Prototyping Distributed Environments with 001

Margaret Hamilton
Ron Hackler

Summary
Conventional software based system prototyping and development techniques rely upon an *"after the fact"* approach. They concentrate on fixing wrong things rather than on doing things right in the first place. Because of the lack of rigor, and therefore control, inherent in the use of these methods, the integrity of a system is reduced, there is a compromise in functionality, deadlines are missed and significant dollars are wasted.

There are many new tools intended to address the desires of the software based systems marketplace. They usually fit into the category of CASE. Most CASE products, however, are based on the conventional life cycle; they automate manual processes of the conventional development process when many of these processes need no longer be necessary. This is instead of solving the real problem, which is to define a system correctly in the first place. There is a serious need for an integrated set of techniques and tools which provides the ability to design systems right; and, then, automatically prototype or develop them. 001 was created to fill this need.

001 is a new paradigm for system engineering and software development. It is a hierarchical functional and object-oriented network modeling technique based upon a unique concept of control. It is used to specify, prototype and develop systems of arbitrary complexity and size (for both data base driven and real time distributed environments) using an approach called *"development before the fact"*. With this approach, a system has properties of *built-in* quality and *built-in* productivity that support its own development. Unlike with conventional approaches, 001 uses a preventative instead of a curative approach. As the user uses 001 to model the behavior and structure of his application system, he inherently models the "life cycle" of his system.

The 001 tool suite (see Figure 1) is an integrated "automated factory" that automates the system development process from specification to the generation of complete production ready code (e.g., Ada or C) with full automated documentation. Each system developed with 001 is defined with built-in properties which include: integration of all aspects of its definition (control flow, data flow, state transition, data structure, etc., are all inherently defined and integrated by the use of the same language); reliability (75%-90% of all errors are inherently eliminated); reuse (all objects are reusable); ability to capitalize on parallel environments; implementation independence (source code can be generated for any environment--including distributed and parallel--from a given specification); maintainability (effects of all changes are traceable and understandable) and high productivity (results have shown significant leverage). Using 001, developers and maintainers respond to any change in the system by making changes to the specification, not to the code; production-ready code is regenerated automatically by the tool. The 001 tool suite has been defined and automatically generated by itself.

Distributed control systems can be defined using stylized models defined with 001's specification language, 001 AXES. These models are based on the language mechanisms of 001 AXES which support aspects of an application that relate to the

0-8186-3040-X/92 $3.00 © 1992 IEEE

distribution of a functional architecture onto a resource architecture. These aspects of the target system include its environment, resources, interrupts, information organization, communication strategies, and the functionality that is to be distributed across the control system. Each aspect has a corresponding graphical representation which is directly associated with a combination of formal language mechanisms. The result is a reduction in complexity in the prototyping and development process. The combined set of graphical representations forms a language which provides a quick and friendly building block kit. This language can be used by a user to define models having distributed properties without having to know the details corresponding to the graphical representations. Because of the formal language details associated with the graphic representations, a system may be automatically implemented by the resource allocation generation tool of the 001 tool suite. This approach provides the user a direct path from their graphical representation of the functional architecture (as a distributed system) to a fully executable distributed implementation.

The set of graphical representations is used as a modeling tool to define a hierarchy of real time distributed controllers where the parent controller is in charge of its children as controllers. A real time distributed controller coordinates communications, interrupts and resources between it and other controllers. A controller performs a portion of a distributed functional system which is defined to respond to some environment. The behavior and structure of a controller is completely defined using 001 real time, asynchronous mechanisms.

The Xecutor component of 001 is used to support rapid prototyping and development for a distributed environment. The Xecutor, a 001 "machine", is a combination of a meta operating system and simulator that understands 001 semantics. It provides a realtime, asynchronous, event-driven execution environment with multiple lines of control over concurrently executing functions (and is thus multi-threaded). The 001 Xecutor supports the real time execution of each distributed controller as an operating environment and automates the communications strategy between controllers. A 001 Xecutor is associated with each controller. We will discuss an approach using the graphical language and distributed Xecutors for defining and prototyping a system of distributed controllers.

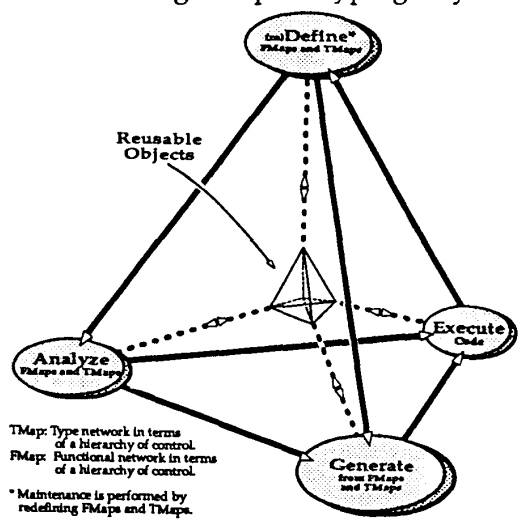

Figure 1: The 001 Tool Suite's *Development Before the Fact* Integrated Life Cycle.