A REVIEW OF FORMAL METHODS

Robert L. vienneau
INDEX

- 1. INTRODUCE
- 2. DEFINING AND OVERVIEW OF FORMAL METHODS
- 3. SPECIFICATION METHODS
- 4. LIFE CYCLES AND THE TECHNOLOGIES WITH INTEGRATED FORMAL METHODS
- CONCLUSION
Introduction

- The 1970s witnessed the structured programming revolution.
- After much debate, software engineers became convinced that better programmers result from following certain precept in program design.
- Formal Methods have the potential of leading to further revolutionary change in practice and hav provided the underlying basis for past change.
Definition and overview of formal methods

2.1. Use of Formal Methods

- They are directly applicable during the requirements, design, and coding phases and have important consequences for testing and maintenance.
- They have influenced the development and standardization of many programming languages, the programmer's most basic tool.
Definition and overview of formal methods

2.1. Use of Formal Methods

A broad view of formal methods includes all applications of (primarily) discrete mathematics to engineering problems.
Definition and overview of formal methods

2.1. Use of Formal Methods

- A more narrow definition, better conveys the change in practice recommended by advocates of formal methods.
  - First, formal methods involve the essential use of a formal language.
  - Second, formal methods in software support formal reasoning about formulae in the language.
Definition and overview of formal methods

2.1.1. What Can Be Formally Specified

- Formal methods support precise and rigorous specifications of those aspects of a computer system capable of being expressed in the language.
Definition and overview of formal methods

2.1.1. What Can Be Formally Specified

- Since defining what a system should do, and understanding the implications of these decisions, are the most troublesome problems in software engineering, this use of formal methods has major benefits.
Definition and overview of formal methods

2.1.1. What Can Be Formally Specified

- Formal methods can be used to specify aspects of a system other than functionality.
- For example, formal methods are sometimes applied in practice to ensure software safety and security properties of computer programs.
- The benefits of proving that unsafe states cannot arise, or that security is assured, can justify the cost of complete formal verifications of the relevant portions of software systems.
Definition and overview of formal methods

2.1.2. Reasoning about a Formal Description

- Dose a description imply a system should be in several states simultaneously?
- Do all legal inputs that yield one and only one output?
- What surprising results, perhaps unintended, can be produced by a system?
Definition and overview of formal methods

2.1.2. Reasoning about a Formal Description

- Formal methods support formal verification, the construction of formal proofs that an implementation satisfies a specification.
- The possibility of constructing such formal proofs was historically the principal driver in the development of formal methods.
Definition and overview of formal methods

2.1.3. Tools and Methodology

- For proponents of formal methods, the ultimate end product of software development is not solely a working system.
- Specification and demonstrations that the program meets its specification are of equal importance.
Definition and overview of formal methods

### 2.1.3. Tools and Methodology

- A proof is very hard to develop after the fact.
- Consequently, proofs and programs should be developed in parallel, with close interconnections in their development history.
- Since programs must be proven correct, only those constructions that can be clearly understood should be used.
Definition and overview of formal methods

2.1.3. Tools and Methodology

- Formal methods have also inspired the development of many tools.
- Programs to help maintain and automate proofs are an obvious example of such tools.
Definition and overview of formal methods

2.1.3. Tools and Methodology

- in some sense, no programmer can avoid formal methods, for every programming language is by definition, a formal language.
- Ever Since Algol 1960 was introduced, standards defining programming languages have used a formal notation for defining language syntax, namely Backus-Naur Form.
Definition and overview of formal methods

2.2. Limitations of Formal Methods

Given the applicability of formal methods throughout the life cycle, and their pervasive possibilities for almost all areas of software engineering, why are they not more widely visible?
Definition and overview of formal methods

2.2. Limitations of Formal Methods

- One issue is pedagogic.
  - Revolutions are no made by conversion, but by the old guard passing away.
- One the other hand, it is not the case that the only barrier to the widespread transition of this technology is lack of knowledge on the part of practitioners.
- Formal methods suffer from certain limitations.
Definition and overview of formal methods

- **2.2.1 Requirements Problem**
  - In particular, a formal verification can prove that an implementation satisfies a formal specification, but it cannot prove that a formal specification captures a user's intuitive understanding of a system.
  - In other words, formal methods can be used to verify a system, but not to validate it.
Definition and overview of formal methods

- **2.2.1 Requirements Problem**
  - One influential study found that the three most important problems in software development are:
    1. The thin spread of application domain knowledge
    2. Change in and conflicts between requirements
Definition and overview of formal methods

2.2.1 Requirements Problem

- These findings suggest the reduction of informal application knowledge to a rigorous specification is a key problem area in the development of large systems.

- Empirical evidence suggests, however, that formal methods can make a contribution to the problem of adequately capturing requirements.
Definition and overview of formal methods

2.2.1 Requirements Problem

- Empirical evidence suggests, however, that formal methods can make a contribution to the problem of adequately capturing requirements.
- The discipline of producing a formal specification can result in fewer specification errors.
- Furthermore, implementers without an exceptional designer's knowledge of the application area commit fewer errors when implementing a formal specification than when relying on hazy knowledge of the application.
- The discipline of producing a formal specification can result in fewer specification errors.
Definition and overview of formal methods

- 2.2.1 Requirements Problem
  - A specification acts as a "contract" between a user and a developer.
  - Using specifications written in a formal language to complement natural language descriptions can make this contract more precise.
Definition and overview of formal methods

2.2.1 Requirements Problem

- Finally, developers of automated programming environments, which use formal methods, have developed tools to interactively capture a user's informal understanding and thereby develop a formal specification.
- Still, formal methods can never replace deep application knowledge on the part of the requirements engineer, whether at the system or the software level.
Definition and overview of formal methods

2.2.2 Physical Implementation

- The second major gap between the abstractions of formal methods and concrete reality lies in the nature of any physically existing computer.
- Formal methods can verify that an implementation satisfies a specification when run on an idealized abstract machine, but not when run on any physical machine.
- Memory chips and integrated circuits may contain bugs.
Definition and overview of formal methods

2.2.3 Implementation Issues

- The gaps between users' intentions and formal specifications, and between physical implementations and abstract proofs, create inherent limitations to formal methods, no matter how much they may be developed in the future.
- The introduction of a new technology into a large-scale software organization is not a simple thing, particularly a technology as potentially revolutionary as formal methods.
Definition and overview of formal methods

2.2.3 Implementation Issues

- Decisions must be made about whether the technology should be completely or partially adopted. Appropriate accompanying tools need to be acquired.
- Current personnel need to be retrained, and new personnel may need to be hired.
- Existing practices need to be modified, perhaps drastically.
Definition and overview of formal methods

- 2.2.3 Implementation Issues
  - Optimal decisions depend on the organization and the techniques for implementing formal methods.
  - One scheme for using formal methods on real-world projects is to select a small subset of components for formal treatment, thus finessing the scalability issue.
Definition and overview of formal methods

- 2.2.3 Implementation Issues

- No matter to what extent an organization decides to adopt formal methods, if at all, training and education issues arise.
- Education in formal methods should not be confined to degreed university programs for undergraduates newly entering the field.
- Means need to be found, such as seminars and extension courses, for retraining an existing workforce.
SPECIFICATION METHODS

- Formal methods were originally developed to support VERIFICATIONS, BUT MANY PROJECTS USING FORMAL METHODS HAVE used them only to establish properties of specifications.
- This section briefly describes some characteristics of different methods now available.
3.1 Semantic Domain

- A formal specification language contains an alphabet of symbols and grammatical rules that define well-formed formulae.
- These rules characterize a language’s "syntactic domain." The syntax of a language shows how the symbols in the language ships between them are characterized by the syntax of a language.
Definition and overview of formal methods

- **3.1 Semantic Domain**
  - Three major classes of semantic domains exist.
    1. Abstract data type specification languages
    2. Process specification languages
    3. Programming languages
Definition and overview of formal methods

3.2 Model-Oriented and Property-Oriented Methods

- The distinction between model-oriented and property-oriented methods provides another dimension for classifying formal methods.
3.2 Model-Oriented and Property-Oriented Methods

- Model-oriented methods have also been described as constructive or operational.
- Typically, a model will use abstract mathematical structures, such as relations, functions, sets, and sequences.
3.2 Model-Oriented and Property-Oriented Methods

- Property-oriented methods are also described as definitional or declarative.
- A specification describes a minimum set of conditions that a system must satisfy.
- Any system that satisfies these conditions is functionally correct, but the specification does not provide a mechanical model showing how to determine the output of the system from the input.
Definition and overview of formal methods

### 3.3 Use of Specification Methods

- In general, formal methods provide for more precise specifications.
- Since the earlier a fault is detected, the cheaper it can be removed, formal specification methods can dramatically improve both productivity and quality.
- In particular, customers should be presented with the English version, not a formal specification.
- Choosing between model-oriented and property-oriented methods also depends on project-specific details and experience.
4.0 LIFE CYCLES AND TECHNOLOGIES WITH INTEGRATED FORMAL METHODS

- Two methods of integrating formal methods in software processes can be distinguished:
  - One with heavy use of automated tools
  - and the other with non-mechanical, non-automated proofs.
4.0 LIFE CYCLES AND TECHNOLOGIES WITH INTEGRATED FORMAL METHODS

4.1 Verification Systems and Other Automated Tools

- An automated verification system provides a means for the user to demonstrate the existence of a formal proof of a software system.
- Another set of tools support model checking.
- Model checking tools overcome state explosion problem in practice by the use of symbolic techniques.
4.0 LIFE CYCLES AND TECHNOLOGIES WITH INTEGRATED FORMAL METHODS

- 4.2 The Cleanroom as a Life Cycle with Integrated Use of Formal Methods

  - The Cleanroom methodology integrates non-mechanized formal methods into the life cycle.
  - Specification developed by the Cleanroom process include:
    - Explicit identification of functionality to be included in successive releases
    - Failure definitions, including level of severity
    - The target reliability as a probability of failure-free operation for a specified time
    - The operational profile for each increment, that is, a model of user behavior of the system
    - The reliability model that is applied in system testing to demonstrate reliability.
Conclusions

- This report has briefly surveyed various formal methods and the conceptual basis of these techniques.
- **Formal methods can provide:**
  - More precise specifications
  - Better internal communication
  - An ability to verify designs before executing them during test
  - Higher quality and productivity
- **knowledge of formal methods is needed to completely understand these popular technologies and to use them most effectively. These technologies include:**
  - Rapid prototyping
  - Object Oriented Design (OOD)
  - Structured programming
  - Formal inspections.
Conclusions

- The full-scale use, transition, and cost-effective use of formal methods is not fully understood. An organization whose leaders can figure out how to effectively integrate formal methods into their software process will be likely to produce higher quality software and thereby gain a competitive advantage.