Formalization Process in Software Development

Aristides Dasso
Universidad Nacional de San Luis, Argentina

Ana Funes
Universidad Nacional de San Luis, Argentina

INTRODUCTION

Nowadays, software engineering (SE) is considered more frequently an engineering discipline. Several definitions have been proposed by different authors, and many of them agree in affirming that SE is the application of principles and systematic practices for the development of software. That is—as it was established by the IEEE (1990)—SE is the application of engineering to the software.

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. Not all the techniques and software development methods have a formal basis. Formal methods[1] (FM) rely on mathematical foundations.

FM are 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, specially a specification language.

As it is expressed in Wikipedia (2006) and Foldoc (2006), we can say that FM are "mathematically based techniques for the specification, development and verification of software and hardware systems."

FM are based on the production of formal specifications—for which they have a formal 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 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, and not always it is convenient to apply them. It is necessary to make an evaluation between pros and cons before applying FM in the software development process.

FORMAL METHODS

It is important to make a distinction between a formal notation or language and a formal system. A formal notation is used to produce a formal specification, and it has a formal syntax and semantics. A formal system, besides these two components, includes a proof system—the deductive mechanism of the formal system. The syntax is described by a grammar and defines the set of well-formed formulas of the language. The meaning of these formulas is given by the semantics of the language. Finally, the syntactic manipulations of these formulas are achieved by using the inference mechanism of the formal system, which allows the derivation of new well-formed formulas from those present in the language.

Alagar and Peruyasamy (1998) establish that the difference between a FM and a formal system is the automatic support for specification and the availability of mechanized proofs.

All FM are based on mathematical formalisms, but we can distinguish two different development approaches. On the one hand, we can find the transformational methods based in transformations and on the other hand, those based in the "invent and verify" principle. The former rely for development on 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. In FM relying on "invent and verify" technique, the engineer starts by inventing a new design, which afterward 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 styles.

In NASA's Langley Research Center site for formal methods there is a nice definition and also an explanation of different degrees of rigor in FM (http://shemesh.larc.nasa.gov/fm/fm-what.html):

Table 1. Summary of specification language characteristics

Model-oriented. Based on mathematical domains. For example numbers, functions, sets, and so forth. 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. Does 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, and so forth.

Table 2. Summary of semantic definitions styles of specification languages

Operational. Concrete, not well suited for proofs. The meaning of a system is expressed as a sequence of actions of a simpler computational model.
Denotational. Abstract, well suited for proofs. The meaning of a system is expressed in the mathematical theory of domains.
Axiomatic. Very abstract, normally only limited to conditional equations. The meaning of the system is expressed in terms of preconditions and postconditions.

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

As it is said in NASA's definition of the levels of degree of rigor, they are imperfect; others exist. Most of the FM included next 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 rigor in its use. For