Resumen Teórica 1 : Small Talk Historia

Tomás Felipe Melli

March 31, 2025

Índice

1 Historia

1.1 Simula 67

Simula 67 es un lenguaje de programación que se desarrolló en Noruega a fines de la década de 1960. Fue creado por Ole-Johan Dahl y Kristen Nygaard y es considerado el primer lenguaje de programación orientado a objetos.
El objetivo original de Simula 67 era modelar sistemas complejos como simulaciones de eventos discretos, en áreas como la investigación de sistemas de comunicación y transporte. El lenguaje introdujo conceptos fundamentales que más tarde se popularizarían en otros lenguajes, como clases y objetos, que son la base de la programación orientada a objetos.
Las principales características que tuvo *Simula 67* fueron :

- **Clases y objetos**: Permite la definición de clases que pueden tener métodos y atributos, así como instancias (u objetos) de esas clases.
- **Herencia**: Introdujo la posibilidad de que una clase derive de otra, lo que permite la reutilización del código.
- **Manejo de procesos concurrentes**: Permite simular procesos en paralelo, algo que también fue novedoso en su época.

1.2 Paradigma Estructurado

El paradigma de programación estructurada surgió a finales de la década de 1960 y principios de la de 1970 como una respuesta a los problemas de mantenimiento y comprensión de los programas escritos en estilos de programación más caóticos, como el uso excesivo de go to (saltos incondicionales) en lenguajes como Fortran o Basic.
El movimiento de la programación estructurada fue liderado por dos figuras clave
Edsgar Dijkstra: En 1968, Dijkstra publicó un influyente artículo titulado **"Go To Statement Considered Harmful"**, en el que argumentaba que el uso indiscriminado de la instrucción goto hacía los programas más difíciles de leer, mantener y depurar. Dijkstra abogó por el uso de estructuras de control de flujo más claras y estructuradas, como bucles (loops) y condicionales (if-else), en lugar de saltos incontrolados de ejecución. **Niklaus Wirth**: En 1971, Wirth introdujo el lenguaje Pascal, que fue diseñado siguiendo principios de la programación estructurada. Pascal promovió el uso de estructuras de control como secuencias, decisiones y bucles, y desalentaba el uso de goto.
Las principales características del paradigma estructurado son :

1. **Eliminación de los saltos incontrolados (goto)**: En lugar de usar goto para saltar de una parte del código a otra, se promovió el uso de estructuras de control claras como if, while, for, y switch.
2. **Modularidad**: El código se organizaba en bloques lógicos o funciones/métodos, lo que hacía que fuera más fácil de entender y mantener.
3. **Secuencialidad**: La ejecución del programa sigue una secuencia lógica, donde se pasan por los bloques de código de arriba hacia abajo, salvo cuando se interviene en el flujo mediante estructuras de control.

1.3 SmallTalk

Smalltalk es un lenguaje de programación orientado a objetos que fue desarrollado a principios de la década de 1970 en el Centro de Investigación de Ciencias de la Computación en Xerox PARC, principalmente por Alan Kay, Dan Ingalls, Adele Goldberg, y otros. Se considera uno de los lenguajes más influyentes en la historia de la informática y el primero en implementar completamente el paradigma de la programación orientada a objetos (POO), mucho antes de que se popularizara.
Smalltalk fue diseñado inicialmente como un lenguaje educativo y como parte de un proyecto llamado **Dynabook**, que Alan Kay visualizaba como una especie de "computadora personal portátil". El objetivo era crear un entorno de programación en el que los niños pudieran aprender fácilmente conceptos de computación y programación.
Las características principales son:

1. **Todo es un objeto**: En Smalltalk, todo, desde números hasta clases y estructuras de control, es un objeto. Este enfoque radical en el que incluso los conceptos más abstractos se tratan como objetos fue una de las grandes innovaciones de Smalltalk.
2. **Modelo de objetos y clases**: El lenguaje implementa el modelo de programación orientada a objetos de manera pura. Cada objeto es una instancia de una clase, y las clases pueden heredar de otras clases, lo que permite la reutilización de código.
3. **Mensajes**: En lugar de invocar funciones o métodos como en otros lenguajes, en Smalltalk los objetos "envían mensajes" a otros objetos. Un mensaje es una solicitud para que el objeto receptor ejecute una acción.

4. **Entorno gráfico y de desarrollo**: Smalltalk fue también pionero en ofrecer un entorno de desarrollo interactivo que permitía a los programadores escribir código, ejecutarlo y modificarlo en tiempo real. Su entorno visual fue una de las primeras implementaciones de interfaces gráficas de usuario (GUI), y su entorno de programación fue una inspiración para la creación de otros entornos como Macintosh y Windows.

5. **Garbage collection**: Smalltalk implementó uno de los primeros sistemas automáticos de garbage collection para la gestión de memoria.

1.4 LISP

Lisp fue reado por John McCarthy en 1958, es uno de los lenguajes de programación más antiguos y es conocido por ser un lenguaje **funcional**. Su enfoque está en el procesamiento de listas y en la manipulación simbólica, lo que lo hizo muy popular en el ámbito de la inteligencia artificial.
Lisp fue diseñado para tratar de manera eficiente las estructuras simbólicas, usando una notación basada en paréntesis (conocida como notación de prefijo o notación polaca). Su flexibilidad y extensibilidad lo convirtieron en una opción poderosa para resolver problemas complejos.
Existe un vínculo con SmallTalk

- Interactividad y Entornos de Desarrollo Dinámicos: Tanto Lisp como Smalltalk impulsaron el desarrollo de entornos de desarrollo interactivos. En Lisp, especialmente en sus variantes más modernas, como Common Lisp o Scheme, el código se evalúa en tiempo real, permitiendo la modificación y ejecución inmediata de programas. Smalltalk también fue pionero en el uso de entornos gráficos e interactivos donde el código se podía editar y ejecutar de manera inmediata, lo que facilitaba la exploración y el desarrollo en tiempo real.

- Flexibilidad y Extensibilidad: Ambos lenguajes son conocidos por su capacidad de extenderse y adaptarse fácilmente a nuevas necesidades. Lisp tiene un sistema de macros poderoso, que permite a los programadores crear nuevos constructos de lenguaje. Smalltalk, por su parte, permite modificar y extender su propio entorno de desarrollo y el lenguaje mismo, añadiendo nuevas clases y métodos sobre la marcha.

- Paradigmas de Programación: Aunque Lisp es principalmente un lenguaje funcional y Smalltalk orientado a objetos, ambos lenguajes tienen una gran flexibilidad paradigmática. Por ejemplo, en Lisp, se pueden crear representaciones orientadas a objetos usando estructuras como listas o árboles, y en Smalltalk, aunque es puramente orientado a objetos, puede incorporar técnicas funcionales en su programación.

2 Filosofía

1. **Programación como una forma de pensar** : Alan Kay, uno de los creadores de Smalltalk, dijo que quería que la programación fuera una forma de pensar sobre el mundo. En lugar de programar con una serie de instrucciones secuenciales (como en los lenguajes estructurados), el enfoque de Smalltalk invita a pensar en términos de modelos del mundo real. Los objetos en el programa representan cosas o conceptos del mundo real, y la forma en que interactúan entre sí refleja cómo esas entidades interactúan en la vida cotidiana.

    - **Modelado del mundo real**: Cada objeto puede ser considerado como una "caja" con sus propios datos y comportamientos, y el sistema entero está compuesto por objetos que interactúan entre sí. Este enfoque facilita la creación de simulaciones y modelos de sistemas reales, lo que es ideal para aplicaciones como simuladores o interfaces gráficas

2. **Énfasis en la comunicación**:En Smalltalk, la clave de la programación no está en la manipulación de datos como en otros paradigmas, sino en la comunicación entre objetos. La programación se convierte en un proceso de diseñar un sistema en el que los objetos envían mensajes entre sí para ejecutar acciones y comportarse de una manera determinada. Este enfoque refleja cómo las entidades en un sistema pueden estar interrelacionadas y cómo interactúan.

3. **Desarrollo iterativo y exploratorio**:Smalltalk promueve un enfoque iterativo y exploratorio de la programación. Los desarrolladores pueden experimentar con el código de manera dinámica, agregando y modificando comportamientos sin necesidad de detener todo el sistema. Esto alienta a un desarrollo más experimental, donde las soluciones se refinan y ajustan con el tiempo.

4. **Enfoque educativo**:Fue diseñado para ser accesible y fácil de aprender, con el fin de permitir que personas sin experiencia previa en programación pudieran entender conceptos complejos como la programación orientada a objetos. Este enfoque también se reflejó en el Dynabook, un concepto precursor de las computadoras portátiles que podía ser utilizado para la educación.

2.1 Piaget, el Constructivismo y SmallTalk

El enfoque de Jean Piaget y la teoría constructivista se encuentra en la visión de cómo los individuos aprenden y comprenden el mundo a través de la interacción y exploración activa.

2.1.1 Piaget y el Constructivismo

Jean Piaget fue un psicólogo suizo cuyo trabajo se centró en cómo los niños desarrollan el conocimiento. Su teoría del constructivismo sostiene que el aprendizaje es un proceso activo donde los individuos construyen su comprensión del mundo a través de experiencias y experimentos. Piaget argumentó que el conocimiento no se transmite simplemente de una persona a otra, sino que se construye activamente en la mente del aprendiz, a medida que interactúa con su entorno.

- **Aprendizaje activo**: Piaget enfatizó que los niños aprenden mejor cuando interactúan con su entorno y resuelven problemas por sí mismos, en lugar de ser simplemente receptores de información.
- **Construcción de esquemas mentales**: El proceso de aprendizaje implica la creación de esquemas mentales o estructuras cognitivas, que luego se modifican y amplían conforme el niño interactúa con nuevas experiencias.
- **Desarrollo cognitivo**: Piaget vio el desarrollo cognitivo como un proceso continuo de asimilación (integración de nuevas experiencias en esquemas existentes) y acomodación (ajuste de los esquemas a nuevas experiencias).

Esto resuena con todo lo dijimos de SmallTalk.