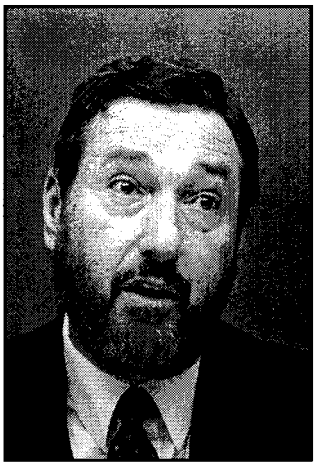
THE LEGACY OF HARLAN D. MILLS



This colloquium is organized in honor of Dr. Harlan D. Mills (1919-1996), and is intended to celebrate his legacy in the theory and practice of software engineering.

Harlan Mills left a huge vacuum when he died in 1996. Those of us who were touched in any way by this gentle man's intellect, humor and philosophy, are painfully aware of that vacuum. Our industry has lost one of its iconic figures.

To create a fitting memorial to Harlan, we turn our attention (as he would wish) to some of the themes that occupied him during his life. Even though it is a commemorative event, the colloquium is not just a reminiscence on one man's work. Rather, it is focused on the extension of that work to today's state of the art. It will underscore the relevance of his ideas to modern challenges in research and practice.

Some of the most recurrently noted themes in the work of Harlan Mills are the following:

1. Intellectual Control: Harlan recognized that much of the problem of software development stems from lack of intellectual control over the software development process. We cannot achieve quality in software unless we learn to develop software under perfect intellectual control. While conventional wisdom provides that we learn to develop software on small examples so as to graduate to larger examples, the premise of intellectual control provides that we learn on small examples in order . . . not to do large examples.
2. Relentless Pursuit of Simplicity: Harlan recognized the importance of simplicity as the key feature of a design; many of his prescriptions are geared towards encouraging simple designs. Software design can in fact be viewed as the identification of simple (but not necessarily intuitive) principles behind seemingly complex behaviors.
3. Emphasis on Quality: In an industry best known for delivering faulty products, Harlan advocated a set of prescriptions dedicated to the design and implementation of products that are provably free of defect. The ways of writing correct programs and knowing are at the heart of his approach. Practice shows that one can achieve quality and reduce overall lifecycle costs at the same time; what seems to happen with quality focused processes is that developers spend more effort on design and much less on testing and certification.
4. Multiplicity of Means: Perhaps the most important success factor in Harlan's approach to software development is the deployment of a multiplicity of means. This is dictated by the multi-faceted nature of software products and software processes. The Mills multiplicity of means is applied across a wide range of issues, including: combining technical with managerial standards in developing software products; combining static and dynamic methods in achieving product quality; combining scientific principles with engineering

methods to define software processes; combining formal semantics and understandability in dealing with program correctness.

The premises laid out above have been applied to the elemental activities of software development to produce great gains in product quality and process productivity.

- Specification: Specifications are represented by black box structures, which map stimulus histories to outputs [3].
- Design: Design is carried out by mapping black boxes into state boxes then clear boxes, which are increasingly detailed design representations [3].
- Verification: Methods of correctness verification are deployed at each step of the transformation process. Also, using Mills' Clean Room methods [2, 11, verification-based inspections substitute for traditional unit testing.
- Testing: Under the Clean Room technique, testing is used exclusively for certification and reliability estimation. Functional (black box) testing is applied, where input data is generated randomly to simulate the statistical usage patterns of the software product.
- Re-engineering: With such contributions as structure theorem transformations and design abstraction and documentation, Harlan provided the basis for a systematic approach to software reengineering.
- Team Organizations: Harlan's concepts of Chief Programmer Teams and Clean Room organizations offered original solutions to the problem of organizing communication channels among members of a programming team. His team organizations are characterized by low communication overhead, narrow communication bandwidth, and clearly distributed responsibilities and prerogatives.
- Education: To illustrate that his ideas on software engineering education are effective even though they are counter-intuitive, Harlan used two recurrent examples: learning how to swim and learning how to type. These two endeavors are examples of how the most natural way to learn is a very poor option.
- Professional Standards: Harlan attempted to define professional standards for the software industry, to parallel the rigor and the maturity of more traditional engineering disciplines.

REFERENCES

[1] R.C. Linger. "Cleanroom software engineering for zero-defect software," *Proceedings, 15th Hawaii International Conference on Software Engineering,* Baltimore, MD, May 1993.

[2] R.C. Linger and P.A. Hausler. "Cleanroom software engineering," *Proceedings, 25th Hawaii International Conference on System Sciences,* Kauai, Hawaii, January 1992.

[3] H.D. Mills, R.C. Linger, and A.R. Hevner. *Principles of Information Systems Analysis and Design,* Academic Press, 1985.