



Contact Information:

info@biglever.com
www.biglever.com
512-426-2227

A Discontinuous Jump into the New Software Product Line Frontier

Report #200905141r2

This report is based on a Guest View article written by Charles Krueger, CEO of BigLever Software, in the January 1, 2009 issue of SD Times.

1. At Long Last, a Discontinuous Jump

For an engineer, creating a dishwasher is more fun than washing dishes. Creating a car or an airplane to go places is more exciting than walking or running. Creating a factory to assemble cars is a more interesting challenge than assembling cars.

The drive to create new and improved tools is an innate part of the human condition, hardcoded into the DNA. We are never quite happy with the current state of affairs and are always looking for ways to create new tools that minimize our labors and maximize our well-being, relaxation and recreation.

The historical timelines for our tool capabilities progress through continuous and incremental improvements, with an occasional discontinuous jump. The term “discontinuous jump” comes from calculus where, for our tool example, a smooth increasing graph reaches a point of discontinuity, makes a vertical stair-step jump, and then continues again as a smooth increasing graph. Discontinuous jumps in the context of tools arise from pivotal discoveries that lead to paradigm shifts.

Archeological indicators of discontinuous jumps in tools can be seen in the concentric fortifications around ancient cities. Smaller inner walls are circumscribed by taller and thicker walls, encircled by huge earthen embankments and trenches. Each concentric layer demarcates a point in time where there was a discontinuous jump in the tools of war that rendered the previous fortifications obsolete.

Most software engineers – at least those not yet of retirement age – have never experienced a discontinuous jump in our tools and methods. Good news: that situation is at long last about to change.

2. Incremental and Discontinuous Jumps in Software Engineering

The software engineering field has exhibited a rather monotonous timeline of continuous and incremental improvements in our tools and methods, with only one notable discontinuous jump in our 60+ year history. And that event occurred over 50 years ago when high-level languages and compilers supplanted low-level binary and assembly language programming to reduce the program statement count by approximately a factor of 20 and software delivery time and effort by approximately a factor of 5.

Of course, this rather mundane reality hasn't dampened the spirits of our marketing teams. They are still having their fun with claims of order-of-magnitude improve-

ments. But if you look beyond the anecdotal evidence and individual case studies, the software field has not experienced an across-the-board discontinuous jump since the advent of Fortran and COBOL compilers.

In his widely cited article No Silver Bullet, Fred Brooks argued convincingly that the lack of discontinuous jumps in software engineering is a predictable and fundamental characteristic of creating software. Although we can expect continuous and incremental improvements in the way we express the problems and solutions we address with software, Brooks argues that the complexity of these problems remains relatively constant and is very high. We can never reduce this essential complexity and it determines the lower limit on how quickly we can express a solution through software.

3. New Forces at Play

In spite of Brooks' prediction, two forces are at play today in creating a new and rare discontinuous jump in software engineering. The first of these forces is in the problems we are asked to solve. It is in the prevalent demand for most software and software-based system companies to create and maintain larger and larger product lines – portfolios of similar products with variations in features and functions – rather than just individual one-of-a-kind products. Using traditional development tools and methods, the complexity of developing a product line grows proportionally to the square of the number of products (or, order-of- N^2). As a result, the complexity of engineering our expanding product lines is outpacing the linear incremental improvements of our software tools and methods.

The second of these forces is a new approach in the software tools and methods for engineering product lines, referred to as Software Product Lines (SPL) engineering. A new generation of SPL tools and methods has constrained the complexity of creating and maintaining a product line from order-of- N^2 to a linear order-of- N . The result is a consistent factor-of-2 to factor-of-10 improvement in software development metrics such as productivity, defect density, time-to-market, and portfolio scalability. In other words, a discontinuous jump.

This raises a few important questions.

In light of Brooks' longstanding prediction that we wouldn't see such an across-the-board discontinuous jump of improvement in software engineering tools and methods, how is this possible? There is an implicit assumption in Brooks' argument that the complexity of the problems we solve with software will evolve slower than the capabilities of the tools we use to solve them. The expanding product line problem, with its order-of- N^2 complexity growth, invalidates this assumption, inducing a rapid increase in a new type of complexity for most organizations and opening the door for pivotal discoveries and innovations.

4. The Right Point of View

What is the key ingredient of an SPL solution that allows for the dramatic reduction in complexity compared to traditional software tools and methods? Traditional approaches – and even early generation SPL approaches – tend to take a product-centric view, where every defect fix or requirement enhancement on any product in a product line may need to be reflected in the same or similar ways in other products in the product line. This interdependency among all products leads to the order-of- N^2 complexity.

The new generation SPL approaches with a linear order-of- N complexity exploit a manufacturing approach to creating and maintaining a product line. It is very similar to engineering a single automated production line for the manufacturing of automobiles. The assets that comprise the products and the automated production capability for assembling and configuring the products are engineered as a single system rather than a multitude of products. The products themselves are demoted to a secondary side effect of the manufacturing system. A defect fix or enhancement intended for any particular product is performed on the singular system of assets and the auto-

mated production capability, such that it can be automatically applied to any or all products. The order-of- N^2 relationship between products is eliminated by taking this single system perspective.

As a colleague once observed, *the right point of view saves 20 points of IQ.*

5. Into the New Frontier

The exciting thing about discontinuous jumps is that they open up new frontiers, offering new possibilities and challenges that could not even be conceived of in the old paradigm. What new challenges and opportunities lie ahead in the new frontier of SPL engineering?

Like the civil engineers of the ancient cities, the first thing to note is that paradigm shifts can be used against you, as well as to your benefit. The first challenge you see may come from those of your competitors who make the shift before you do.

Another challenge is that organizations do well with continuous and incremental improvements, but not with paradigm shifts. This is particularly true in software engineering since we haven't experienced discontinuous jumps and therefore haven't established a culture that is accepting of paradigm shifts.

As for opportunities in the new frontier of SPL engineering, there have been many published unexpected scenarios and remarkable strategic benefits. But here, I'll leave to your imagination to contemplate how your business and engineering organization could take advantage of a discontinuous jump that offered you factor of 2 to 10 improvements in productivity, reaction time for new market opportunities, portfolio size and product diversity. What if the scale, scope and diversity of your product line were only limited by the imagination of your organization rather than limited by the capacity of your engineering team?

