

Software Lacks Adult Supervision

By
Gordon Morrison

[Breaking the Time Barrier: The Temporal Engineering of Software](#)

In 1995 I was interviewing for a consulting position with the VP of engineering when he made the comment that “this project lacks adult supervision”. The design and architecture were out of control. With millions of dollars invested the project failed and ceased operations.

That comment has stuck with me; it re-energized my search for a way to improve the process of software engineering. I found the object approach used by this company was no better than any other approach; in fact it created so many classes and objects that the application could not be maintained. As a consultant I’ve been pulled into projects late in their development. All were in trouble. And very bright capable people headed every project. The failure was caused by a lack of leadership, respect for that leadership, and an engineering discipline. The technical problems were not that difficult.

So when I say “software lacks adult supervision” does that mean there is someone to blame? The answer is no. Software engineering has always been the poor stepchild in the engineering professions. When a bright middle-school kid can crank out code it does not bode well for the profession. But that perception and many others lie at the root of the problem for the professional that wants to produce quality code.

Academic and commercial software engineering has focused on the management of the source code and the build process. The engineering of software has been ignored. The professional coder has model driven tools like “Rational Rose” from IBM. But this tool is known to produce a lot of wallpaper and quickly get out of sync with the final application. The tools that have been created under the guise of Model Driven Architecture (MDA) have been “a good idea poorly implemented”.

Why do the current MDA tools get out of sync? They are the result of many ideas and many tools combined into a massive abstraction of every possible concept in computer thought; that one or many could conceive; and implement without planning; or discipline in the ease of use or the coherent implementation of a complete application. Most of the tools that I’ve used create an obstacle course of dialog boxes preventing productivity. I want an MDA tool that’s as easy to use as the Delphi version 7 environment.

Harry Pierson wrote that the “code is model” for an application. I believe that of all the modeling tools I’ve used or read about that is true. Therefore it is my premise that every model should start with a form of BNF that begins to define that application as I do in my book because eventually everything that goes into the application must be parsed through a compiler. And, what better way to stay in sync throughout the development process than to stay within a modeling BNF.