# PRINCIPIOS DE DISEÑO DE SMALLTALK

- El propósito de Smalltalk es proveer soporte de computación a nuestro espíritu creativo.

- Dos áreas principales:
  - Lenguaje de descripción (lenguaje de programación) que sirve de interfaz entre los modelos en la mente humana y los del hardware.
  - Lenguaje de interacción (interfaz al usuario) que adapte el sistema de comunicación humano a la computadora.

- Dominio personal: Si un sistema es para servir al espíritu creativo, debe ser entendible para un individuo solitario.

- Buen diseño: Un sistema debería ser construido con un mínimo conjunto de partes no modificables, tan generales como sean posibles y mantenidas en un esquema uniforme.

## LENGUAJE

- Al diseñar un lenguaje, todo lo que sabemos sobre cómo la gente piensa y se comunica es aplicable.

- Propósito del lenguaje: Proveer un esquema para la comunicación. Comparado con una persona, en las computadoras el "cuerpo" provee una pantalla visual de información, y percepción de la entrada del usuario humano. La "mente" incluye los elementos de memoria y procesamientos internos y sus contenidos.

- Alcance: El diseño de un lenguaje debe tratar con modelos internos, medios externos, y con la interacción entre ellos.

## OBJETOS QUE SE COMUNICAN

- Objetos: Un lenguaje debe soporter el concepto de "objeto" y proveer una manera uniforme de referirse a los objetos del universo.
  El administrador de almacenamiento de Smalltalk provee un modelo orientado a objetos de la memoria de todo el sistema. (Se asocia un entero que no se repite a cada objeto). Cuando todas las referencias a un objeto desaparecen del sistema, el objeto se esfuma, y se recupera su espacio de almacenamiento.

- Administración del almacenamiento: Para ser "orientado a objetos" un sistema debe proveer administración automática del almacenamiento.

- Mensajes: La computación debería ser vista como una capacidad intrínseca de los objetos que pueden ser invocados uniformemente enviándoles mensajes.

Smalltalk envía el nombre de la operación deseada, junto a cualquier parámetro necesario, como un mensaje a, por ejemplo, un número, entendiendo que el número sabe cómo realizar la operación deseada.

- Metáfora uniforme: Un lenguaje debería ser diseñado alrededor de una metáfora poderosa que pueda ser aplicada uniformemente en todas las áreas.
  Todo objeto en Smalltalk, tiene un conjunto de mensajes, un protocolo, que define la comunicación explícita a la que el objeto puede responder. Internamente, los objetos pueden tener almacenamiento local y acceso a otra información compartida que comprenden el contexto implícito de toda comunicación.

**ORGANIZACIÓN**

- Modularidad: Ningún componente en un sistema complejo debería depender de los detalles internos de ningún otro componente.
  La metáfora de envío de mensajes provee modularidad al desacoplar la intención del mensaje (incorporado en el nombre) del método usado por el receptor para realizar la intención. La información estructural es similarmente protegida porque todo acceso al estado interno de un objeto es a través de esta misma interfaz de mensajes.
  El agrupamiento de componentes similares es a través de clases en Smalltalk. Una clase describe otros objetos (su estado interno, protocolo de mensajes que reconocen y métodos internos para responder.). Los objetos así descriptos se llaman instancias de la clase (Las propias clases son instancias de la clase Class)

- Clasificación: Un lenguaje debe proveer un medio para clasificar componentes similares y para agregar nuevas clases de objetos en pie de igualdad con las clases centrales del sistema.

- Polimorfismo: Un programa solo debería especificar el comportamiento esperado de los objetos, no su representación.

- Factorización: Cada componente independiente de un sistema solo debería aparecer en un solo lugar. Una falla en la factorización implica una violación de la modularidad. Smalltalk promueve diseños bien factorizados a través de la herencia. Todas las clases heredan su comportamiento de su superclase. Esta herencia se desarrolla a través de clases cada vez más generales terminando con la clase Object que describe el comportamiento mínimo de todos los objetos del sistema.

- Reaprovechamiento: Cuando un sistema está bien factorizado, un gran reaprovechamiento está disponible para los usuarios e implementadores.

- Máquina virtual: Una especificación de máquina virtual establece un marco para la aplicación de tecnología.
  La máquina virtual de Smalltalk establece un modelo orientado a objetos para el almacenamiento, uno orientado a mensajes para el procesamiento y uno de bitmap para el despliegue visual de información.

**INTERFAZ AL USUARIO**

- Es un lenguaje en donde la mayor parte de la comunicación es visual. La estética y la flexibilidad son esenciales.

- Principio reactivo: Cada componente accesible al usuario debería ser capaz de presentarse de manera entendible para ser observado y manipulado.

- Sistema operativo: Colección de cosas que no encajan dentro de un lenguaje. No debería existir.
  Smalltalk no tiene "sistema operativo" como tal. Las operaciones primitivas necesarias, como leer una página del disco, son incorporadas como métodos primitivos en respuesta a mensajes Smalltalk normales.

**TRABAJO FUTURO**

- El sistema Smalltalk-80 es deficiente en su factorización porque sólo soporta herencia jerárquica. Futuros sistemas Smalltalk generalizarán a herencia múltiple.

- El protocolo de mensajes no se ha formalizado.

- Selección natural: Los lenguajes y sistemas que son de buen diseño persistirán, sólo para ser reemplazados por otros mejores.