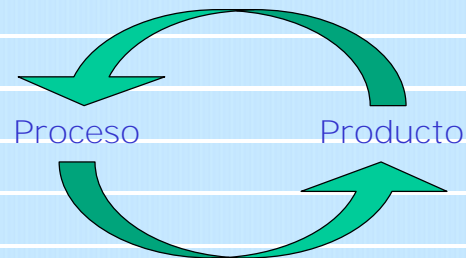


Principios de la Ingeniería de Software

Motivación

- ¿Cómo se logra construir un software de calidad?
- ¿Existe algún principio o práctica general que me asegure que el producto de software creado tendrá ciertas cualidades?



Algunos Conceptos

- Métodos
 - forma sistemática de abordar una actividad
- Técnicas
 - más técnicos y restringidos que los métodos
- Metodología
 - organización de métodos y técnicas para guiar el proceso de resolución de un problema
- Herramienta
 - encarna y fuerza la metodología (métodos y técnicas)

3

Relaciones entre conceptos



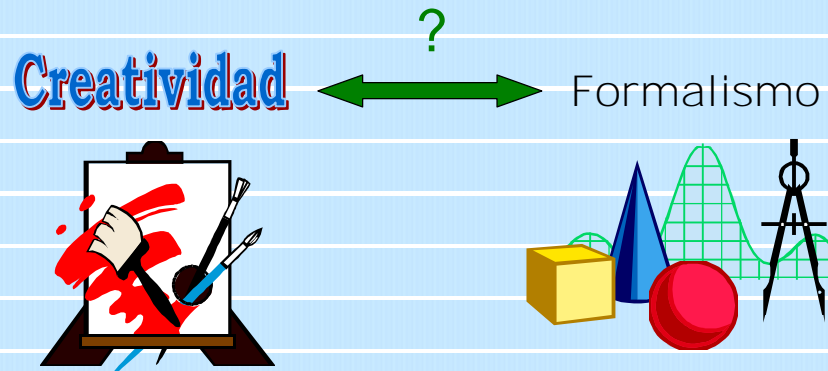
4

Principios

1. Rigor y formalidad
2. Separación de intereses
3. Modularidad
4. Abstracción
5. Anticipación del cambio
6. Generalidad
7. Incrementalidad

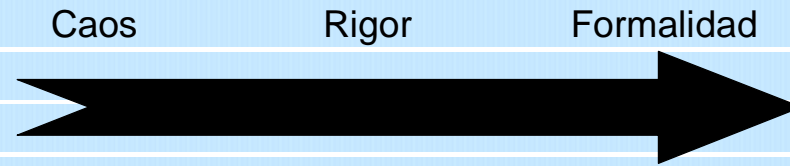
5

1. Rigor y Formalidad



6

Rigor y Formalidad



7

Rigor y Formalidad

- Una serie de pasos
 - definidos,
 - precisos,
 - coherentes.
- En cada paso se aplica alguna técnica.
- La técnica aplicada depende de:
 - resultados teóricos derivados de un modelo de la realidad,
 - ajustes empíricos para casos no cubiertos por el modelo,
 - experiencia del desarrollador.
- Esta mezcla define una metodología que puede usarse repetidamente.

Rigor o Formalismo

- El grado de formalismo usado en cada paso depende de varios factores:
 - experiencia en el área de desarrollo,
 - cuán crítico es el producto,
 - tiempo en ejecución.
- Es necesario que el desarrollador sepa decidir sobre el grado de formalidad necesario para cada paso del desarrollo.

9

Ejemplo

- El número de usuario es único en el sistema.

o

- $\forall u, u' \in \text{USUARIOS} \Rightarrow$
 $u.\text{número} = u'.\text{número} \Leftrightarrow u = u'$

Cualidades del Software

Documentación
rigurosa y/o formal

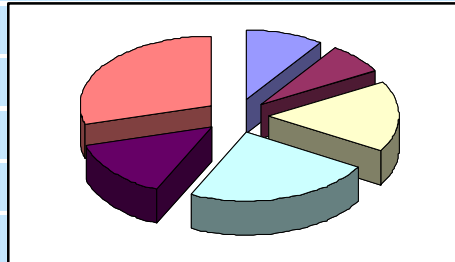
- Confiable
- Verificable
- Mantenible
- Reusable
- Portable
- Comprensible
- Interoperable

Cualidades del producto y del proceso de desarrollo.

11

2. Separación de Intereses

- “Divide et Impera”.
- Separar el problema en sus diferentes aspectos y abordarlos separadamente.



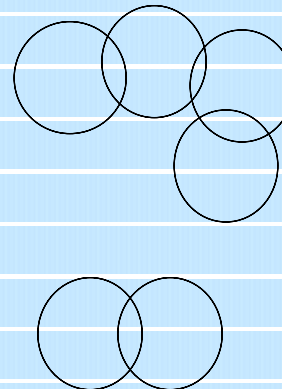
Distintos enfoques

- Para un producto de software:
 - funcionalidad ofrecida,
 - confiabilidad esperada,
 - eficiencia en el uso del tiempo y el espacio,
 - dependencias del ambiente,
 - interfaces del usuario.
- Para el proceso de desarrollo:
 - ambiente de desarrollo,
 - estructura y organización del equipo de trabajo,
 - agenda,
 - procedimientos de control,
 - estrategias de diseño,
 - mecanismos de recuperación de errores.
- Otros:
 - asuntos económicos y financieros.

13

Abordar los Distintos Enfoques

- Algunos aspectos no pueden decidirse en forma completamente independiente:
 - dinero disponible,
 - ambiente de desarrollo,
 - agenda.



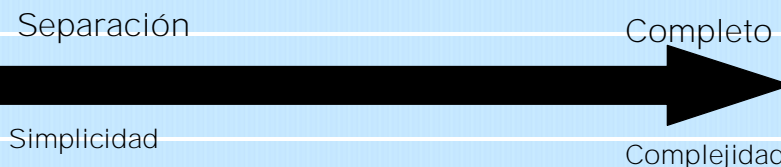
Separación de Intereses

- Tiempo:
 - organización de las actividades en el tiempo,
 - base del ciclo de vida del software,
 - secuencia de actividades.
- Cualidades:
 - dedicarse a desarrollar distintas cualidades por separado,
 - ej.: funcionalidad y eficiencia.
- Visiones:
 - flujo de datos y flujo de control,
 - interfaces del usuario y diseño de la base de datos.
- Partes:
 - refinamiento del sistema por partes dependiendo del tamaño,
 - modularidad.

15

Separación y Optimalidad

- La separación del problema completo en pequeñas partes sugiere que es posible no considerar una solución óptima global.
- En general, la complejidad del problema completo nos impide resolver bien los problemas parciales y también el problema global.



Consecuencias

- La separación de intereses determina la división de las tareas posiblemente entre personas con diferentes habilidades.
- También las necesidades de personal en las distintas etapas de desarrollo del software son diferentes:
 - cantidad,
 - calificación.

17

3. Modularidad

- Un sistema complejo puede dividirse en partes llamadas *módulos*.
- Un sistema dividido en módulos es modular.
- La modularidad es importante en casi todos los productos y procesos de ingeniería:
 - estandarización
 - reutilización.

18

Objetivos de la Modularidad

- Descomposición
 - descomponer un sistema en grandes bloques,
 - aplicar la descomposición recursivamente a bloques complejos.
- Composición
 - usando varios elementos (módulos) básicos, se construye un sistema más complejo,
 - la reutilización de módulos de software beneficia todo el proceso de desarrollo.
- Comprensión
 - es más fácil localizar cambios en un sistema modular,
 - también es más fácil comprender módulos pequeños.

19

Cohesión y Acoplamiento

- Alta Cohesión (interna)
 - los elementos comprendidos dentro de un módulo están altamente relacionados,
 - cooperan para lograr un mismo fin,
 - no están juntos al azar.
- Bajo Acoplamiento (externo)
 - la relación e interacción entre los distintos módulos debe ser simple,
 - si dos módulos dependen altamente uno del otro, es probable que los cambios en un módulo afecten al otro.

Dos Fases de Modularidad

- Fase 1: Nivel de Módulos
 - los detalles de cada módulo se tratan en forma independiente.
- Fase 2: Nivel del Sistema
 - la interacción entre los módulos se trata independientemente de los detalles internos.

Fase 1 - Fase 2:

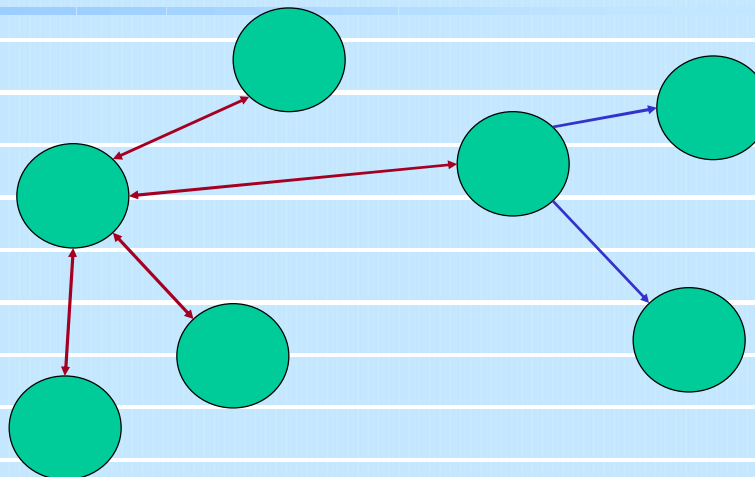
Bottom - up

Fase 2 - Fase 1:

Top - down

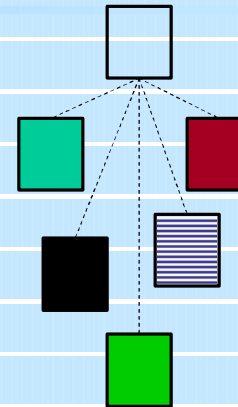
21

Buena Modularización



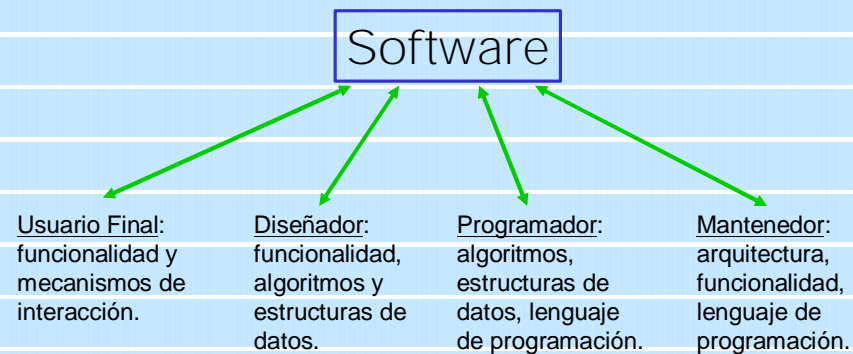
4. Abstracción

- Distinguir los elementos esenciales e ignorar los detalles.
- Es una forma de separación de intereses (esencia y detalles).
- La abstracción no es única:
 - depende del software,
 - del usuario de la abstracción.



23

Abstracciones del Software



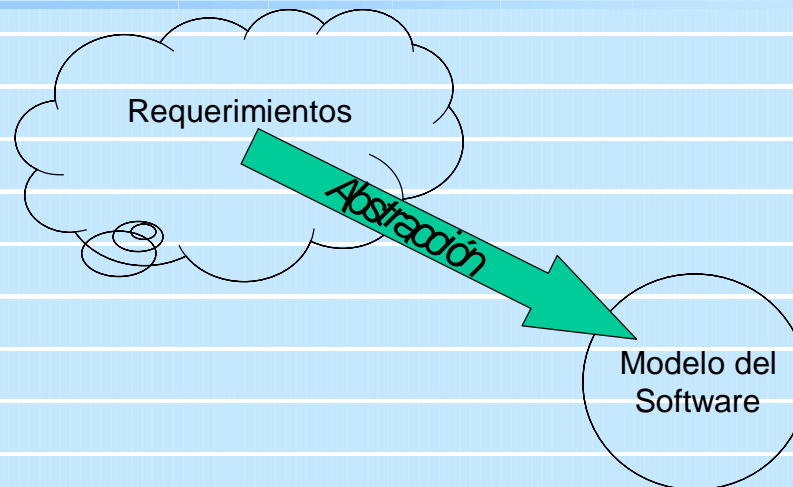
24

Abstracción en Otras Áreas

- Arquitectura
 - el plano de una casa describe:
 - orientación,
 - tamaño relativo de las habitaciones,
 - distribución,
 - pero no incluye:
 - materiales de construcción,
 - colores de las paredes.
- Circuitos eléctricos
 - incluyen:
 - resistencias, capacitores, etc.,
 - ecuaciones,
 - no incluyen:
 - resistencia de los conectores

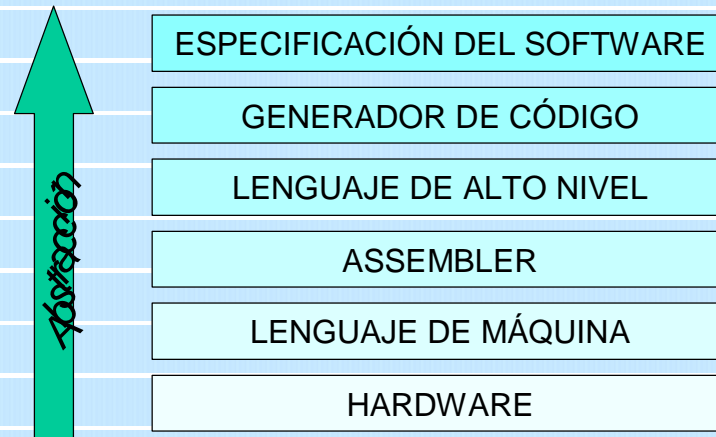
25

Modelo del Software



26

El Software es una Abstracción



27

Abstracción de Producto y Proceso

- Producto de Software:
 - el comentario en el encabezado describe una abstracción de la funcionalidad del procedimiento.
- Proceso de Software:
 - la estimación del costo de desarrollo se realiza considerando los principales factores:
 - personal,
 - capacitación,
 - local,
 - equipamiento;

5. Anticipación del Cambio

- La mantenibilidad es una cualidad deseable del software:
 - corrección de errores,
 - aumentar la funcionalidad.
- Desarrollar software fácilmente mantenible no es gratis:
 - anticipar el cómo y dónde van a ocurrir los cambios,
 - proceder de modo que haga fácil la aplicación de futuros cambios.
- Este principio se aplica generalmente en la etapa de diseño del software:
 - los lugares que van a cambiar se aíslan de modo que los cambios no afecten otras porciones del software.

29

Principio Único del Software

- Las aplicaciones de software se desarrollan a pesar de que sus requerimientos muchas veces no se han comprendido completamente.
- Después de instalado el sistema, el usuario realiza comentarios que dan lugar a cambios y ajustes.
- Las organizaciones se ven afectadas por la instalación de un nuevo sistema, de modo que los requerimientos cambian con su implantación.
- è La anticipación de los cambios es casi la única forma de abordar esta situación eficientemente.

30

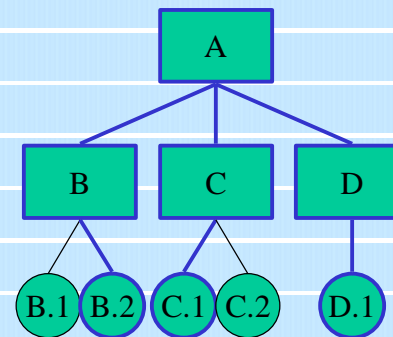
Reutilización y Anticipación del Cambio

- Una componente es reusable si puede ser utilizada para construir un nuevo sistema de software (sin cambios o con cambios mínimos).
- Reutilización es una evolución en pequeña escala, a nivel de componentes.
- Si puede anticiparse los contextos en que la componente será utilizada, se la diseña de modo de poder acomodar fácilmente los cambios necesarios.

31

Control de la Configuración

- Los cambios en las componentes deben ser registrados y almacenados.
- Cada sistema debe tener la versión apropiada de cada una de sus componentes.
- Cada configuración consistente debe también ser almacenada.
- Las herramientas son esenciales.



32

Anticipación en el Proceso

- Deben anticiparse acciones en caso de cambio en el personal.
- También debe considerarse el mantenimiento para la estimación de costos del sistema.
- Debe considerarse la construcción de componentes reutilizables como parte del proyecto o como un proyecto paralelo.

33

Ejemplo: Algoritmo de Sort

- Consideremos un algoritmo de ordenamiento, por ejemplo QuickSort.
- Algunos posibles cambios en una componente que implemente este algoritmo:
 - cambio en el tipo de datos a ordenar,
 - la cantidad de datos es tan grande que deben almacenarse en disco.
- ¿Qué consideraciones haría para prever este tipo de cambios?

34

6. Generalidad

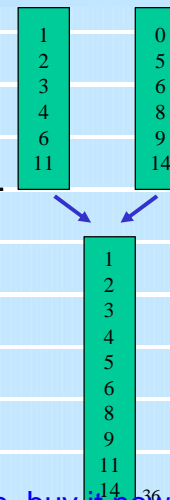
- Cada vez que deba resolver un problema, trate de concentrarse en descubrir un problema más general que se esconde bajo el problema particular.
- Puede suceder que el problema generalizado sea más sencillo que el original.
- Una solución más general es potencialmente más reutilizable.
- Es inclusive posible que ya exista una solución general disponible.
- También es posible que la solución general sea invocada en varias circunstancias en la aplicación en lugar de tener varias soluciones particulares.
- Como contrapartida, una solución general puede ser menos eficiente (ejecución, uso de memoria, tiempo de desarrollo).

➡ Hay que evaluar la conveniencia de una solución general.

35

Ejemplo: Intercalar dos Archivos

- Se desea intercalar dos archivos ordenados para obtener un único archivo ordenado.
- Se sabe que no existen claves duplicadas.
- Si se desarrolla un algoritmo que acepte también claves duplicadas, habrá una mayor cantidad de casos para reusar el algoritmo.
- Además existe un algoritmo para esta tarea que puede tomarse.



36

Generalidad e Industria

- La generalidad es esencial si se intenta desarrollar herramientas o paquetes para comercializar:
 - más generalidad implica más aplicaciones y por lo tanto más potenciales compradores.
- Configurar un producto general es más fácil, rápido y barato que desarrollar una aplicación a medida.
- Tendencia natural en todas las áreas de la ingeniería:
 - ¿ropa a medida?
 - ¿autos a medida?

37

7. Incrementalidad

- Un proceso es incremental si se compone de pequeños pasos (incrementos).
- El objetivo se logra como una sucesión de aproximaciones construidas cada una sobre la anterior.
- En el software, un proceso de desarrollo es incremental si se aplica el modelo Evolutivo de desarrollo.

38

Proceso Incremental



Desarrollo Incremental

- Desarrollo controlado cuando los requerimientos no son estables o completamente claros.
- En la mayor parte de los casos es imposible tener todos los requerimientos al momento de comenzar el desarrollo.
- La clarificación de los requerimientos surge como consecuencia de la experimentación con el sistema.
- La retroalimentación permite desarrollar el sistema.
- Está íntimamente relacionado con la anticipación del cambio.

Incrementalidad de Cualidades

Cualidades
incrementales

- Evolucionable
- Eficiencia
- Amigable
- Robustez
- Confiabilidad

- Los estados intermedios del sistema pueden considerarse prototipos del sistema final.
- El control de la configuración resulta también esencial para administrar la incrementalidad.