11 0065

N95- 16480

Epistemology, Software Engineering, and Formal Methods

Abstract of Presentation

C. Michael Holloway

One of the most basic questions anyone can ask is, "*How do I know that what I think I know is true?*" The study of this question is called **epistemology**. Traditionally, epistemology has been considered to be of legitimate interest only to philosophers, theologians, and three-year-old-children, who respond to every statement by asking, "*Why?*" Software engineers need to be interested in the subject, however, because a lack of sufficient understanding of epistemology contributes to many of the current problems in the field.

Epistemology is a complex subject, one to which many philosophers and theologians have devoted their entire careers. The discussion here is necessarily brief and incomplete; however, it should be sufficient to demonstrate the critical importance of the subject to software engineering.

To the fundamental question of how do we know what is true, there are three basic answers: authority, reason, and experience. An epistemology based on *authority* states that truth is given to us by someone more knowledgeable than ourselves. The two primary variations of authority-based epistemologies are *omniscient authority* (the authority is God), and *human authority* (the authority is a human expert).

An epistemology based on *reason* claims that what is true is that which can be proven using the rules of deductive logic. Finally, an epistemology based on *experience* claims that what is true is that which can be encountered through one or more of the senses.

Several different variations of experience-based epistemologies exist. The two variations relevant to this discussion are *anecdotal experience* and *empirical evidence.* The first states that truth for any particular individual or group of individuals is that which the individual, or group, personally experiences. The second states that truth is that which can be verified through carefully controlled experiments.

The relative strengths of these epistemological approaches are as follows. Omniscient authority provides absolute truth; if there is a God and He has spoken on something, then what He says must, by definition, be true. Reason yields conditional absolute truth; if the premises on which a valid deductive argument are known to be true, then the conclusion of the argument must also be true.

Empirical evidence provides probable truth; if controlled experiments are designed properly and repeated enough times, then it is highly probable that the results accurately describe reality. Anecdotal experience yields possible truth; if something happened for one person, it is possible it might happen to others also. Finally, human authority provides opinion.

On which of these approaches to epistemology is software engineering mostly based?

The software engineering literature is filled with pronouncements about how software should be developed (e.g., "*Object-oriented development is the best way to obtain reusable software*"). Rarely, if ever, are these pronouncements augmented with either logical or experimental evidence. Thus, one is forced to conclude that much of software engineering is based on a combination of anecdotal experience and human authority. That is, we know that a particular technique is good because John Doe, who is an expert in the field, says that it is good (human authority); John Doe knows that it is good because it worked for him (anecdotal experience). This is a weak epistemological foundation on which to base an entire discipline.

This current state should not be surprising; the development of software engineering is following the same pattern as the development of many other disciplines. Civil engineering, chemical engineering, aeronautical engineering, and others all had periods in which they relied almost exclusively on anecdotal experience and the subsequent authority of the "experts". Often, it took major disasters before practitioners in such fields began to investigate fully the foundations on which their field was based.

To date, although there have been many, many software problems, there have been no major disasters that have been directly attributed to software. However, unless a sound epistemological foundation is established for software engineering, disasters will come one day. To avoid this, research is needed to develop valid approaches to answering questions about both software products (e.g., are these requirements consistent?) and software processes (e.g., is method A better than method B?).

The Assessment Technology Branch (ATB), which is part of the Information and Electromagnetic Technology Division, Research and Technology Group, is currently investigating empirical methods to answer process-type questions and logical methods to answer product-type questions. The remainder of the presentation discusses the second of these two avenues of research.

A team led by Ricky W. Butler has been studying the discipline of *formal methods* for over 6 years. Other civil-servants on the team are Jim L. Caldwell, Victor A. Carreño, C. Michael Holloway, and Paul S. Miner. Vigyan, Inc., Stanford Research Institute International (SRI), Odyssey Research Associates (ORA), and Computational Logic, Incorporated (CLI) conduct research under contract.

Formal methods is[1] the applied mathematics of computer systems engineering[2]. Formal methods aims to be to software engineering what fluid dynamics is to aeronautical engineering and what classical mechanics is to civil engineering. The mathematics of formal methods includes predicate calculus (first order logic), recursive function theory, lambda calculus, programming language semantics, and discrete mathematics (e.g., number theory, abstract algebra). To this mathematical base, formal methods adds notions from programming languages such as data types, module structure, and generics.

There are many different types of formal methods with various degrees of rigor. The following is a useful classification of the possible degrees of rigor in the application of formal methods:

- Level 0: No use of formal methods
- Level 1: Formal specification (using mathematical logic or a specification language with formal semantics) of all or part of a system
- Level 2: Formal specification at two or more levels of abstraction and paper-and-pencil proofs that the detailed specification satisfies the abstract one
- Level 3: Like level 2, except paper-and-pencil proofs are replaced by formal proofs checked by a semi-automatic theorem prover.

Presently, a complete (level 3) verification of a large, complex system is impractical; however, application of formal methods to critical portions of a system is practical and useful.

The specification of a simple phone book provides a suitable simple example of many of the basic ideas and benefits of formal methods. Please see the presentation visuals that follow this abstract for this example.

Because of the promise that formal methods offers, a considerable amount of high-quality research is being conducted or sponsored by ATB. This research includes, but is not limited to, the following projects:

- Detailed design with complete level 3 verification of the Reliable Computing Platform, which is a fault-tolerant computing base able to recover from both permanent and transient faults
- Design with level 2/3 verification of a transient fault-tolerant clock synchronization circuit; this circuit has also been fabricated, but the layout was done by hand without formal verification
- In cooperation with SRI and Rockwell-Collins, level 3 specification and verification of the microcode of the AAMP5 microprocessor
- In cooperation with ORA and Union Switch and Signal, level 3 specification and verification of a next-generation railroad control system
- Under contract, ORA is working with Honeywell on level 3 specification and verification of aircraft navigation functions
- Under contract, Vigyan and SRI are working with Loral, Johnson Space Center, and the Jet Propulsion Laboratory on level 3 specification and verification of some Space Shuttle functions
- Under contract, SRI is working with Allied-Signal on level 3 specification and verification of important algorithms for fault-tolerance

In addition to these, and other, projects, the branch conducts periodic workshops on formal methods. Previous ones were held in 1990 and 1992; the next one is planned for 1995. Also, an extensive collection of information on the research is available through the World Wide Web at the following Universal Resource Locator:

http://shemesh.larc.nasa.gov/fm-top.html

Interested individuals are encouraged to explore this collection.

A lot of ground has been covered in this presentation, but the most important point is simple:

Epistemology: It's important, learn about it
Software Engineering: It's immature, work on it
Formal Methods: It's promising, look for it

1. Just like mathematics, formal methods should be treated as a singular, not plural, noun.

2. The ideas apply equally well to both software and complex hardware devices.

Epistemology Software Engineering and Formal Methods

C. Michael Holloway

Assessment Technology Branch
Information & Electromagnetic Technology Division
Research & Technology Group
Langley Research Center
National Aeronautics and Space Administration
United States Government

The Role of Computers in LaRC R & D (June 15-16, 1994)

Introduction

- One of the most basic questions anyone can ask is

  "How do I know that what I think I know is true?"

- The study of this question is called ***epistemology***
- Traditionally, epistemology has been considered to be of legitimate interest only to philosophers, theologians, and three-year-old children
- At least one other group should be very interested in epistemology -- software engineers -- because lack of understanding in this area plagues the field

The Basics of Epistemology

- There are three basic answers to the question of how do we know what is true
  - -- *Authority:* truth is given to us by a knowledgeable person
  - -- *Reason:* truth is what can be proven using the rules of deductive logic
  - -- *Experience:* truth is what can be encountered through one or more of the senses
    - * *Anecdotal experience:* truth is what an individual or a group of individuals experiences personally
    - * *Empirical evidence:* truth is what can be verified through carefully controlled experiments

Examples of Truth by Authority

- The Ten Commandments

  (omniscient authority)

- 1-year-old, pointing to the family cat: "Whatsthat?"

  father: "Kitty"

  (human authority)

Examples of Truth by Reason

- If that creature is a tove, then it is slithy
  That creature is a tove
  Therefore, that creature is slithy
- If the airplane was built by Boeing, then it is a jet
  The airplane is not a jet
  Therefore, the airplane was not built by Boeing
- $X + Y = 7$
  $3Y - 2X = 1$
  Therefore, $X = 4$ and $Y = 3$

Examples of Truth by Anecdotal Experience

- Smoking doesn't shorten your life because my father smoked all his life and lived to be 95.
- Whenever I have the hiccups, I hold my breath and count to 10 and they go away. Therefore, holding your breath and counting to 10 cures the hiccups.
- We used method M and had 40% fewer bugs in testing. You should use method M, too.

Examples of Truth by Empirical Evidence

- The dive-recovery flap for the P38 in World War II developed through tests in Langley's 8-Foot High Speed Tunnel
- 5,000 patients were given drug X. 5,000 patients were given no drugs at all. 4,998 of the patients given drug X got better within 1 week. 3 of the patients given no drugs at all got better within 1 week. Drug X helps.

Relative Strengths

· Omniscient Authority:	*absolute truth*
· Reason:	*conditional absolute truth*
· Empirical Evidence:	*probable truth*
· Anecdotal Experience:	*possible truth*
· Human Authority:	*opinion*

How Does This Apply to Software Engineering?

- The software engineering community is full of claims

  "The best way to develop reusable software is to use object-oriented design."

  "Programmers should never be allowed to test their own code."

  "Getting control of the software process is the key – SEI's CMM is the way to do this."

  "We need more standards!"

  "Much progress has been made in the last few years in improving the way we develop software."

  "GOTO's are harmful."

  "CASE tools are the best way to improve software productivity."

- Many people accept these, or other similar, claims as being true

The Fundamental Question

What is the

epistemological foundation

for accepting

these claims?

The Answer

- Logically sound arguments are rarely given
- Virtually no empirical evidence is cited
- Instead, software engineering is based almost entirely on a combination of human authority and anecdotal experience
  - -- *We know that technique C is good because Jane Doe, who is a recognized authority in the field, says that it is good (human authority)*
  - -- *Jane Doe knows that it is good because she used it on a project once and got good results (anecdotal experience)*
- This is a weak epistemological foundation, one on which no legitimate claims of success can be based

Implications of This Epistemological Weakness

- Until we get adequate evidence, we should be very cautious in the claims we make and the standards we set
  - -- It is fine to say, *"Method M seems to have improved our productivity, so you might want to try it."*

    But it is dishonest to say, *"If you want to improve your productivity, you must use Method M."*
  - -- *"Company R used method F and found errors they don't think they would have found using their old methods,"* is fine

    "*Method F finds errors that other methods do not find,*" is dishonest
- The software engineering community should be investigating methods for obtaining strong (that is, logical or empirical) evidence

Why Has More Not Been Done?

- The development of software engineering is following the same pattern as the development of other disciplines
  - -- *Civil engineering, chemical engineering, aeronautical engineering, etc. all had periods in which they relied almost exclusively on anecdotal evidence*
  - -- *Often, it took major disasters to prompt changes*
- It is hard
- It is expensive
- It is not glamorous
- Few people care: We haven't had a *major* disaster yet

Why Must More Be Done?

- Without adequate evidence, we are easily influenced by the latest bandwagon that goes rumbling by
- Without adequate evidence, we may well "cast-in-concrete" something that ought not even be "cast-in-mud"
- Without adequate evidence, the following two statements are equally as meaningless:
  - -- *You shall use method M in developing your software*
  - -- *'Twas brillig by the slithy tove*
- Without adequate evidence, disasters are inevitable

Towards Establishing a Valid Epistemological Foundation

- Recognize the fundamental need for such a foundation
- Understand the different approaches needed for *process* and *product*
  -- Process questions (e.g., *Is method A better than method B?*) need to be answered empirically
  -- Product questions (e.g., *Are my requirements consistent?)* need to be answered by an appropriate combination of logical and empirical methods
- Refuse to accept claims based on insufficient evidence

Current Research at LaRC

- Kelly Hayhurst (IETD/ATB) is leading an effort to develop an empirical evaluation of a particular approach to IV & V

  For more information, contact Kelly

  Email: k.j.hayhurst@LaRC.NASA.GOV

  Phone: 46215
- The formal methods team led by Ricky Butler (IETD/ATB) is investigating logical methods for answering product-type questions
  -- *Other team members are Jim Caldwell, Victor Carreño, Michael Holloway, and Paul Miner*
  -- *Remainder of talk concerns this work*

Further Reading on Epistemology

- If you are interested in more information on epistemology, I recommend you start with the following two books:

  -- *Thales to Dewey, by Gordon H. Clark, 2nd edition, 1989, ISBN 0-940931-26-5*

  -- *The Philosophy of Science, by Gordon H. Clark, 2nd edition, 1987, ISBN 0-940931-18-4*

- These two books contain pointers to most of the important philosophical works throughout the ages

Singular or Plural?

- Which of the following is correct?

  Formal methods is the applied mathematics ...

  OR

  Formal methods are the applied mathematics ...

- Answer depends on the writer or speaker

- I will tend to use "formal methods" as singular

What is Formal Methods?

- Formal methods is the applied mathematics of computer systems engineering
- The mathematics of formal methods includes:

  -- *predicate calculus (1st order logic)*

  -- *recursive function theory*

  -- *lambda calculus*

  -- *programming language semantics*

  -- *discrete mathematics: number theory, abstract algebra, etc.*

What is Formal Methods?
(continued)

System Designed	*Engineering*	*Theory*
Bridge	**Civil**	**Classical Mechanics**
Airframe	**Aeronautical**	**Fluid Dynamics**
Nuclear Reactor	**Nuclear**	**Quantum Mechanics**
Digital Avionics System	**Software**	**Formal Methods**

Classical vs Computer Systems

Classical Systems	*Computer Systems*
continuous state space	discrete state space
smooth transitions	abrupt transitions
finite testing & interpolation acceptable	finite testing inadequate, interpolation unsound
mathematical modeling	prototyping & testing
build to withstand additional stress	build to specific assumptions
predictable	surprising

What Makes a Technique a Formal Method?

- Formal method = logic + programming language concepts
- Important attributes:

  -- *logic based*

  -- *programming language concepts (e.g., data types, module structure, generics)*

  -- *fully and formally specified semantics*

  -- *should be able to express what is done without saying how it is done (i.e., non-procedural)*

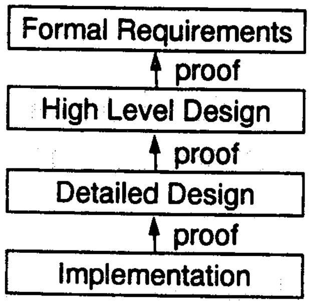
  -- *supports the building of useful tools for analysis*

Levels of Rigor of Formal Methods

- *Level 0*: No use of formal methods
- *Level 1*: Formal specification (using mathematical logic or a specification language with formal semantics) of all or part of a system
- *Level 2*: Formal specification at two or more levels of abstraction and paper-and-pencil proofs that the detailed specification satisfies the abstract one
- *Level 3*: Like level 2, except paper-and-pencil proofs are replaced by formal proofs checked by a semi-automatic theorem prover

Extent of Application

- Formal Methods is not an all-or-nothing approach
- Complete formal verification of a large complex system is impractical at this time



- Application of formal methods to critical portions of a system is practical and useful

Extent of Application (example)

- In the Reliable Computing Platform, we use formal methods to establish:

  ENOUGH_WORKING_HARDWARE

  $\supset$

  PROPER_OPERATION

- We use reliability analysis to calculate:

  Probability[ENOUGH_WORKING_HARDWARE]

- Reliability analysis relies on physical testing of devices to establish some important parameters

Level 1 Example: Phone Book
English Requirements

- The phone book shall store the phone numbers of a city

- Given a name, there shall be a way to retrieve an associated phone number

- It shall be possible to add and delete entries from the phone book

Level 1 Example: Phone Book
Choosing a Specification Approach

- How do we represent the phone book mathematically?

  1. A set of ordered pairs ⟨name, number⟩. Adding and deleting entries is by set addition and deletion.
  2. A function whose domain is all possible names and range is all phone numbers. Adding and deleting entries is by modification of function values.
  3. A function whose domain is only names currently in the phone book and range is phone numbers. Adding and deleting entries is by modification of the function domain and values. (Z style)

- We choose to use approach 2

Level 1 Example: Phone Book
Specifying the Book

- Using traditional mathematical notation, we would write:

  Let N = set of names

  P = set of phone numbers

  $book: N \rightarrow P$

- To indicate that we do not have a phone number for all possible names, but only for names of real people, we decide to use a special number: $\rho \in P$

- An empty phone book is specified as follows:

$$emptybook: N \rightarrow P$$
$$emptybook(nm) \equiv \rho$$

Level 1 Example: Phone Book
Accessing an Entry

Let N = set of names

P = set of phone numbers

$book\colon N \to P$

B = set of functions: $N \to P$

$FindPhone\colon B \times N \to P$

$FindPhone(bk, name) = bk(name)$

Note that $FindPhone$ is a higher-order function, because its first argument is a function

Level 1 Example: Phone Book
Adding/Deleting an Entry

Let N = set of names

P = set of phone numbers

$book\colon N \to P$

$\rho \in P$

B = set of functions: $N \to P$

$AddPhone\colon B \times N \times P \to B$

$$AddPhone(bk, name, num)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ num & \text{if } x = name \end{cases}$$

$DelPhone\colon B \times N \to B$

$$DelPhone(bk, name)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ \rho & \text{if } x = name \end{cases}$$

Level 1 Example: Phone Book
Complete Specification

Let N = set of names
P = set of phone numbers
$book: N \to P$
$\rho \in P$
B = set of functions: $N \to P$

$emptybook: N \to P$
$emptybook(nm) \equiv \rho$

$FindPhone: B \times N \to P$
$FindPhone(bk, name) = bk(name)$

$AddPhone: B \times N \times P \to B$
$$AddPhone(bk, name, num)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ num & \text{if } x = name \end{cases}$$

$DelPhone: B \times N \to B$
$$DelPhone(bk, name)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ \rho & \text{if } x = name \end{cases}$$

Level 2 Example: Phone Book
Putative Theorems

A putative theorem is a theorem that we know must be true if we have formulated the specification correctly.

Lemma putative 1:

$FindPhone(AddPhone(bk, name, num), name) = num$

Proof:

$FindPhone(AddPhone(bk, name, num), name) = num$

$AddPhone(bk, name, num)(name) = num$

$num = num$

Q.E.D.

Level 2 Example: Phone Book
Putative Theorems (continued)

Lemma putative 2: $\mathit{bk(name)} = \rho \supset$
$\mathit{DelPhone(AddPhone(bk,name,num),name)} = \mathit{bk}$

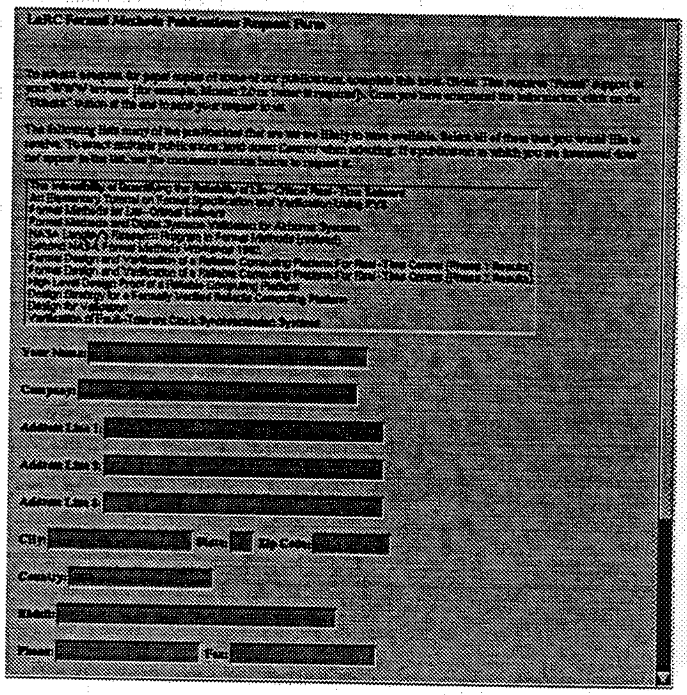
Lemma putative 3: $(\forall i{:}\mathit{name}_i \neq \mathit{name}) \wedge$

$$\begin{aligned}
&\mathit{book} = \mathit{AddPhone(bk, name, num)} \wedge \\
&\mathit{book}_1 = \mathit{AddPhone(book, name_1, num_1)} \wedge \\
&\mathit{book}_2 = \mathit{AddPhone(book_1, name_2, num_2)} \wedge \\
&\quad\vdots \qquad\qquad \vdots \\
&\mathit{book}_n = \mathit{AddPhone(book_{n-1}, name_n, num_n)} \\
\supset\ &\mathit{FindPhone(book_n, name)} = \mathit{num}
\end{aligned}$$

Formal methods can establish that a property holds even in the presence of an arbitrary number of operations; testing can never establish this.

Level 3 Example: Phone Book
PVS Specification

```
phonebook: THEORY
BEGIN

  names: TYPE
  name0: names
  ph_number: TYPE
  p_0: ph_number
  book: TYPE = [names -> ph_number]

  name: VAR names
  emptybook(name): ph_number = p_0
  bk: VAR book

  FindPhone(bk, name): ph_number = bk(name)

  num: VAR ph_number
  AddPhone(bk, name, num): book = bk WITH [name := num]

  DelPhone(bk, name): book = bk WITH [name := p_0]

  putative_1: LEMMA FindPhone(AddPhone(bk,name,num),name) = num

  putative_2: LEMMA bk(name) = p_0 IMPLIES
                    DelPhone(AddPhone(bk,name,num),name) = bk

END phonebook
```



LaRC Formal Methods Publications Request Form

[illegible]

[illegible]

[illegible]

Your Name:

Company:

Address Line 1:

Address Line 2:

Address Line 3:

City: State: Zip Code:

Country:

Email:

Phone: Fax:

Epistemology
It's Important, Learn About It

Software Engineering
It's Immature, Work On It

Formal Methods
It's Promising, Look For It

ORIGINAL PAGE IS
OF POOR QUALITY

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE October 1994	3. REPORT TYPE AND DATES COVERED Conference Publication

4. TITLE AND SUBTITLE The Role of Computers in Research and Development at Langley Research Center	5. FUNDING NUMBERS WU 505-90-53
6. AUTHOR(S) Carol D. Wieseman, Compiler	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, VA 23681-0001	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	10. SPONSORING / MONITORING AGENCY REPORT NUMBER NASA CP-10159
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 62	12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This document is a compilation of the presentations given at the workshop, "The Role of Computers in Research and Development at Langley Research Center," on June 15-16, 1994. The objectives of the workshop sponsored by the Computer Systems Technical Committee were to inform the LaRC community of the current software system and software practices being used at LaRC. To meet these objectives, there were talks presented by members of the Langley community, Naval Surface Warfare Center, Old Dominion University, and Hampton University.

The workshop was organized in 10 sessions as follows: Software Engineering; Software Engineering Standards, Methods, and CASE Tools; Solutions of Equations; Automatic Differentiation; Mosaic and the World Wide Web; Graphics and Image Processing; System Design Integration; CAE Tools; Languages; and Advanced Topics.

14. SUBJECT TERMS Software Engineering	15. NUMBER OF PAGES 604
	16. PRICE CODE A99

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Level 3 Example: Phone Book
Proof Using PVS

```
putative_1 :

  |-------
{1}   (FORALL (bk: book), (nm: names),(num: ph_number):
               FindPhone(AddPhone(bk, nm, num), nm) = num)


Rule? (skosimp*)
Repeatedly Skolemizing and flattening,
this simplifies to:
putative_1 :

  |-------
{1}   FindPhone(AddPhone(bk!1, nm!1, num!1), nm!1) = num!1


Rule? (expand "FindPhone" )
```

Level 3 Example: Phone Book
Proof Using PVS (continued)

```
Rule? (expand "FindPhone" )
Expanding the definition of FindPhone,this simplifies to:
putative_1 :

  |-------
{1}   AddPhone(bk!1, nm!1, num!1)(nm!1) = num!1

Rule? (expand "AddPhone" )
Expanding the definition of AddPhone,this simplifies to:
putative_1 :

  |-------
{1}   TRUE

which is trivially true.
Q.E.D.

Run time  = 1.02 secs.
Real time = 20.00 secs.
```

Level 1 Example: Phone Book
Deficiencies in the Specification

- Our specification does not rule out the possibility of someone having a "ρ" phone number
- We have not allowed multiple phone numbers per single name
- Our specification does not say anything about whether there should be a warning if a deletion is requested on name that is not in the phone book

How do we remedy these deficiencies?

Level 1 Example: Phone Book
Overcoming Deficiencies 1 & 2

Let N = set of names
P = set of phone numbers
$\mathit{book}: N \rightarrow 2^P$
B = set of functions: $N \rightarrow 2^P$

$\mathit{emptybook}(\mathit{name}) \equiv \varnothing$

$\mathit{FindPhone}: B \times N \rightarrow 2^P$
$\mathit{FindPhone}(\mathit{bk},\mathit{name}) = \mathit{bk}(\mathit{name})$

$\mathit{AddPhone}: B \times N \times P \rightarrow B$
$$\mathit{AddPhone}(\mathit{bk},\mathit{name},\mathit{num})(x) = \begin{cases} \mathit{bk}(x) & \text{if } x \neq \mathit{name} \\ \mathit{bk}(\mathit{name}) \cup \{\mathit{num}\} & \text{if } x = \mathit{name} \end{cases}$$

$\mathit{DelPhone}: B \times N \rightarrow B$
$$\mathit{DelPhone}(\mathit{bk},\mathit{name})(x) = \begin{cases} \mathit{bk}(x) & \text{if } x \neq \mathit{name} \\ \varnothing & \text{if } x = \mathit{name} \end{cases}$$

Level 1 Example: Phone Book
An Additional Deficiency

- Notice that the function `DelPhone` deletes all of the phone numbers associated with a name
- There is no way to remove just one of the phone numbers that is associated with a given name
- The original requirements did not address this situation; to address it, we must define an additional function:

$$DelPhoneNum: B \times N \times P \rightarrow B$$

$$DelPhoneNum(bk, name, num)(x) = \begin{cases} bk(x) & \text{if } x \neq name \\ bk(name) \setminus \{num\} & \text{if } x = name \end{cases}$$

Example: Phone Book
Revised Requirements

Original Requirements

- The phone book shall store the phone numbers of a city
- Given a name, there shall be a way to retrieve an associated phone number
- It shall be possible to add and delete entries from the phone book

Revised Requirements

- For each name in the city, a set of phone numbers shall be stored
- Given a name, there shall be a way to retrieve the associated phone numbers
- It shall be possible to add a new name and phone number
- It shall be possible to add new phone numbers to an existing name
- It shall be possible to delete a name from the phone book
- It shall be possible to delete one of the phone numbers associated with a name
- A warning need not be given for a requested deletion of a name not in the city
- A warning need not be given for a requested deletion of a non-existent phone number

Example: Phone Book
Observations

- Our specification is abstract. The functions are defined over infinite domains.
- In translating the requirements from English into a more formal notation, many things that were left out of the English were explicitly enumerated.
- The formal process exposed ambiguities and deficiencies in the requirements. E.g., we had to choose between

$$book: N \rightarrow P$$

$$book: N \rightarrow 2^P$$

- Putative theorem proving and scrutiny revealed deficiencies in the formal specification

Example: Phone Book
More Observations

- There are many different ways to formally specify
- No matter what representation you chose you are making some decisions that bias the implementation
- The goal is to minimize this bias and yet be complete
- The process of formalizing the requirements can reveal problems and deficiencies and lead to a better English requirements document also
- The formal specification process is similar to the mathematical modeling process of engineering disciplines

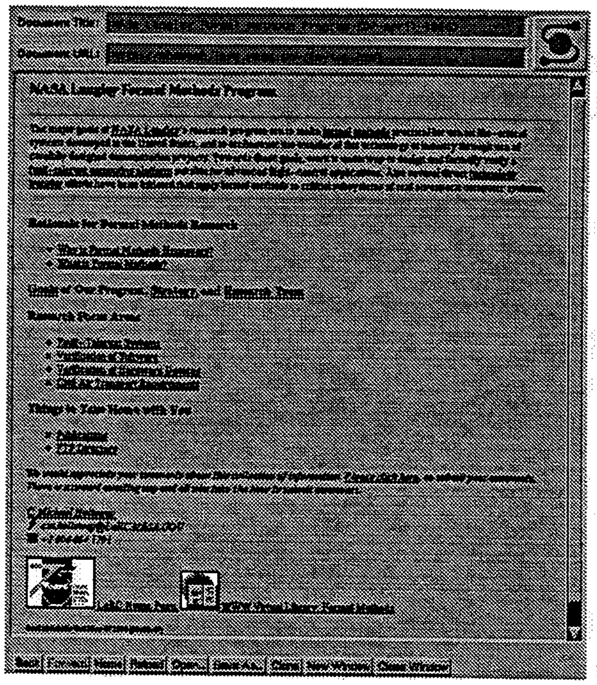
Formal Methods Research at LaRC

- Detailed design with complete level 3 verification of a Reliable Computing Platform
- Design with level 2/3 verification of a transient fault-tolerant clock synchronization circuit and fabrication of the circuit
- With SRI International & Rockwell-Collins, level 3 specification and verification of the microcode of the AAMP5 microprocessor
- With Odyssey Research Associates & Union Switch and Signal, level 3 specification and verification of next-generation railroad control system
- ORA & Honeywell, level 3 specification and verification of aircraft navigation functions

Formal Methods Research at LaRC
(continued)

- Vigyan & SRI working with Loral, JSC, JPL on level 3 specification and verification of some Space Shuttle functions
- SRI working with Allied-Signal on level 3 specification and verification of important algorithms for fault-tolerance
- Conduct periodic workshops on formal methods; previous ones in 1990, 1992, with next one planned for 1995
- Maintain extensive collection of information on the research, accessible through the World Wide Web at URL

  http://shemesh.larc.nasa.gov/fm-top.html



Document Title: [illegible]
Document URL: [illegible]

NASA Langley Formal Methods Program

The major goals of NASA Langley's research program are to make formal methods practical for use on life-critical systems developed in the United States, and to accelerate the transfer of this technology to industry through use of carefully designed demonstration projects. Towards these goals, work is underway to define and formally verify a fault-tolerant operating platform suitable for advanced flight-control applications. Also several direct technology transfer efforts have been initiated that apply formal methods to critical subsystems of real aerospace computer systems.

Rationale for Formal Methods Research

- Why is Formal Methods Necessary?
- What is Formal Methods?

Goals of Our Program, Strategy, and Research Team

Research Focus Areas

- Fault-Tolerant Systems
- Verification of Software
- Verification of Hardware Devices
- Civil Air Transport Requirements

Things to Take Home with You

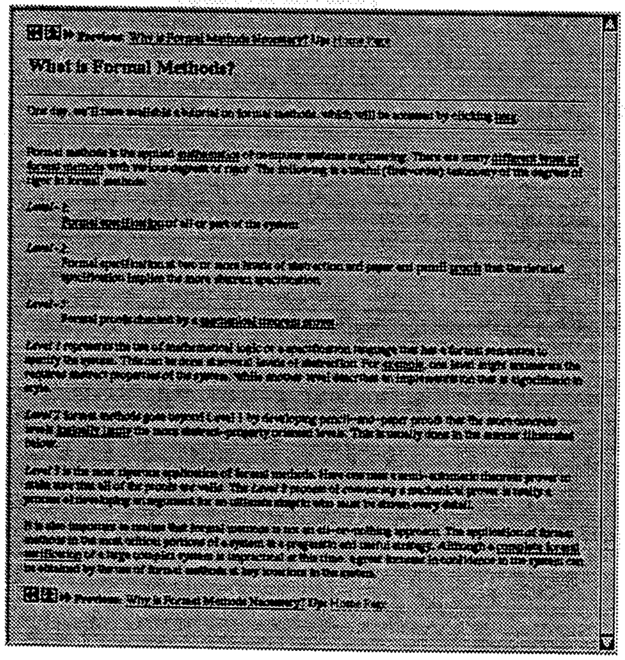
- Publications
- FTP Directory

We would appreciate your comments about the usefulness of information. Please click here to submit your comments. [illegible]

C. Michael Holloway
c.m.holloway@LaRC.NASA.GOV
+1 804-864-1701

LaRC Home Page WWW Virtual Library: Formal Methods

[illegible]



Previous: Why is Formal Methods Necessary? Up: Home Page

What is Formal Methods?

One day, we'll have available a tutorial on formal methods, which will be accessed by clicking here.

Formal methods is the applied mathematics of computer systems engineering. There are many different types of formal methods with various degrees of rigor. The following is a useful (first-order) taxonomy of the degrees of rigor in formal methods:

Level-1
: Formal specification of all or part of the system.

Level-2
: Formal specification at two or more levels of abstraction and paper and pencil proofs that the detailed specification implies the more abstract specification.

Level-3
: Formal proofs checked by a mechanical theorem prover.

Level 1 represents the use of mathematical logic or a specification language that has a formal semantics to specify the system. This can be done at several levels of abstraction. For example, one level might enumerate the required abstract properties of the system, while another level describes an implementation that is algorithmic in style.

Level 2 formal methods goes beyond Level 1 by developing pencil-and-paper proofs that the more concrete levels logically imply the more abstract-property oriented levels. This is usually done in the manner illustrated below.

Level 3 is the most rigorous application of formal methods. Here one uses a semi-automatic theorem prover to make sure that all of the proofs are valid. The *Level 3* process of convincing a mechanical prover is really a process of developing an argument for an ultimate skeptic who must be shown every detail.

It is also important to realize that formal methods is not an all-or-nothing approach. The application of formal methods to the most critical portions of a system is a pragmatic and useful strategy. Although a complete formal verification of a large complex system is impractical at this time, a great increase in confidence in the system can be obtained by the use of formal methods at key locations in the system.

Previous: Why is Formal Methods Necessary? Up: Home Page

ORIGINAL PAGE IS
OF POOR QUALITY