Software:
Tecnología para el Procesamiento de Información

Jorge L. Ortega Arjona
Octubre 2017

Introducción

- La humanidad controla **la materia y la energía** mediante ciencia y tecnología.
- En contraste, a la fecha el **procesamiento de información** no había tenido modificación o cambio considerable, debido tal vez a que el cerebro humano es un poderoso medio para el manejo y control de información.

Introducción

Desarrollos anteriores:

- El **lenguaje escrito** (5 mil años aprox.) es la capacidad de registrar información trascendiendo espacio y tiempo: almacenar, recobrar y comunicar información
- Las **operaciones aritméticas simples y la representación numérica** (4 mil años, aprox.) son la habilidad de manipular datos cuantitativos
- Los **métodos de impresión** (500 años) es la creación de copias idénticas del mismo registro, para difundirlo a un número mayor de personas

Introducción

Software: un Nuevo Desarrollo

- Hoy, mucho de la actividad humana depende del **procesamiento de información**
- La información puede ser almacenada, recobrada, comunicada, reordenada, seleccionada, dirigida y transformada y difundida en grandes cantidades y velocidades usando **software**
- Todo procesamiento mecánico y repetitivo de información puede hacerse usando computadoras y **software**
- Cualquier procesamiento, en forma de **una secuencia de operaciones que pueda ser precisamente especificada**, puede realizarse sin mayor intervención humana

Pero ¿qué es Software?

- Ingeniería de Software
- Arquitectura de Software
- Software Propietario
- Software de Aplicación
- Software Base
- Utilerías de Software
- Herramientas de Software
- Software de Control de Sistemas
- Etc…

- Se programa..
- Se desarrolla…
- Se diseña…
- Se compra…
- Se copia…
- Se corre, se ejecuta…
- Se administra…
- Se modela…
- Etc…

¿Qué es Software?

- **Software** es el conjunto de instrucciones que le dicen a la computadora qué hacer
- Actualmente su **importancia es mayor** que la computadora misma: una computadora sin software es tan sólo un máquina inútil compuesta de circuitos electrónicos
- La **cantidad de conocimiento** necesario para crear el software básico que convierta a la máquina en una computadora útil es **comparable** al requerido para **crear la computadora** misma
- El proceso de creación de software, conocido como **programación**, puede ser el alfabetismo del tercer milenio: el conocimiento de software será parte importante de la educación

Programación: Lenguajes de Alto nivel

- Un lenguaje de alto nivel, o algebraico, permite una **representación** del **software** en términos de lenguaje natural humano.
- Lenguajes de alto nivel: Ada, Algol, APL, Basic, C, Cobol, Fortran, Lisp, Pascal, Simula, C++, Java, Haskel, Miranda.

Programación: Lenguajes de Alto nivel

- **Cada instrucción se compone o traduce en un cierto número de instrucciones más sencillas en lenguaje de máquina**
- **La mayoría puede realizar las mismas tareas básicas**
- Se requiere un **software traductor** que convierta un programa fuente en lenguaje de alto nivel a lenguaje de máquina: **intérprete ó compilador**.

Programación: Intérpretes y Compiladores

- **Un intérprete trabaja durante el tiempo de ejecución:** empezando con la primera instrucción, va traduciendo una a una las instrucciones a lenguaje de máquina, **ejecutándolas enseguida**
- **Un compilador traduce todo el programa fuente a un programa objeto en lenguaje de máquina.** El programa se traduce fuera de tiempo de ejecución. El programa objeto se almacena, y **se ejecuta al invocarlo o cargarlo en memoria**.

Programación: Estructuras de Datos

- Cualquier lenguaje de alto nivel puede usar **estructuras de datos** (formas de organizar datos en memoria: arreglos, listas, cadenas, árboles, colas y pilas)
- La eficiencia en la operación sobre los datos depende de la **representación** de las estructuras de datos, y la manera como organiza su **almacenamiento** en memoria

No Silver Bullet

Essence and Accidents of Software Engineering

Fred Brooks Jr. (1987)

- "Fashioning complex conceptual constructs is the *essence*; *accidental* tasks arise in representing the constructs in language. Past progress has so reduced the accidental tasks that future progress now depends upon addressing the essence."
- "La elaboración de construcciones conceptuales y complejas es la **esencia**; las tareas **accidentales** surgen al interpretar construcciones en un lenguaje. El progreso en el pasado ha reducido tanto las tareas accidentales que el progreso futuro depende ahora en abordar la esencia."

No Silver Bullet

Essence and Accidents of Software Engineering

Fred Brooks Jr. (1987)

- **Dificultades Esenciales:** inherentes de la naturaleza del software
- "La esencia de una entidad de software es una construcción de conceptos interconectados: conjuntos de datos, algoritmos, e invocaciones a funciones. Esta esencia es abstracta dado que la construcción conceptual es la misma bajo diferentes representaciones"
- **Dificultades accidentales:** atienden a la producción de software, pero no son inherentes.

"I believe the hard part of building software to be the specification, design and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation"

No Silver Bullet

Essence and Accidents of Software Engineering

Fred Brooks Jr. (1987)

Dificultades Esenciales

- **Complejidad**. No hay dos partes iguales (a nivel instrucción); gran número de estados; no escalable linealmente.
- **Conformidad.** Interfaces entre instituciones humanas a las que debe adecuarse. La variación en interfaces no se debe a necesidad, más bien porque han sido diseñados por diferentes personas.
- **Cambiabilidad.** El software cambia constante debido a que incorpora funcionalidad y porque puede ser "fácilmente" cambiado.
- **Invisibilidad**. El software es invisible y no visualizable. Las abstracciones geométricas no capturan su esencia, ya que el software se encuentra inherentemente inmerso en la memoria.

No Silver Bullet

Essence and Accidents of Software Engineering

Fred Brooks Jr. (1987)

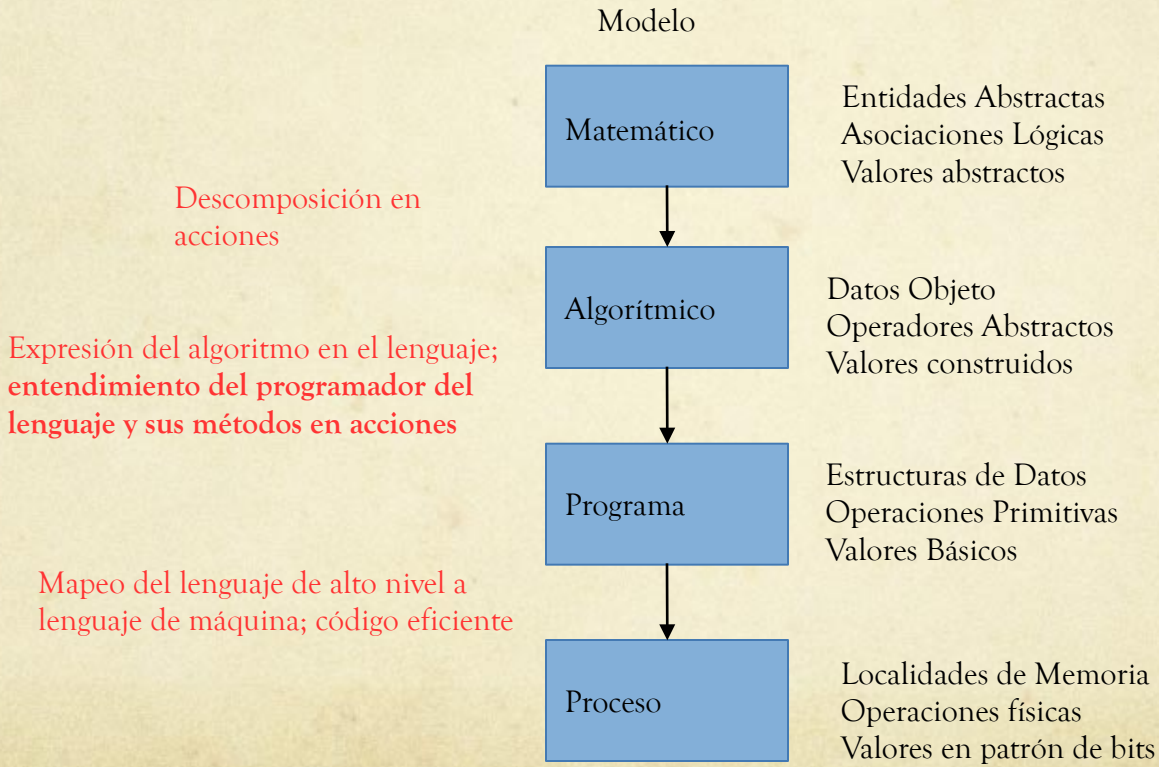
Logros que resolvieron Dificultades Accidentales

- **Lenguajes de Alto nivel.** Liberan al software de su complejidad accidental, dando al programa abstracciones manejables
- **Tiempo Compartido.** Preserva la inmediación, lo que permite mantener en mente las complejidades del software
- **Ambientes unificados de programación.** Atacan la dificultad accidental de pensar individualmente en programas

Esperanzas: Ada y otros lenguajes de alto nivel, **Programación Orientada a Objetos**, Inteligencia Artificial, Sistemas Expertos, Programación "Automática", Programación Gráfica, Verificación de Programas, Ambientes y Herramientas, Estaciones de Trabajo

El Concepto de Proceso

Proceso: Cambio en el estado de la memoria por acción del procesador



*C.M. Pancake & D. Bergmark (1990)

Arquitectura de Software

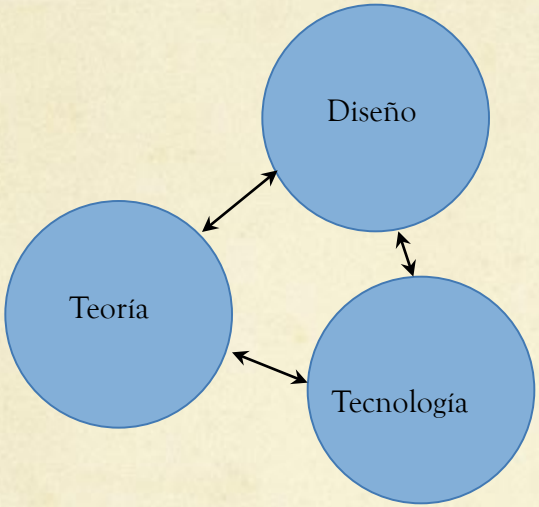
Sistema de software: colección, integración o ensamble de (un gran número de) partes o componentes independientes de software

- **Complejo:** compuesto de partes interconectadas o entremezcladas

Arquitectura de software: disciplina o estudio de la descripción de sistemas de software como resultado del ensamblado de componentes de software

- **Abstracción:** eliminar detalles irrelevantes en la descripción de un sistema

Arquitectura de Software



Diseño de software: estudio de los principios fundamentales y técnicas de composición para crear sistemas de software, y sus descripciones formales e informales

Teoría de software: conjunto de conceptos y términos usados para definir las partes e interacciones de los sistemas de software

Tecnología de software: descripciones de cómo los sistemas de software se implementan, y las tecnologías relacionadas tanto en hardware como software

¿Qué es Software, de nuevo?

Software:

Es un “material” del que las funciones de procesamiento se construye

Es algo usado en la computadora, pero es más que un autómata programado

Es un aspecto clave de cómo los programadores organizan una función en términos de una forma, mediante composición

Es más que código neutral: las preguntas son **¿cómo puede organizarse? ¿qué forma tiene esa organización?**

Diseño de Software

Debido a su complejidad, el Software es un reto para la aplicación de conocimiento de Diseño

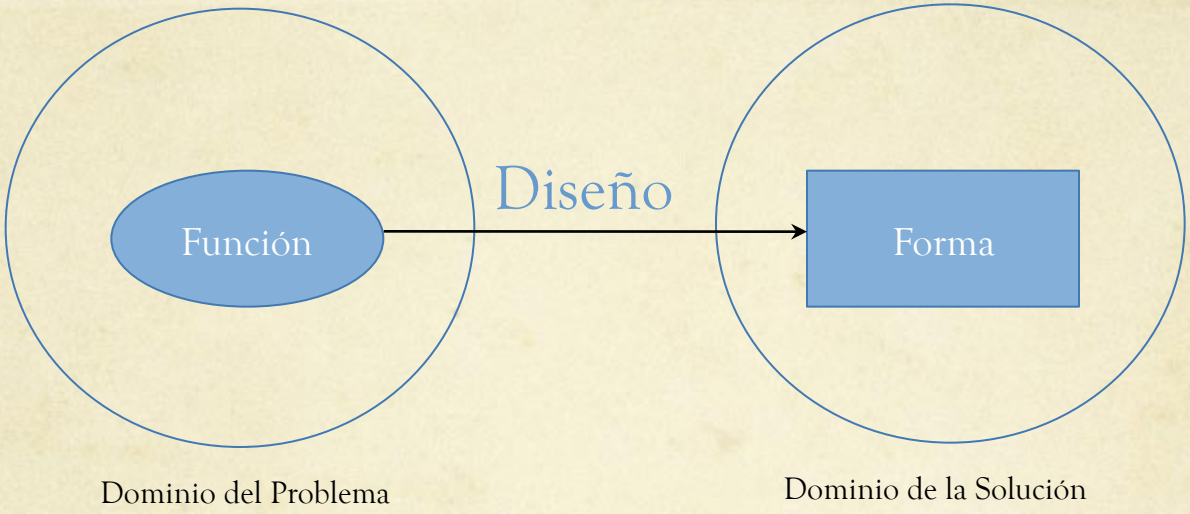
El Software:

Provee una funcionalidad extendible por otras piezas de software, que representan otras funcionalidades

Es una entidad técnica y económicamente evolucionable

No es una entidad independiente, sino parte de un sistema de hardware y software

Diseño en general y Diseño de Software



Diseño es encontrar una forma que satisfaga a la función y a sus requerimientos. Es una actividad que obtiene una forma a partir de la descripción de una función.

En Software, forma y función pueden verse como organizaciones. Una organización de la función describe al software a nivel comportamiento; una organización de la forma describe al software a nivel estructura

Diseño de Software es una relación de una clase de organizaciones posibles de la función y una clase organizaciones posibles de la forma

Diseño en general y Diseño de Software

Diseño de Software se realiza mediante:

(a) organizar los componentes y conexiones de software para formar un sistema

(b) representar al propio sistema de software

(c) organizarse dentro de un método

El problema se describe como una función y cómo debe realizarse, es decir sus requerimientos: un algoritmo y datos

La solución se describe como una forma y sus propiedades: componentes de software y relaciones o conexiones entre ellos

Idealmente, requerimientos y propiedades se mapean entre sí

La situación en Diseño de Software

El software solo puede accederse mediante descripciones como código fuente o código ejecutable, lo que lo hace no-visible

¿Qué tiene el diseño de otros artefactos que le hace falta al software?

- Una base de experiencia y técnicas de diseño de software
- Una representación tangible del producto, particularmente su estructura
- Mediciones y evaluaciones para determinar si los requerimientos deseados se encuentran como propiedades en el producto final

Una base de experiencia y técnicas

Industrias maduras tienden a generar manuales de experiencia y técnicas de diseño, describiendo soluciones exitosas a problemas conocidos

Diseñadores e ingenieros rara vez comienzan sus diseños desde cero, sino que reusan soluciones de diseño conocidas con un registro de éxitos

En tal sentido, los **Patrones de Software** y la **Comunidad de Patrones** representan el más claro intento de colectar experiencia y técnicas de Diseño de Software

Un Patrón de Diseño de Software es **una relación forma-función que ocurren en un contexto en el cual la función se describe en términos del dominio del problema como un conjunto de fuerzas, y la forma es una estructura descrita en términos del dominio de la solución que logra un equilibrio aceptable entre las fuerzas**

Una descripción tangible

Una descripción tangible del software normalmente consiste de los componentes de software, las conexiones de software y una frontera del sistema de software

Los componentes de software se hacen tangibles mediante encerrarlos en contenedores de implementación

Todo nivel de contenedores de implementación tienen una relación de contención entre sí

La necesidad de mediciones

La principal razón para un aproximación de Diseño de Software es mejorar la calidad del software

"Calidad de software es el grado con que el software posee una combinación deseada de atributos"

Medición y evaluación. Habilidad de medir cualidades y evaluarlas para confirmar su presencia o ausencia en otros sistemas de software. **Repetitibilidad**: los resultados pueden reproducirse en sistemas de software similares

Evaluación de alternativas. Como potenciales soluciones, los sistemas de software deben revisarse respecto a su habilidad de satisfacer la funcionalidad y los requerimientos

Una Invitación

- El mundo de la programación de software es fascinante, intrincado, gratificante, y a la vez inmisericorde, complicado y demandante de grandes esfuerzos y toneladas de paciencia
- Quisiera invitar a todo aquél con una computadora a su disposición a escribir un programa, o dos
- La sensación de logro es grandiosa cuando se observa en acción un programa que uno ha pensado, trabajado y construido
- Tal vez es porque el proceso es adictivo, y consume mucho más tiempo de lo que se cree en un principio
- Al final, lo cierto es que en cada programador hay un optimista, siempre pensando que cada error en el programa es el último.

¿Preguntas?