David Lambertson

**A Review of Formal Methods Models Summary**

**Important Points from Article**

* “After much debate, software engineers became convinced that better programs result from following certain precepts in program design. Recent imperative programming languages provide constructs supporting structured programming. Achieving this consensus did not end the debate over programming methodology.” [323]
* “Formal methods have provided a unifying philosophy and central foundation upon which these methodologies have been built. Those who understand this underlying philosophy can more easily adopt these and other programming techniques.” [323]
* “Formal methods have the potential of leading to further revolutionary changes in practice and have provided the underlying basis for past changes. These reasons make it imperative that software managers and engineers be aware of the increasingly widespread debate over formal methods.” [324]
* “A formal method in software development is a method that provides a formal language for describing a software artifact (for instance, specifications, designs, or source code) such that formal proofs are possible, in principle, about properties of the artifact so expressed.” [324]
* “The concept of formalism in formal methods is borrowed from certain trends in 19th and 20th century mathematics. The development of consistent non-Euclidean geometries, in which supposedly parallel lines may intersect, led mathematicians to question their methods of proof and to search for more rigorous foundations.” [324]
* “Master of formal methods in software requires an understanding of this mathematics background. Mathematical topics of interest formal logic, both the propositional calculus and predicate logic; set theory; formal languages; and automata, such as finite state machines.” [324]
* “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.” [325]
* “Software engineers produce models and define the properties of systems at several levels of abstraction. Formal methods can be employed at each level. A specification should describe what a system should do, but not how it is done.” [325]
* “Developments in supporting tools and methodologies have an accompanied new concepts for formalizing software artifacts. For proponents of formal methods, the ultimate end product of software development is not solely a working system. Specifications and demonstrations that the program meets its specifications are of equal importance.” [326]
* “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 (BNF).” [326]

**Things I Didn't Agree With**

“Since software development is, therefore, a ‘cut and fit’ process, such complex systems can be expected to be full of bugs.” [323]

I don’t agree with this statement because while software development is like a ‘cut and fit’ process, it does not guarantee the code will be full of bugs. Programmers take different components and build them to work with each other. This will introduce some bugs, but programmers work hard to fix them as they are in the building the process.

**Things I Did Not Understand**

There was nothing in this article I did not understand.