SmallTalk Design Principles

The purpose of the SmallTalk project was to provide computer support for creative spirit.

- **Personal Mastery:**
  - If a system is to serve the creative spirit, it must be entirely comprehensible to a single individual.
- **Good Design:**
  - A system should be built with a minimum set of **unchangeable** parts; those parts should be as **general** as possible; all parts of the system should be held in a uniformed framework.

(Language)

- It will save time if we make our computer models compatible with human minds, rather than the other way round.

- **Purpose of the Language:**
  - To provide a framework for communication.
- **Scope:**
  - The design of a language for using computers must deal with internal models, external media and the interaction between these in both the human and the computer.

(Communicating Objects)

- Oneness vs. Distinction

- **Object:**
  - A computer language should support the concept of an "object" and provide a uniform means for referring to objects in its universe.
- **Storage Management:**
  - To be truly "object-oriented", a computer must provide automatic storage management.
- **Messages:**
  - Computing should be viewed as an intrinsic capability of objects that can uniformly invoked by sending messages.
  - The receiver knows best how to carry out the desired operation.
  - A universe of well-behaved objects that courteously ask each other to carry out their various desires.
  - The transmission of messages is the only process that is carried out outside of the objects and this is as it should be.

- **Uniform Metaphor:**
  - A language should be designed around a powerful metaphor that can be uniformly applied in all areas.
  - Large applications are viewed in the same way as fundamental units from which the system is built.
  - Every object has a **protocol**.

(Organization)

- A *uniform metaphor* provides a framework in which complex systems can be built.
- **Modularity:**
  - No component in system should depend on the internal details of any other components.
- **Classification:**
  - A language must provide a means for classifying similar objects, and for adding new classes of objects on equal footing with kernel classes of the system.
  - Classification is objectification of *ness*ness.
- **Polymorphism:**
  - A program should specify only the behavior of objects, not their representation.
  - Generic description is crucial to the models of the real world.
- **Factoring:**
  - Each independent component in system appears in only one place.
  - Well-factoring design through *Inheritance*.
- **Leverage:**
  - When a language is well factored, great leverage is available for users and implementers alike.
- **Virtual Machine:**
  - It is natural to ask what set of primitive operations would be sufficient to support an entire computing system.
  - A virtual machine specification establishes a framework for the application of technology.
  - The Smalltalk virtual machine establishes an object-oriented model for storage, a message-oriented model for processing, and a bitmap model for visual display of information. Through the use of micro-code, and ultimately hardware, system performance can be improved dramatically without any compromise to the other virtues of the system.

(User Interface)

Visual Communication

- **Reactive Principle:**
  - Every component accessible to the user should be able to present itself in a meaningful way of observation and manipulation.

(Future Work)

- **Natural Selection:**
  - Languages and systems that are of sound design will persist, to be supplanted by only better ones.