Formal Methods in Software Engineering

Aristides Dasso
Universidad Nacional de San Luis, Argentina

Ana Funes
Universidad Nacional de San Luis, Argentina

INTRODUCTION

As a general rule, all engineering applications use mathematics or mathematical tools as a basis for their development. However, software engineering is an exception to this rule.

Formal methods (FM) are[1] a collection of methodologies and related tools, geared to the production of software employing a mathematical basis. There are a number of different formal methods each having its own methodology and tools, especially a specification language.

We can say that FM are "mathematically based techniques for the specification, development and verification of software and hardware systems" (retrieved on October 15, 2003, http://foldoc.doc.ic.ac.uk/foldoc).

Most FM are based mainly on specifications – for which they normally have a language to express it. Sometimes, there is also a method to use the language in the software development process.

The aims of FM can vary according to the different methodologies, but they all shared a common goal: the production of software with the utmost quality mainly based on the production of software that is error free. To achieve this, the different FM have developed not only a theory but also different tools to support the formal process.

FM can cover all of the steps of the life cycle of a software system development from requirement specification to deployment and maintenance. However, not all FM have that capacity.

BACKGROUND

Some FM rely on development of a calculus or transformation, where the engineer starts with an expression and then following predefined rules applies them to obtain an equivalent expression. Successive calculations lead to implementation. On the other hand, there are FM that rely on the "invent and verify" technique, where the engineer starts by inventing a new design, which afterwards needs to be verified as correct. From this verified design, implementation follows.

There are several styles of formal specification. Some are mutually compatible while others are not. Table 1 shows a possible classification of the different styles.

Formal languages have formal definitions not only of their syntax but also of their semantics. Table 2 shows a possible classification for the different formal semantic definitions.

Hinchey and Bowen (1995) say that "formal methods allow us to propose properties of the system and to demonstrate that they hold. They make it possible for us to examine system behavior and to convince ourselves that all possibilities have been anticipated. Finally, they enable us to prove the conformance of an implementation with its specification".

In NASA's Langley Research Center site for formal methods, there is a nice definition and also an explanation of different degrees of rigour in FM:

"Traditional engineering disciplines rely heavily on mathematical models and calculation to make judgments

Table 1. Summary of specification language characteristics

Model-oriented. Based on mathematical domains. For example, numbers, functions, sets, etc. Concrete.	**Property-oriented**. Based on axiomatic definitions. Abstract.
Applicative. Does not allow the use of variables.	**Imperative or State-oriented**. Allows the use of variables.
Static. Do not include provisions for handling time.	**Action**. Time can be considered in the specification. There are several ways of doing this: considering time as linear or branching, synchronous, asynchronous, etc.

Copyright © 2005, Idea Group Inc., distributing in print or electronic forms without written permission of IGI is prohibited.

Table 2. Summary of semantic definitions styles of specification languages

Operational. Concrete, not well suited for proofs.
Denotational. Abstract, well suited for proofs.
Axiomatic. Very abstract, normally only limited to conditional equations.

about designs. For example, aeronautical engineers make extensive use of computational fluid dynamics (CFD) to calculate and predict how particular airframe designs will behave in flight. We use the term 'formal methods' to refer to the variety of mathematical modelling techniques that are applicable to computer system (software and hardware) design. That is, formal methods is the applied mathematics of computer system engineering, and, when properly applied, can serve a role in computer system design analogous to the role CFD serves in aeronautical design.

Formal methods may [be] used to specify and model the behavior of a system and to mathematically verify that the system design and implementation satisfy system functional and safety properties. These specifications, models, and verifications may be done using a variety of techniques and with various degrees of rigour. The following is an imperfect, but useful, taxonomy of the degrees of rigour 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 rigourous 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.

Formal methods is not an all-or-nothing approach. The application of formal methods to only 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. (Retrieved October 31, 2003, http://shemesh.larc.nasa.gov/fm/fm-what.html)

NASA as well as other government bodies in the USA, Europe and elsewhere are using FM especially in avionics and systems where the utmost reliability is needed. Some examples from NASA are: Small Aircraft Transportation System (SATS), Formal Analysis of Airborne Information for Lateral Spacing (AILS), also NASA's contractors use FM. For more information on this and other projects, see http://shemesh.larc.nasa.gov/fm/.

As it is stated previously in NASA's definition of the levels of degree of rigour, they are imperfect. Others exist. Most of the FM included in the following text have as an integral part not only a language but also a methodology included, and most of the time this methodology implies different levels of rigour in its use. For example, RAISE – that have its own method – presents three degrees of formality (The RAISE Method Group, 1995):

- *formal specification only*, where formality is only applied to the specification procedure.
- *formal specification and rigourous development*, where formality is applied to the specification procedure as above, and rigour to the development process. This means that the developer starts writing abstract specifications, goes on developing more concrete ones and recording the development relations between them. These relations are then examined, however they are not justified.
- *formal specification and formal development*, it is the extension of the previous degree to do the justification as well.

Here is a not all-inclusive list of FM:

- **ASM** (Abstract State Machines) "methodology for describing simple abstract machines which correspond to algorithms" (Retrieved September 15, 2003, from http://www.eecs.umich.edu/gasm/).
- **B–Method** "B is a formal method for the development of program code from a specification in the Abstract Machine Notation" (Retrieved October 23, 2003, from http://www.afm.lsbu.ac.uk/b/).
- **CSP** (Communicating Sequential Processes) " process algebra originated by C. A. R. Hoare (http://www.afm.lsbu.ac.uk/csp/).

5 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/formal-methods-software-engineering/14412

Related Content

Fault Tolerance for Distributed and Networked Systems
Wenbing Zhao, Louise E. Moserand P. Michael Melliar-Smith (2005). *Encyclopedia of Information Science and Technology, First Edition (pp. 1190-1196).*
www.irma-international.org/chapter/fault-tolerance-distributed-networked-systems/14409

Systems Requirements and Prototyping
Vincent C. Yen (2006). *Cases on Information Technology Planning, Design and Implementation (pp. 122-136).*
www.irma-international.org/chapter/systems-requirements-prototyping/6365

It's All in the Game: How to Use Simulation-Games for Competitive Intelligence and How to Support Them by ICT
Jan Achterberghand Dirk Vriens (2008). *Information Communication Technologies: Concepts, Methodologies, Tools, and Applications (pp. 1445-1458).*
www.irma-international.org/chapter/all-game-use-simulation-games/22748

The Influence of National and Organisational Culture on Knowledge Sharing in Distributed Teams
Kerstin Siakas, Elli Georgiadouand Dimitrios Siakas (2020). *Information Diffusion Management and Knowledge Sharing: Breakthroughs in Research and Practice (pp. 533-555).*
www.irma-international.org/chapter/the-influence-of-national-and-organisational-culture-on-knowledge-sharing-in-distributed-teams/242148

ISEkFT: An IBE-Based Searchable Encryption Scheme With k-Keyword Fuzzy Search Trapdoor
Mamta, Brij B. Guptaand Syed Taqi Ali (2019). *Journal of Information Technology Research (pp. 133-153).*
www.irma-international.org/article/isekft/234477