This paper is basically a statement on the current tendency of researchers to involve themselves in the software development process without properly addressing the type of problems they are hoping to solve. As the title suggests, the abundance of tools, i.e. specification languages, design languages and special purpose languages fail to confront the major obstacle of programming -- mastery of the language.

General programming skills are divided into two categories: syntactic and semantic mastery. The later is often confused by syntactic resemblances of other languages. Smoliar and Barstow give examples of different methods, based on environment, used in assisting programmers. Error correcting systems which resolve minor syntactic blunders, specific language editors, natural language query systems, are some of the tools for syntactic understanding. Extensible languages, knowledge-based program-editing and automatic programming remove some of the burden of semantic mastery.

The remainder of the article covers their rule-based-system, (designed for quantitative log interpretations) as an integration of those two categories.

Despite beign intuitively reasonable in their ideas, they ignore assissing the value of having a gamut of different programming languages. This is termed as "throwing languages at a problem domain in great numbers of great flexibility." Refinement of concepts and ideas do not seem to be part of the protrayed picture. Nevertheless, the need for simplicity and ease in programming cannot be avoided.