Elements Report SELF

Kyle Dymowski, Aidan Miller

April 14, 2016

1 The Team

Two peope are accredited with the development of SELF, David Ungar and Randy Smith, they began working on the language in 1986 while working at Xerox PARC then in 1987 they moved to Stanford University to build the compiler.

1.1 David Ungar

Dr. Ungar pursued his Ph.D in Computer Science at the University of California Berkeley from 1980 to 1985. Where he participated in architectural design of SOAR, Smalltalk on a RISC microprocessor. He also modified a simulator and virtual machine to benchmark architectural ideas, and designed, analyzed, implemented and measured the first two generation garbage collector. In 1985 he joined Stanford University as a Assistant Professor where he led the Self team, invented mirror-based reflection architecture and a storage system for prototype-based languages. He then began working at Sun Microsystem Laboratories in 1991 where he continued working on self until 1995. He left sun enterprises in 2006 and started working at IBM where he currently works on the Watson project. [1]

1.2 Randy Smith

Dr. Smith started his education with a double major in physics and mathematics with the highest honor of summa cum laude in 1974 at the University California, Davis. He then pursued his PhD in Theoretical Physics graduating in 1981 from the University of California at San Diego. He began working on self in 1986 at Xerox PARC, then continued his work at Stanford in 1987, and finally finished the project at Sun Microsystems Laboritory in 1991. Where he still works today as a consulting member of the technical staff, he is currently working in the areas of modelling, simulation and optimization.

2 Innovations

The team that designed Self recognized that object-oriented programming languages were gaining acceptance and becoming widely used, because they offer useful implementation for designing programs. Ungar and team had a different idea on object oriented disign leading them to create the first prototype based language. Object oriented languages inherently have uniform access to different kinds of stored and computated data, SELF was looking to extend this and create more expressive power. [2]

In SELF, and any other prototype based language, everthing is an object, but the difference from languages like C++ is instead of an instance being either a instance of, or a subclass of, it inherits from the relationship by cloning the object. This is how SELF eliminates classes, and creates a "inherits from" relationship that describes how objects share behavior and state. This is done in the cloning step that happens during instantiation, where information formats are copied. This is how SELF creates one of a kind objects that have there own behavior. Each object is stored in a named slot that can hold state and behavior. Becuase every slot is unique it can hold its own states and behaviors, unlike class based systems that have multiple mutliple objects with the same behavior. Classes have no real support for an object to posses it's own behaviors, and there is no way to garuentee that there is only one instance of the class. SELF has neither of these problems, and there is no need to create a new class to extend the behavior. [2]

References

[1] David Ungar Software innovator
https://www.linkedin.com/in/davidungar 2016

[2] David Ungar, Randall B. Smith. SELF: The Power of Simplicity. Xerox Palo Alto Research Center, Palo Alto, California 94304