Software: Where We've Been And Where We're Going

Has software design finally come of age? Mechanical, electrical, chemical, industrial, and other engineering design disciplines have been in place for centuries and have experienced the associated growth and cross-pollination of shared concepts, tools, trial and error, etc. However, software engineering is still relatively new.

The discipline of software design has only been matured for a few decades. It wasn't until the 1960s that the first software products hit the marketplace. Standards such as American Standard Code for Information Interchange and concepts such as object-oriented code didn't appear until the 1960s. Our dominant programming language C++ didn't emerge until the 1980s. More recently, we have seen design tools such as CASE [Computer-Aided Software Engineering] and fourth generation programming languages.

Our discipline of software engineering has really experienced phenomenal growth right before our eyes. A sign that software design has really arrived is indicated by the hardware changes that are being driven by the complexity of software designs. Our phenomenal growth is also accompanied by real challenges. The increasing complexity of our software also necessitates that our industry stay focused on process improvement. As you read some of this month's articles, take yourself back 10 or 20 years, and revel at how far we have come, but also envision where we need to go.

Kevin Stamey
Oklahoma City Air Logistics Center, Co-Sponsor

Design Focuses on the *How*

This month we cover another critical phase in the software system life cycle – design. Industry has been successful over time with emphasizing and providing many methods and tools for improving design. Software engineers know their hands will be slapped if they rush from requirements to programming code without taking the time to design. The design phase can be looked at as a problem-solving process. Another way to think of design is to focus on the *how*. Requirements are focused on *what* problem will be solved. The design focuses on *how* the problem will be solved.

Our theme section begins with *Selecting Architecture Products for a Systems Development Program* by Michael S. Russell. This author describes a repeatable process that emphasizes architecture as the source of information that decision makers can turn to throughout the systems engineering process. In *Dependency Models to Manage Software Architecture*, Neeraj Sangal and Frank Waldman relate a new approach using inter-module dependencies to specify and manage system architecture. Next, in *UML Design and Auto-Generated Code: Issues and Practical Solutions* by Ilya Lipkin and Dr. A. Kris Huber, lessons learned from designing with the Unified Modeling Language (UML) are presented from a real-time control system perspective. Lastly, Dr. Hans-Peter Hoffmann describes how systems engineers can capture requirements and specify architecture in *UML 2.0-Based Systems Engineering Using a Model-Driven Development Approach*.

In this month's supporting articles, Arlene F. Minkiewicz and Jeffrey Voas each offer additional information on software security, another important factor to consider during the design phase. I hope this month's issue provides helpful information as new and improved methods continue to evolve design processes.

Tracy L. Stauder
Publisher