***Essential Difficulties***

The anomaly is not that software progress is so slow but that computer hardware progress is so fast. The essence of a software entity is a construct of interlocking concepts: data sets, relationships among data items, algorithms, and invocations of functions. This essence is abstract, in that the conceptual construct is the same under many different representations. It is nonetheless highly precise and richly detailed. The hard part of building software is the specification, design, and the testing of the conceptual construct.

Inherent properties of the irreducible essence of modern software systems:

**Complexity:** Software entities are more complex for their size than any other human construct because no two parts are alike. Digital computers are more complex due to their very large number of states.

It is necessary an increase in the number of different elements when it comes to scaling-up a software. Its complexity increases much more than linearly and is in essential property, not an accidental one.

The difficulty of communication among team members leads to product flaws, cost overruns and schedule delays. Not only technical problems but management problems come from the complexity, and it makes overview hard.

**Conformity:** The software must confirm because it has recently come to the scene, in others, it must conform because it is perceived as the most conformable. Anyhow, in all cases, much complexity comes from conformation to other interfaces; this cannot be simplified out by any redesign of the software alone.

**Changeability:** The software entity is constantly subject to pressures for change and usually superseded by later models. All the successful ones get changed as people try in new cases at the edge of, or beyond, the original domain.

The software product is embedded in a cultural matrix of applications, users, laws, and machine vehicles. These all change continually, and their changes inexorably force change upon the software product.

**Invisibility:** Software is invisible. Geometric abstractions are powerful tools, and geometric reality is captured by it. The reality of software is not inherently embedded in space.

***Past Breakthroughs Solved Accidental Difficulties***

**High-level Languages:** the most powerful stroke for software productivity, reliability and simplicity. It frees a program from much of its accidental complexity. On the other hand, at some point the elaboration of a high-level language becomes a burden that increases, not reduces, the intellectual task of the user who rarely uses the esoteric constructs.

**Time-sharing:** It preserves immediacy, and hence enables us to maintain an overview of complexity. The most serious effect may well be the decay of grasp of all that is going on in a complex system. Meanwhile, the main effect is to shorten system response time.

**Unified Programming Environments:** *Unix* and *Interlisp* attack the accidental difficulties of using programs together, by providing integrated libraries, unified file formats, piles and filters.