Andy Hudson

CIS 350

Software Apocalypse Response

The coming software apocalypse is, according to the article, the future possibility of widespread critical software failures due to the complexity of software development and maintenance increasing beyond the human capacity to manage it, such that too wide a gap is created between what problems programmers intend to solve and how the code they write solves these problems correctly. The article presents model-based design as the primary solution to the software apocalypse: a method of software engineering that focuses heavily on writing requirements and logic, or “specifications” in the given language TLA+, based on how a system should operate and then generating code based on those specifications that is exhaustively tested to operate exactly according to those specifications. The hope of model-based design is to remove the need for a programmer to think about the problem they’re solving more than the method of solving it (through code) because too many human errors result from not thinking a problem through well enough beforehand.

It is, in my opinion, completely obvious that a software apocalypse is coming, and it’s really surprising that there hasn’t been some kind of a national tragedy yet (like a traffic accident or some hospital-related incident) due to widespread critical software failures. Things like this are part of the reason that I wanted to become a programmer: the world is going to depend on code, it needs to be managed one way or another, and it’s clearly spiraling out of human control now, much like the androids of science-fiction novels. Model-based design is a fantastic idea for fixing the problem (I’ve been thinking of something similar, a way to streamline solving basic and frequent problems like code libraries do, but on a grand scale), but the only problem lies in actually fixing all of the code already developed, which is a huge undertaking by amount of code and by computer scientists’ perspectives on programming.