INGALLS81
Ingalls Daniel H.H.; Design Principles Behind Smalltalk; Byte 6, 8 (August 1981).

Summary of Ideas

A system should be built which can be fully understood by one person and which has a minimum set of unchangeable parts which are general and uniform.

Computer languages should provide a communications framework similar to that used in human communication. These languages should support the idea of an object, provide a uniform means for referring to objects and provide automatic storage management of objects.

Objects should be sent messages indicating actions to be performed by them.

"A language should be designed around a powerful metaphor that can be uniformly applied in all areas".

Modularity is needed to assist in complex human tasks. Similarly, the ability to classify similar objects is inherent in human thinking and should be present in a computer languages (via classes and inheritance).

Programs should only specify behaviour and not representation of objects.

Independent components in a system should be factored out so to appear in only one place. This can be done with classes and inheritance.

"Every component accessible to the user should be able to present itself in a meaningful way for observation and manipulation". This is done with the message protocol for objects.

"An operating system is a collection of things that don't fit into a language. There should not be one."

Important Points

If a single person cannot entirely understand a system then there will be an impediment to creative expression. Similarly, a system that cannot be changed or is not sufficiently general or consistent will also pose an impediment. Small talk thus has a minimum set of unchangeable parts.

Smalltalk provides an object oriented model with automatic storage management and a message sending technique as the means is initiating actions.

Like LISP which is modelled on lists and APL which is modelled on arrays, Smalltalk is built around the powerful uniform metaphore of communicating objects. Thus large applications are viewed in exactly the same way as the fundamental units of the system.

Both modularity and classification with inheritance allow Smalltalk to be simple yet manage complexity very well. Specifying the behaviour and not the representation of objects also allows great flexibility for further extension of the language.

Factoring is encouraged by the fact that a class in Smalltalk inherits behaviour from its superclass. Because the system is built on a small set of primitive operations, a small improvement in the performance of one of these operations will yield great improvement in the system.

Smalltalk has no operating system as far as other languages are concerned, rather it incorporates operating system primitives in the Smalltalk system itself.

Relevance

This article outlines the philosophy behind object oriented languages in general and Smalltalk in particular. Object oriented languages are a departure from the mainstream language model and this paper gives some of the justifications for doing so.