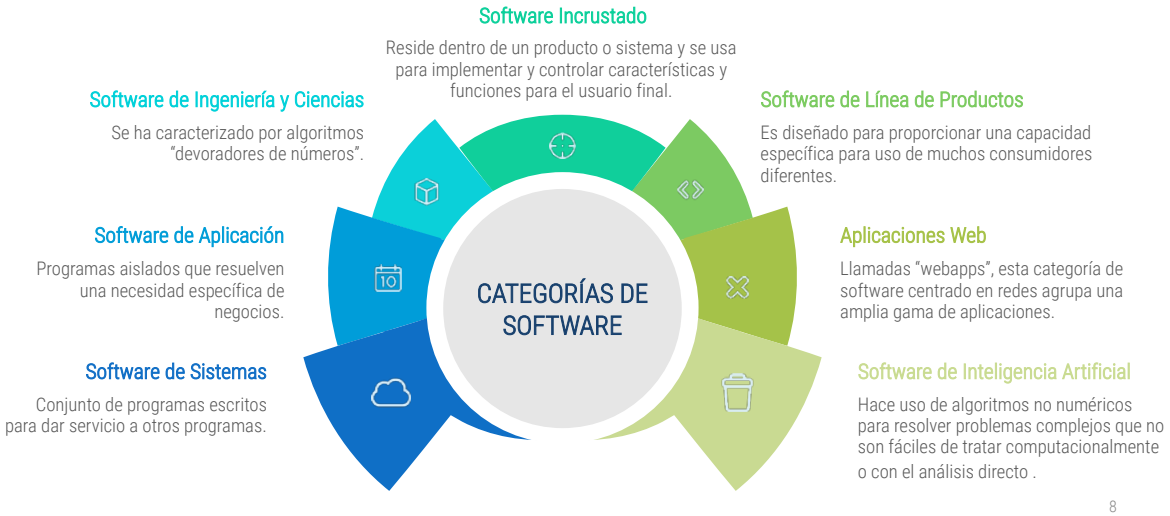
6

COSTOS DEL SOFTWARE

- El 55% de los sistemas cuestan más de lo esperado, el 68% superan la fecha de entrega y el 88% tuvieron que ser sustancialmente rediseñados – IBM.
- La media era 100 dólares por línea de código, se esperaba pagar 500 dólares por línea, y se terminó pagando entre 700 y 900 dólares por línea, 6.000 millones de dólares de trabajo fueron descartados - Advanced Automation System.
- El 74% de todos los proyectos de tecnologías de la información fallan porque se pasan de presupuesto, porque no cumplen el plazo de entrega, y el 28% de los proyectos fallan completamente - The Standish Group.

7

DOMINIOS DE APLICACIÓN



8

PRINCIPALES DESAFÍOS

ESCUELA DE INGENIERÍA INFORMÁTICA — PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Computación en un mundo abierto

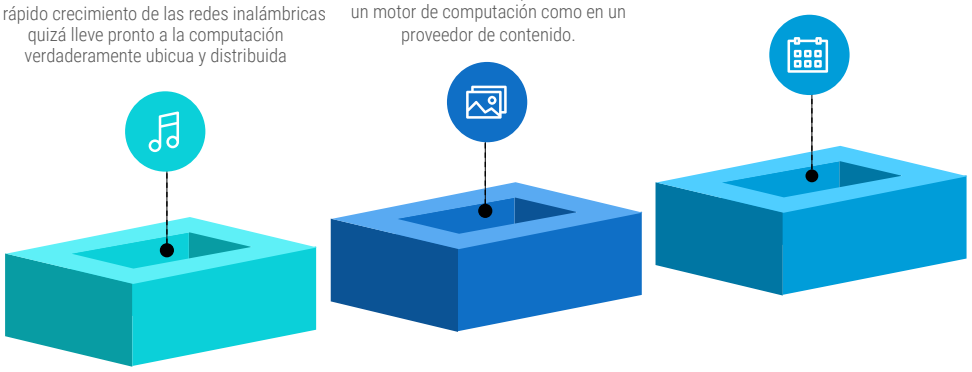
El rápido crecimiento de las redes inalámbricas quizá lleve pronto a la computación verdaderamente ubicua y distribuida

Construcción de redes

La red mundial (World Wide Web) se está convirtiendo con rapi- dez tanto en un motor de computación como en un proveedor de contenido.

Fuente abierta

Tendencia creciente que da como resultado la distribución de código fuente para aplicaciones de sistemas de modo que mucha gente pueda contribuir a su desarrollo.



9

SOFTWARE HEREDADO

ESCUELA DE INGENIERÍA INFORMÁTICA — PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

- Cientos de miles de programas de cómputo caen en uno de los siete dominios amplios de aplicación.
- Algunos de ellos son softwares muy nuevos, disponible para ciertos individuos, industria y gobierno.
- Pero otros programas son más viejos, en ciertos casos muy viejos.
- Estos programas antiguos —que es frecuente denominar software heredado— han sido centro de atención y preocupación continuas desde la década de 1960.

10

SOFTWARE HEREDADO

- Desafortunadamente, en ocasiones hay otra característica presente en el software heredado: mala calidad.
- Si el software heredado satisface las necesidades de sus usuarios y corre de manera confiable, entonces no falla ni necesita repararse.

→

11

SOFTWARE HEREDADO

- Sin embargo, conforme pase el tiempo será frecuente que los sistemas de software evolucionen por una o varias de las siguientes razones:
  - Debe adaptarse para que cumpla las necesidades de los nuevas tecnologías.
  - El software debe ser mejorado para implementar nuevos requerimientos del negocio.
  - El software debe ampliarse para que sea operable con otros sistemas o bases de datos.
  - La arquitectura del software debe rediseñarse para hacerla viable dentro de un ambiente de redes.

12

INGENIERÍA DE SOFTWARE

- El software se ha incrustado profundamente en casi todos los aspectos de nuestras vidas y, como consecuencia, el número de personas que tienen interés en las características y funciones que brinda una aplicación específica ha crecido en forma notable.
- Los requerimientos de la tecnología de la información que demandan los individuos, negocios y gobiernos se hacen más complejos con cada año que pasa.

→

13

INGENIERÍA DE SOFTWARE

- Los individuos, negocios y gobiernos dependen cada vez más del software para tomar decisiones estratégicas y tácticas, así como para sus operaciones y control cotidianos.
- A medida que aumenta el valor percibido de una aplicación específica se incrementa la probabilidad de que su base de usuarios y longevidad también crezcan.

14

INGENIERÍA DE SOFTWARE

- La ingeniería de software es una tecnología con varias capas.
- Cualquier enfoque de ingeniería (incluso la de software) debe basarse en un compromiso organizacional con la calidad.

15

PROCESO DE SOFTWARE

- Un proceso es un conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto del trabajo.
- En el contexto de la ingeniería de software, un proceso no es una prescripción rígida de cómo elaborar software de cómputo.
- Por el contrario, es un enfoque adaptable que permite que las personas que hacen el trabajo (el equipo de software) busquen y elijan el conjunto apropiado de acciones y tareas para el trabajo.

16

ESTRUCTURA DEL PROCESO

MODELADO

Ya sea usted diseñador de paisaje, constructor de puentes, ingeniero aeronáutico, carpintero o arquitecto, a diario trabaja con modelo.

PLANEACIÓN

Cualquier viaje complicado se simplifica si existe un mapa.

CONSTRUCCIÓN

Esta actividad combina la generación de código (ya sea manual o automatizada) y las pruebas que se requieren para descubrir errores en éste.

COMUNICACIÓN

Antes de que comience cualquier trabajo técnico, tiene importancia crítica comunicarse y colaborar con el cliente.

DESPLIEGUE

El software se entrega al consumidor que lo evalúa y que le da retroalimentación, misma que se basa en dicha evaluación

17

ACTIVIDADES CLAVE

- Es común que las actividades clave sean las siguientes:
  - Seguimiento y control del proyecto de software.
  - Administración del riesgo.
  - Aseguramiento de la calidad del software.
  - Revisiones técnicas.
  - Medición.
  - Administración de la configuración del software.
  - Administración de la reutilización.
  - Preparación y producción del producto del trabajo.

18

PRÁCTICA DE INGENIERÍA DE SOFTWARE

- En un libro clásico, How to Solve It, escrito antes de que existieran las computadoras modernas, George Polya describió la esencia de la solución de problemas y, en consecuencia, la esencia de la práctica de la ingeniería de software:
  - Entender el problema (comunicación y análisis).
  - Planear la solución (modelado y diseño del software).
  - Ejecutar el plan (generación del código).
  - Examinar la exactitud del resultado (probar y asegurar la calidad).

19

PRINCIPIOS DE LA PRÁCTICA DE INGENIERÍA DE SOFTWARE

1. La razón de que exista todo.
2. MSE (Mantenlo sencillo, estúpido...).
3. Mantener la visión.
4. Otros consumirán lo que usted produce.

5. Ábrase al futuro.
6. Planee por anticipado la reutilización.
7. Piense!

20

MITOS DEL SOFTWARE

- Mitos de la administración:
  - Tenemos un libro lleno de estándares y procedimientos para elaborar software. ¿No le dará a mi personal todo lo que necesita saber?
  - Si nos atrasamos, podemos agregar más programadores y ponernos al corriente (en ocasiones, a esto se le llama "concepto de la horda de mongoles").
  - Si decido subcontratar el proyecto de software a un tercero, puedo descansar y dejar que esa compañía lo elabore.

21

MITOS DEL SOFTWARE

- Mitos del cliente:
  - Para comrequerimientosenzar a escribir programas, es suficiente el enunciado general de los objetivos.
  - Los del software cambian continuamente, pero el cambio se asimila con facilidad debido a que el software es flexible.
- Mitos del profesional:
  - Una vez que hacemos que funcione el software, nuestro trabajo ha terminado.
  - El único producto del trabajo que se entrega en un proyecto exitoso es el programa.
  - La ingeniería de software hará que generemos documentación voluminosa e innecesaria, e invariablemente nos retrasará.

22

PRINCIPIOS DE LA INGENIERÍA DE SOFTWARE

- Principios de planeación:
  - Entender el alcance del proyecto.
  - Involucrar en la actividad de planeación a los participantes del software.
  - Reconocer que la planeación es iterativa.
  - Estimar con base en lo que se sabe.
  - Al definir el plan, tomar en cuenta los riesgos
  - Ser realista.

→

PRINCIPIOS DE LA INGENIERÍA DE SOFTWARE

- Principios de planeación:
  - Ajustar la granularidad cuando se defina el plan.
  - Definir cómo se trata de asegurar la calidad.
  - Describir cómo se busca manejar el cambio.
  - Dar seguimiento al plan con frecuencia y hacer los ajustes que se requieran.
- Principios de modelado:
  - El equipo de software tiene como objetivo principal elaborar software, no crear modelos.
  - Viajar ligero, no crear más modelos de los necesarios.

→

PRINCIPIOS DE LA INGENIERÍA DE SOFTWARE

ESCUELA DE INGENIERÍA INFORMÁTICA
PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

- Principios de modelado:
  - Tratar de producir el modelo más sencillo que describa al problema o al software.
  - Ser capaz de enunciar un propósito explícito para cada modelo que se cree.
  - Adaptar los modelos que se desarrollan al sistema en cuestión.
  - Tratar de construir modelos útiles, pero olvidarse de elaborar modelos perfectos.
  - No ser dogmático respecto de la sintaxis del modelo. Si se tiene éxito para comunicar contenido, la representación es secundaria.
  - Si su instinto dice que un modelo no es el correcto a pesar de que se vea bien en el papel, hay razones para estar preocupado.
  - Obtener retroalimentación tan pronto como sea posible.

25

PRINCIPIOS DE LA INGENIERÍA DE SOFTWARE

ESCUELA DE INGENIERÍA INFORMÁTICA
PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

- Principios de construcción:
  - Principios de codificación: preparación, programación, validación.
- Principios de prueba:
  - Todas las pruebas deben poder rastrearse hasta los requerimientos del cliente.
  - Las pruebas deben planearse mucho antes de que den comienzo.
  - El principio de Pareto se aplica a las pruebas de software.
  - Las pruebas deben comenzar “en lo pequeño” y avanzar hacia “lo grande”.
  - No son posibles las pruebas exhaustivas.

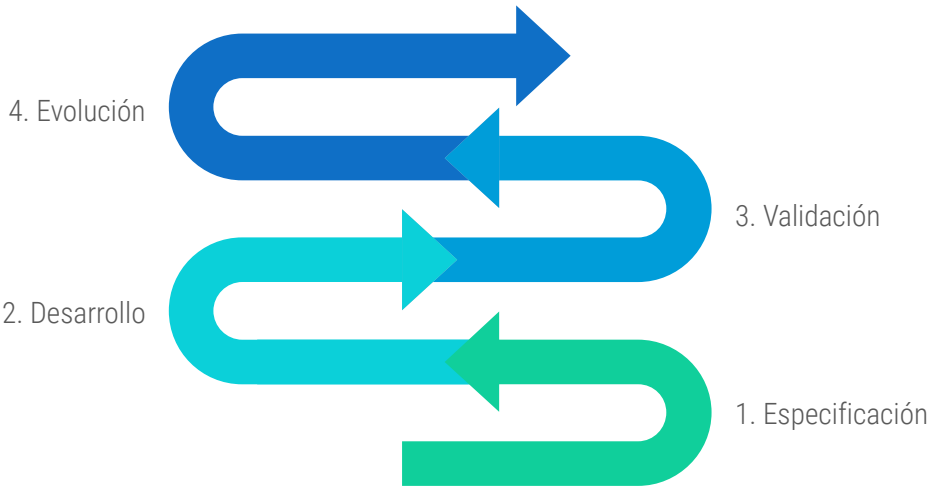
26

PRINCIPIOS DE LA INGENIERÍA DE SOFTWARE

- Principios de despliegue:
  - Deben manejarse las expectativas de los clientes.
  - Debe ensamblarse y probarse el paquete completo que se entregará.
  - Antes de entregar el software, debe establecerse un régimen de apoyo.
  - Se deben proporcionar a los usuarios finales materiales de aprendizaje apropiados.
  - El software defectuoso debe corregirse primero y después entregarse.

27

ACTIVIDADES DE LA INGENIERÍA DE SOFTWARE



28

PROCESOS DE SOFTWARE

- Conocidos como Paradigmas de Procesos de Software
- Presentan un proceso desde una perspectiva particular
- Son abstracciones:
  - Sirven de referencia para modelos más detallados
  - Pueden ser extendidos
  - Distintos modelos podrían combinarse en distintas fases del desarrollo
- Modelos más conocidos: Modelo en cascada, Modelo evolutivo, Desarrollo basado en componentes, Modelos iterativos (Desarrollo incremental y Desarrollo en espiral)

29

MODELO CASCADA

- El primer modelo propuesto, derivado de procesos de ingeniería más genéricos.
- Cada etapa entrega como resultado documentación aprobada.
- Es un modelo inflexible: errores detectados en una etapa implican reiniciar toda una iteración.
- Sólo es útil cuando los requerimientos se comprenden claramente y es difícil que cambien.

30

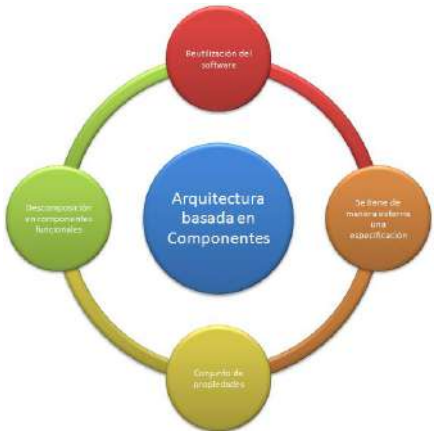
MODELO EVOLUTIVO

- Refinación de implementaciones iniciales tras una constante interacción con los usuarios y las etapas esenciales se entrelazan.
- Práctico cuando los requerimientos necesitan ser refinados:
  - Desarrollo exploratorio: refinación de los requerimientos que mejor se comprenden
  - Prototipos desechables: exploración gradual de los requerimientos menos comprendidos.
- Recomendado en proyectos de baja-mediana envergadura, en donde:
  - La visibilidad del proceso no es relevante
  - No se requiere una arquitectura sofisticada

31

DESARROLLO BASADO EN COMPONENTES

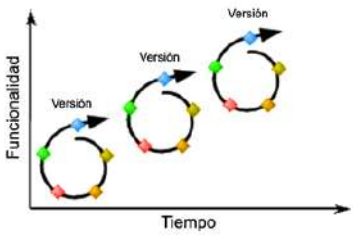
- Foco en integrar componentes de software reutilizables
- Permite agilizar las etapas asociadas a la implementación
- Riesgos inherentes:
  - Desviación de los requerimientos
  - Pérdida de control sobre la evolución de los componentes



32

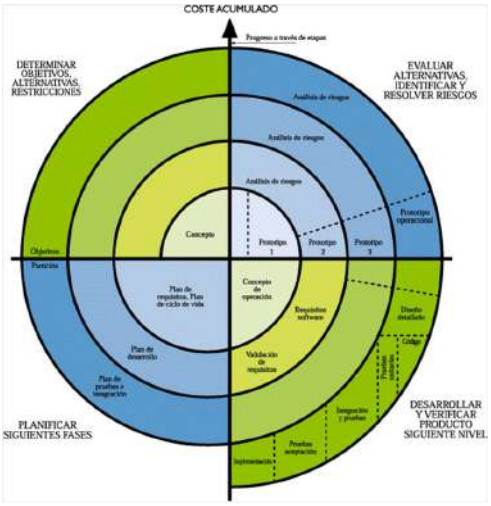
DESARROLLO INCREMENTAL

- Enfoque intermedio entre cascada y evolutivo
- Incrementos según prioridad de las funcionalidades
- Permite al usuario el contar tempranamente con las funcionalidades más importantes.
- El usuario comprende mejor los requerimientos a medida que avanzan los incrementos
- Dificultad para adaptar requerimientos a incrementos



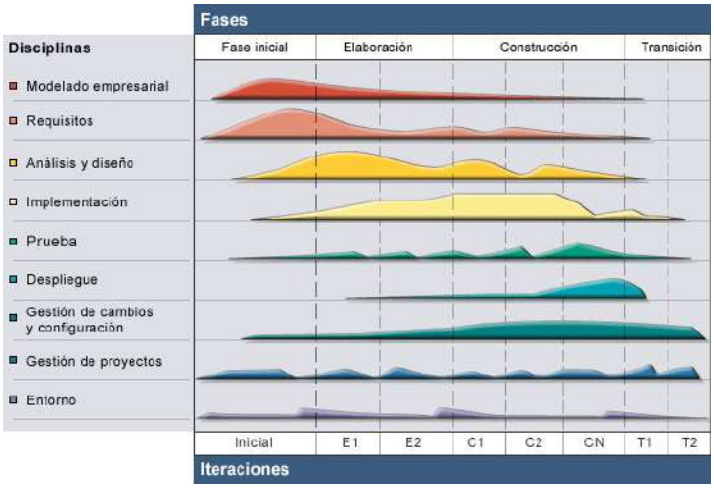
33

DESARROLLO EN ESPIRAL



34

PROCESO UNIFICADO DE RATIONAL (RUP)



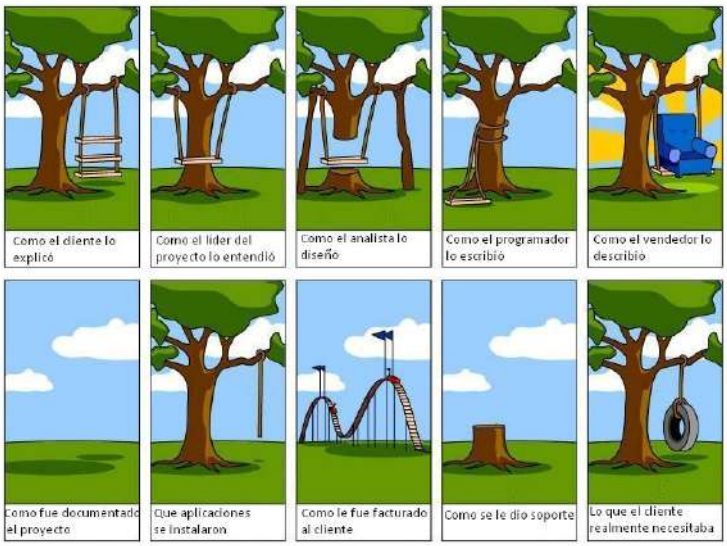
35

OBJETIVOS DE DISEÑO PREVIO AL DESARROLLO

- El principal objetivo de diseñar (modelar) los sistemas de información en minimizar los costos al momento de implementar, debido a que se disminuye el riesgo.
- Al disminuir los riesgos el proyecto se vuelve más seguro
- Al volverse un proyecto más seguro es aumenta la probabilidad de que el cliente quede satisfecho
- Para quien controla es importante conocer la necesidad de estos elementos en los proyectos de software.

36

OBJETIVOS DE DISEÑO PREVIO AL DESARROLLO



REQUERIMIENTOS DE SISTEMA

- Descripción de servicios proporcionados y sus restricciones operativas
- Responde a distintos niveles de generalidad según el contexto:
  - Abstracto: Cuando se necesita explicar el problema y buscar quién lo resolverá.
  - Detallado y formal: Cuando se necesita una mayor comprensión del problema y un mayor acercamiento a su solución.
- Requerimientos de usuario
- Requerimientos de sistema

CLASIFICACIÓN DE REQUERIMIENTOS

- Requerimientos funcionales: Servicios ofrecidos
  - Ingresar información detallada del cliente
  - Obtener un informe de ventas que ha hecho el cliente en el mes
- Requerimientos no funcionales: Restricciones asociadas
  - Fiabilidad
  - Tiempos de respuesta
  - Seguridad

39

CLASIFICACIÓN DE REQUERIMIENTOS

- Requerimientos no funcionales:

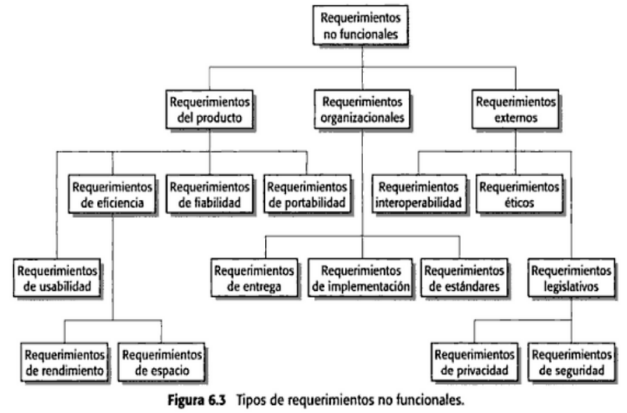


Figura 6.3 Tipos de requerimientos no funcionales.

40

REQUERIMIENTOS A NIVEL FUNCIONAL

- Requerimientos del usuario
  - No deben atender temas de diseño.
  - Definir un estándar y usarlo consistentemente
  - Usar referencias (identificadores)
  - Diferenciar entre lo obligatorio y lo deseado
  - Minimizar el lenguaje técnico

Identificador	Requerimiento
RF01	Módulo que brinde las funciones de catálogo en línea.
RF02	Función de búsqueda en el catálogo de productos.
RF03	Módulo para la gestión de usuarios.
RF04	Soporte para autenticación de usuarios.
RF05	No debe ser necesario ingresar al sistema para usar el módulo de

41

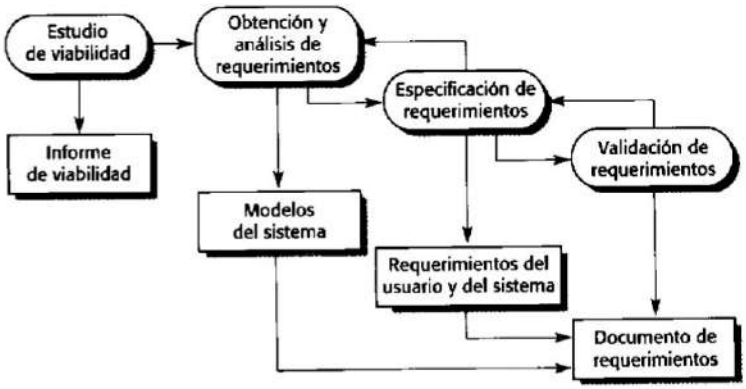
INGENIERÍA DE REQUERIMIENTOS

- Ingeniería de requerimientos
  - Proceso mediante el cual se logra un entendimiento acabado del sistema a desarrollar
  - Se asocia a la generación y mantención del documento de requerimientos de sistema
  - Adaptable a las circunstancias.

42

INGENIERÍA DE REQUERIMIENTOS

- Tareas asociadas a la ingeniería de requerimientos según Sommerville:



43

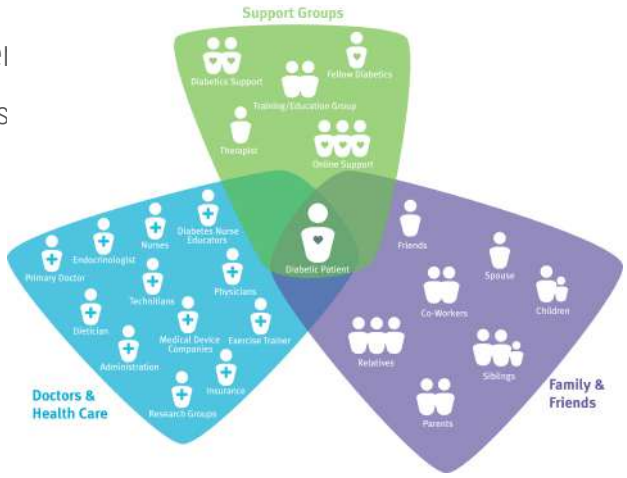
INGENIERÍA DE REQUERIMIENTOS

- Viabilidad: ¿es factible desarrollar el sistema en términos de:
  - Los objetivos de la organización
  - La tecnología y recursos disponibles
  - Los sistemas actuales
- Obtención y análisis de requerimientos:
  - Se descubren
  - Se ordenan por prioridades
  - Se negocian posibles conflictos
  - Se documentan

44

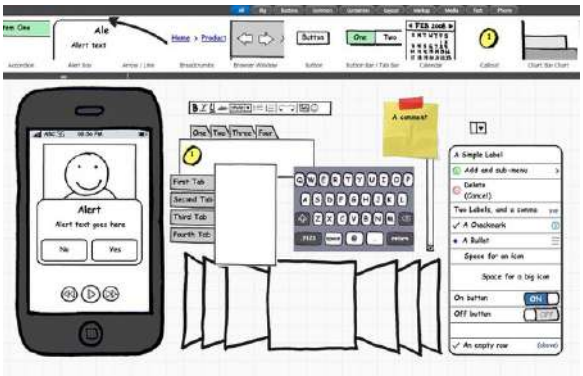
INGENIERÍA DE REQUERIMIENTOS

- Interacción con stakeholder
  - Entrevistas con los usuarios
  - Evaluación de escenarios



VALIDACIÓN DE REQUERIMIENTOS

- Validez, consistencia, completitud, realismo y verificalidad
- Algunas técnicas:
  - Revisiones
  - Prototipos
  - Casos de prueba



ESCUELA DE INGENIERÍA INFORMÁTICA

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

MÓDULO 1

PRINCIPIOS DE INGENIERÍA DE SOFTWARE

Prof. Mg. Rafael Mellado S.
rafael.mellado@pucv.cl
INF3242 – Modelamiento de Sistemas de Software
Escuela de Ingeniería Informática

47