Tech Info Library

AI: Some Working Definitions Of The Terminology (Part 2 of 2)

Revised: 6/17/92
Security: Everyone

AI: Some Working Definitions Of The Terminology (Part 2 of 2)

===

Article Created: 19 August 1988
Article Last Reviewed: 9 June 1992
Article Last Updated:

This article continued from "Working Definitions Of Some "AI" Terminology (Part 1 of 2)"

Expert Systems:

Computer systems that address subjects previously requiring a human expert on hand. An excellent example of this is AppleCat diagnostics from Apple. This software can be used to isolate a Macintosh hardware problem that previously required a service person (expert).

Flavors:

The Object Orientated Programming implementation used by the Explorer.

Garbage Collection:

LISP programs inherently involve frequent memory allocation and deallocation. Garbage collection is the process of reclaiming memory that was allocated, but is no longer required by the application. While LISP machines address garbage collection through hardware, any conventional machine must duplicate this effort in software, typically forcing frequent delays while waiting for the machine to return to the user (think of printing with and without a spooler). This is one of the reasons for the claim that LISP is a slow language or could never be used for real-time processing.

LISP Language vs. LISP environment vs. LISP machine:

The LISP language is the implementation of LISP, the syntax of the language.

Most of the industry recognizes that Common LISP is becoming the standard language for LISP. When we talk of a LISP environment, we are referring to the programming features offered to assist one in programming in LISP, such as the LISP listener (interpreter), debugger, inspector, object-oriented programming implementation, compiler, tag implementation, and garbage collection utilities to name a few. These are available (in a varying degree) for whatever LISP you are using. When we mention a LISP machine, we are referring to a processor/computer built specifically for running the LISP language, with hardware features optimized to run LISP.

Knowledge Engineering:

Using AI programming techniques (methodologies) -- such as knowledge representation, search, or reasoning strategies -- to solve applications requiring symbolic processing.

Knowledge Engineer :

This title gets a lot of attention these days, and often a considerably higher salary for the software developer/ programmer/ analyst familiar with AI programming techniques, languages and shells and their proper implementation.

LISP Machine:

A computer specifically designed to run the LISP language. Includes hardware support for critical "LISP" functions, such as garbage collection and data type processing.

Object Orientated Programming:

A method of programming with abstraction that allows a more natural representation of real world items (or objects). An object can be defined as a "class" that can have associated or inherited procedures, arguments, or component-objects.

An example of this is to define a new "class" of object called "COMPUTER". Each COMPUTER can be defined to have associated with it such items as a keyboard, monitor and CPU. New classes can then be defined based on this class -- a "MACINTOSH" can be a type of "COMPUTER", and as such we can assume that it has a keyboard and monitor.

When we create this new class MACINTOSH, we can also define a default CPU type of 68000. We could go on and define a procedure for the MACINTOSH called HOW-TO-RUN-APPLICATION which has a value of CLICK-ICON, (whereas the procedure for the original COMPUTER class may have an associated HOW-TO-RUN-APPLICATION value of ENTER-ARCHAIC-COMMAND). The idea is that with Object Orientated Programming, we are able to define objects in a way that more closely associates how we naturally think of the objects. Also, once we know an object is defined, and what values it has, we don't have to worry about HOW it was defined -- as long as it works. This leads to a "Black Box" approach, where we do not concern ourselves with what is in the Black Box as long as we know it is available and works.

Rapid Prototyping:

The process of using a short "develop-test-modify" cycle, with frequent feedback, as a way to help define the final system solution. The idea is to work on the conceptual level to define a proper solution and worry about the final implementation details later, as opposed to a programmer spending significant time in development, only to arrive at a working solution that does not address the original problem correctly.

Tagged Architecture:

Hardware support for implementing Data Type Processing on LISP Machines. This allows processing "tags" in parallel with other instructions. Any conventional hardware running LISP must duplicate this effort in software.

Tech Info Library Article Number:3203