**OBJECT-ORIENTATION: MOTIVATIONAL STUFF**

William Bricken
October 1984

This list of ideas and bullets serves as an introduction to benefits of system-oriented-languages (SOL).

## QUALITIES

***UNIQUE:*** offers model building capabilities not available in any other expert system architecture

- -- unified support of multiple representations
- -- easy to change modeling metaphor
- -- supports multiple viewpoints or models,
  and multiple levels of abstraction in a model
- -- explicit control of the behavior of objects
- -- coordinated action between objects via types of communication (sends)
- -- intelligent object behavior via internal processors for each object
- -- intelligent object creation via explicit control of inheritance

***USABLE:*** offers programming capabilities that support state-of-the-art modeling and programming techniques

- -- highly interactive
- -- graphic display of system states and processes
- -- rapid proto-typing and development
- -- explicit control of programming environment and granularity
- -- tools to support
  - object-oriented programming
  - frame-based programming
  - rule-based programming
  - probability propagation
  - parallel hypotheses
  - parallel execution

## TOP-LEVEL DESCRIPTION

OBJECT ORIENTED PROGRAMMING is a style of code-writing. The programmer thinks about the domain in terms of objects and generic operations on the objects.

**BENEFITS**

- -- conceptual clarity, naturalistic modeling
- -- enforced step-wise modularity
- -- generic operations on abstract objects
- -- rapid experimental prototyping
- -- overt debugging
- -- evolution of programming goals

SOLs are SYSTEMS ORIENTED LANGUAGES for object-oriented programming. Data objects within a SOL are active systems; their behavior is explicitly controllable as if each were an operating system.

**ADDITIONAL BENEFITS**

- -- integration of object and process
- -- extended modeling capabilities
- -- modularization of control information
- -- explicit representation of context and history
- -- generalized architecture for decision making

## TOP-LEVEL DESCRIPTION WITH COMMENTS

*OBJECT ORIENTED PROGRAMMING* is a style of code-writing. The programmer thinks about the domain in terms of objects and generic operations on the objects.

"The advantages of object oriented programming make it the state-of-the-art programming style."

**BENEFITS**

-- *conceptual clarity, naturalistic modeling*

"People think in objects. The behavior of a program written in objects is easier to follow and to explain, and once the programmer is familiar with the style, coding and debugging is easier."

-- *enforced step-wise modularity*

"Objects are a natural partition of the space of representations. Changing an object or its behavior is an incremental process, that does not affect the procedural parts of the code. In fact, this style separates the domain representation from the procedural code and inference mechanisms."

-- *generic operations on abstract objects*

"Each type of object has METHODS which define the behavioral aspects of the object. Methods apply to all instances with an abstract characteristic. For example, the Movement Method for the abstract class of MOVING OBJECTS can calculate velocity, and this method can be used for anything that moves. Duplicate code for moving Boats and moving airplanes is avoided."

-- *rapid experimental prototyping*

"Object attributes and methods can be quickly changed to find out the characteristics of different domain models. No associated procedural code needs to change."

-- *overt debugging*

"When an object does not act appropriately, you can go directly to the code for the method that creates that action. Searching for bugs is greatly reduced."

-- *evolution of programming goals*

"Consistent with the iterative approach, small subsets of the problem can be coded and evaluated. If the objects are not what is wanted, they are easy to change. Little effort is lost when the objectives or specifications for the objects are changed."

SOLs are *SYSTEMS ORIENTED LANGUAGES* for object-oriented programming. Data objects within a SOL are active systems; their behavior is explicitly controllable as if each were an operating system.

"The philosophy is to make each object into an entire SYSTEM. Methods, or behaviors, can be given explicit control structures, so that each object becomes an intelligent object, or entity."

**ADDITIONAL BENEFITS**

-- *integration of object and process*

"A system-object has process characteristics, just like an operating system."

-- *extended modeling capabilities*

"We can model the behavior of intelligent objects, and address a wide range of problems in a supportive coding environment."

-- *modularization of control information*

"Control is localized in objects, and is not global. Control paradigms can change depending on the context."

-- *explicit representation of context and history*

"A system-object is a context. When the object is active, it is the active context. Global context tracking is not necessary. Each object keeps its own record of it previous behaviors, so a burdensome global history is not needed."

-- *generalized architecture for decision making*

"Other architectures are a subset of system-oriented-languages. Decision-making can be modeled as any interaction of intelligent participants."