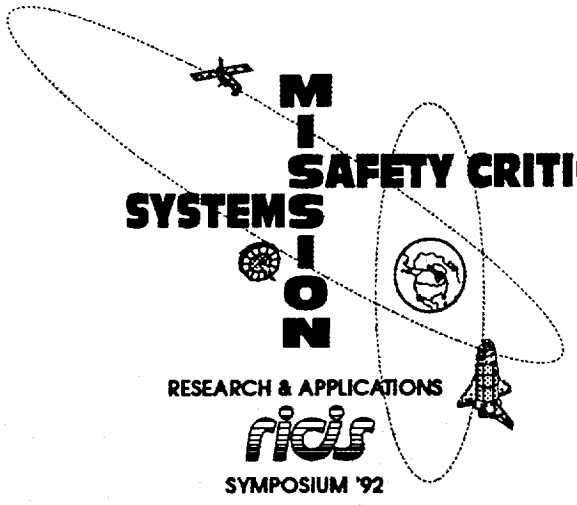


# FORMAL METHODS AT IBM FSC

*David Hamilton*

## ABSTRACT

IBM has a long history in the application of formal methods to software development and verification. There have been many successes in the development of methods, tools, and training to support formal methods. And formal methods have been very successful on several projects. However, the use of formal methods has not been as widespread as hoped. This presentation summarizes several approaches that have been taken to encourage more widespread use of formal methods, and discusses the results so far. The basic problem is one of technology transfer, which is a very difficult problem. It is even more difficult for formal methods. General problems of technology transfer, especially the transfer of formal methods technology, are also discussed. Finally, some prospects for the future are mentioned.

## BIOGRAPHY

David Hamilton is an advisory programmer in the IBM Federal Systems Company. He joined IBM in 1982 in a Space Shuttle Flight Software Testing Group and worked in several areas of software testing and test tool development. In 1987, he began investigating tools that supported formal verification, and began studying the reasons why formal methods have not gained more widespread use. More recently, he has been involved in the development of a formal specification language for expert systems and supporting a NASA feasibility study of formal methods. His general interests are software verification and validation, and the use of AI in software engineering.

# Formal Methods Technology Transfer

## Some Lessons Learned

David Hamilton
IBM Federal Sector Corporation

RICIS Symposium '92

Oct/92

## Contents





## Introduction and Purpose

- To cover
  1. Some IBM involvement in Formal Methods (FM) projects
  2. A perspective on difficulties of technology transfer (beyond a single project)
- Purpose is **not** to
  - sell the "IBM approach"
  - argue against feasibility of FM
- Purpose is to
  - learn from other FM technology transfer projects
  - suggest some possible future directions



## Harlan Mills and SEW

- Mills led massive software engineering education program
  - Software Engineering Workshop was cornerstone
    - 2 week course
    - Taught to all programmers
    - Required to pass final exam
- SEW centered on mathematically-based verification
  - Functional instead of axiomatic
    - model oriented instead of property oriented
    - designed to scale up (stepwise refinement)
    - easier for programmers to understand
  - 2 pieces
    1. Deriving program functions
       - Trace tables (basically manual symbolic execution)
       - Recurson instead of loop invariants
    2. Module-oriented
       - abstract data types
       - constraints/closure on state data (abstract state machine)

## Harlan Mills and SEW ...

- SEW designed to be practical
  - relatively informal
  - scaled up via abstraction/refinement
  - lots of examples and exercises
  - final test : pass/fail
- Advocated for all programming, not just critical parts
- no support beyond education
  - no tools
  - no consulting
- General reaction was that it was impractical
  - too tedious
  - seemed only for toy problems
- Did not gain widespread use



## Cleanroom

- Named after silicon chip manufacturing environment
- Built on SEW foundation, adding
  - Continuous inspections (SEW style verification)
  - Statisical testing (MTTF prediction)
- Advertised through case studies, not classes
  - Demonstration projects using highly skilled developers
  - To demonstrate benefits
  - To show it can be done, it is practical
- Demonstrations projects were success stories



## Cleanroom ...

- Showcase project was COBOL/SF
  - Transforms unstructured COBOL into structured COBOL
  - 52,000 SLOCS developed using Cleanroom
  - Results
    - 740 SLOCS / labor month
    - 3.4 errors / KSLOC (before first execution) (70 avg incl. UT)
    - no error ever found during operational use
- Advocacy of Cleanroom continues
  - Widespread use not yet attained
  - But there is a lot of interest in Cleanroom



## SEDL

- Intended to support SEW/Cleanroom verification concepts
- Built as an extension to Ada
- SEDL compiler generates Ada
- Supports design execution
  - though SEDL generated code my be inefficient
- Includes
  - Abstract data types (set, list, map, etc.)
  - User defined data models
    - model vs. representation
    - constraints
  - Supports mathematical notation
    - {X in CHARACTER : x /= 'Q'}
    - exists X in S : P(X) and exists Y in T : P(Y)
    - P > 1 and not (exists Q in 2..P-1 : P rem Q = 0)
- Use of SEDL is not widespread

## Stepwise Verification

- Goal: increase use of more formal verification
  - Build on SEW, Cleanroom foundation
  - Investigate tools that support SEW
  - Help projects get started
- Step 1: Understand why SEW approach not widely used
  - Survey of developers
  - Interviewed to those who participated in Cleanroom pilots
  - Results
    - Generally inconclusive, no primary reason(s) found
    - Some themes were:
      1. Lack of experience
      2. Lack of support
      3. SEW reputation



## Stepwise Verification ...

- Step 2: Contact specific projects
  - Demo simple editing tools (support specs)
  - Demo on actual code from the project
  - Discuss methodology
  - Motivate use
  - Offer follow-up consulting
- Results
  - Very positive results on one tool (SEED)
  - Negative results on the methodology
    - redundant work
    - incompatible with current methodology
    - impractical



## CICS

- CICS is '60s vintage IBM product (assembler) transferred to Hursley, U.K.
- Hursley began big program towards more formal S/W Eng.
- Key approach was formal specs using Z
- Worked hand-in-hand with PRG at Oxford
- Began with selected modules and later expanded use
- Results
  - Some initial "culture shock" for both parties
  - Now some 50 people work directly with Z specs
  - Very positive qualitative results (people like it)
  - Limited quantitative data indicates
    - earlier error removal
    - fewer inserted errors
    - slight cost reduction (9%)
- Use of Z continues at Hursley, but very few other places



## TOP (Verification of ESs)

- Some concern existed about verifiability of Expert Systems (ESs)
- Study of the problem pointed to one area: Specification
  - Poor low level languages
  - Almost no design
- Led to development of an ES design language
  - Based on work done at USC ISI (LOOM and CLASP)
    - higher level language (term subsumption + OOP)
  - we added annotations
    - Pre and post conditions
    - Global constraints
    - etc.
  - Supported by a compiler (a la SEDL)
  - Supports modularity (a la Ada)
  - Supports annotations
- Cleanroom has also been extended to expert systems
- Neither approach has gained widespread use

## Other Projects and Approaches

- Application above the code level
  - Development of a "Box Structures" design language
  - Development of a "Box Structures" approach to requirements
  - Results
    - SA/SD approach to design most popular new approach
    - Requirements still written in English
- Emphasis on SEW concepts
  - Concepts generally well accepted
  - Loss of rigor reduces mathematical basis



## Note on Quality Emphasis

- Software quality has extreme emphasis
  - Great emphasis on process improvement
  - Serious attention given to quality goals and measurement
  - Quality motivation programs
    - awards and recognition
    - Manned Flight Awareness program
- There is willingness to work hard and invest for quality
- The question is not what or how much but **how**
  - FM is generally perceived as not helping



## Summary

- Goal was to increase the use of formal mathematical approaches to software development (beyond a single project)
  1. First through education
  2. Then through demonstration projects
  3. Then through tool support
  4. Then by making methods more practical
  5. Finally through direct support (consulting)
- There have been successes
  - not nearly as widespread as desired
- This story is **not** unique to FM
  - The problem is with technology transfer, not with technology



## Conclusions

- Conclusion: Technology Transfer is very hard, even with
  - extensive education
  - tools support
  - demonstrated successes
- Possible future directions
  - More consulting ("hand holding") (product champions)
  - Use only a core group (FM may just not be for everybody)
  - Require use of FM (selected groups)
  - Success story close to home
    - technology transfer diminishes rapidly as a function of distance
    - long term committment is required (success story requires continued follow-up)
  - Different formal method(s)
  - Different tools (e.g., theorem prover)