

# VisiSoft the Visual Software Environment

May 3, 2004

## THE TECHNOLOGY "BREAKTHROUGHS"

*VisiSoft* - A leap forward in technology for building systems.

Instead of describing a system in a programming language, designers first produce a graphical description of the architecture of the system and its modules. This is done using engineering drawings, symbolic design descriptions, and a graphical CAD environment. To accomplish this requires a major change in thinking about automating the system design process. It is based upon *two new paradigms* that render current computer programming approaches obsolete. These are described below.

**Separation of ARCHITECTURE from LANGUAGE.** The first major paradigm change required to automate the module development process is separating architectural design from detailed implementation. Figure 1 depicts the results of this separation in *VisiSoft*.

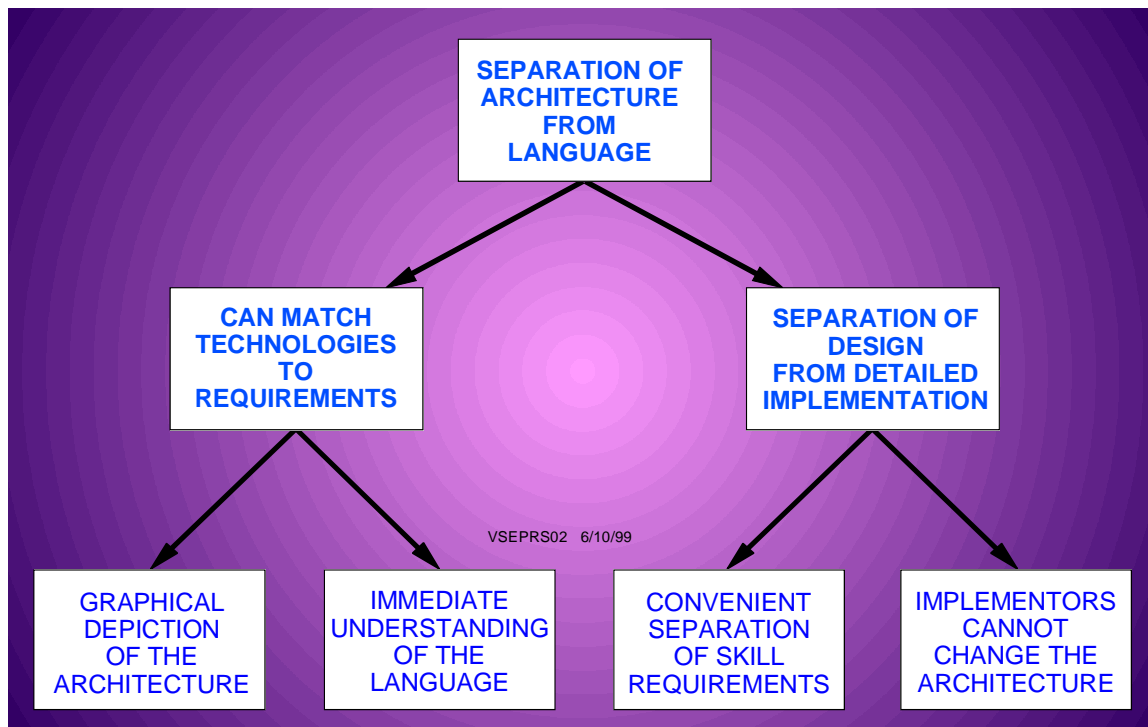
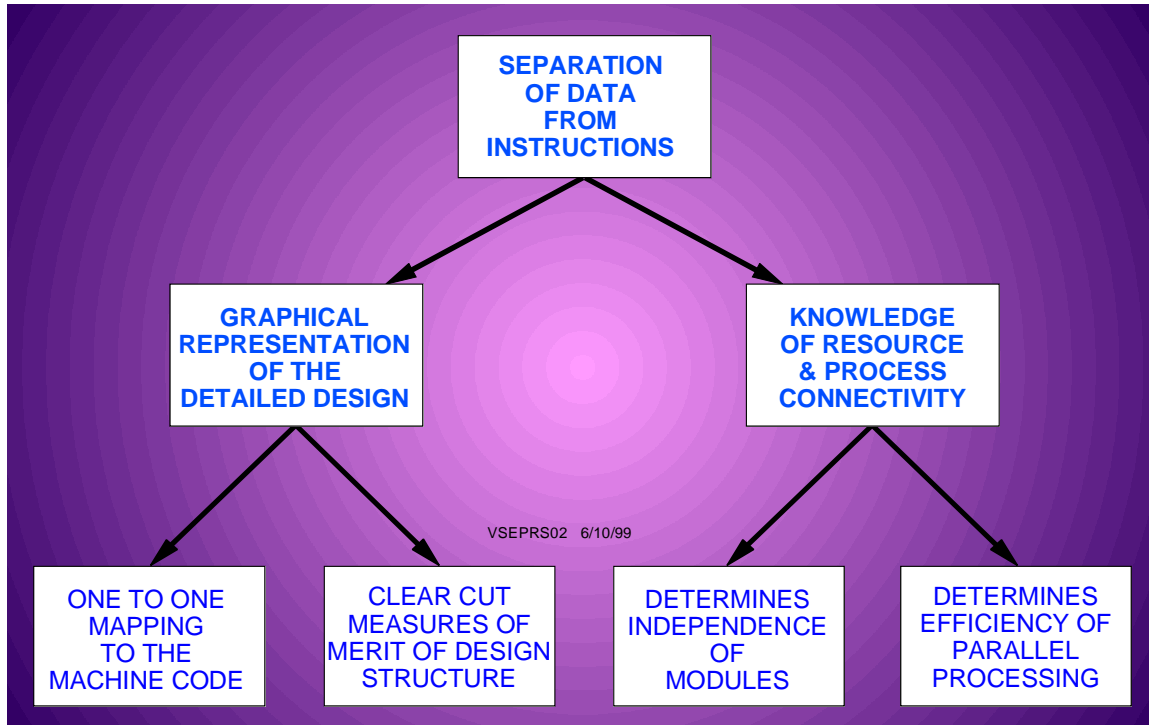


Figure 1. Separation of Language from Architecture.

**Separation of DATA from INSTRUCTIONS.** The second major paradigm change required to automate the system development process is separation of data from instructions in the Language environment. Figure 2 depicts the results of this separation in *VisiSoft*.



**Figure 2. Separation of Data from Instructions..**

To separate architecture from language requires that the graphical depiction of the architecture provide a transition to the language without abstractions that cloud the picture. In addition, graphical representation of the architecture must accurately represent the design, to the primitive element (language) level, without ambiguity. This one-to-one mapping from graphics to language can only be accomplished by separating data from instructions - a unique property of *VisiSoft*.

Upon reflection, this is precisely the paradigm at the hardware level, particularly when dealing with parallel machines. VSI's language translators put data together with instructions to get through the C compiler, only to have them pulled apart again coming out of an assembler. **Modern machines want to see separate instruction sets and data sets.**

Decomposition of module architectures is best performed using graphics.

The designer can break a system into hierarchical modules that can easily be reorganized and modified as the design progresses and more details are understood. By reviewing the engineering drawings and visually checking to see if design rules are followed, one can easily determine if the modules are independent, and if the design is flexible for future change, as well as easy to understand.

In *VisiSoft*, the architectural design decomposes a module into a distributed set of *data structures*, and *transformations* on those data structures. This is done at the primitive element level. The primitive elements are Resources (attributes/data structures) and Processes (rules/transformations). All this is done without reference to a language.

When the architectural design is complete, then implementation can begin with definition of the detailed information required to describe the system. The state of a module is described by its *attributes*. Transformations that change the state of a module are described by *rules* that depend on its inputs and current state. These are described in two separately tailored English-like languages that anyone can understand.

The architecture determines what Processes (rules) have access to what Resources (attributes). This *connectivity* of data and instructions is stored by the system. It determines the *independence* of modules, and thus the modularity, understandability, and reusability of the design.

Systems are decomposed into independent modules for parallel processing just by following simple design rules. Since a map of the connectivity is kept by the *VisiSoft* system, efficient assignment of parallel processors to *VisiSoft* Processes can be implemented automatically, without concern by the user.

In *VisiSoft*, implementors can be "locked out" of the Architecture environment. Since there are no language statements that affect the architecture, they can only implement changes to existing Resources (attribute structures) or Processes (rules). They cannot add, change, or delete resources and processes, or the connections between them.

By separating data from instructions, we obtain a visual picture of the organization of a piece of software at its most primitive level.

We also see the map of connectivity between distributed data structures (Resources) and transformations (Processes).

Control flow is localized to the Process. Otherwise, one easily *loses control* of the flow.

When building parallel systems, e.g., in discrete event simulation using *VSI's General Simulation System, (GSS)*, one need only know *which* Processes are scheduled *when* (automatically taken care of by the *GSS* scheduler), and which ones can run concurrently, i.e., are independent (automatically determined by the Process-Resource connectivity map).

Also, in *VisiSoft* there is no such thing as global data - a real bottleneck when trying to use parallel processors.

Finally, and most important from a project management standpoint, one can quickly scan the engineering drawings, and determine if the design rules that guarantee module independence have been followed by the system architects. These design checks are also easily automated.

## VISUAL SOFTWARE INTERNATIONAL

309 Morris Avenue  
Spring Lake, New Jersey 07762

 (732) 449-6800     (732) 449-0897  
 [VSI@Visisoft.US](mailto:VSI@Visisoft.US)     [www.VisiSoft.US](http://www.VisiSoft.US)