Componentology: A Hard Scientific Foundation for Software Engineering

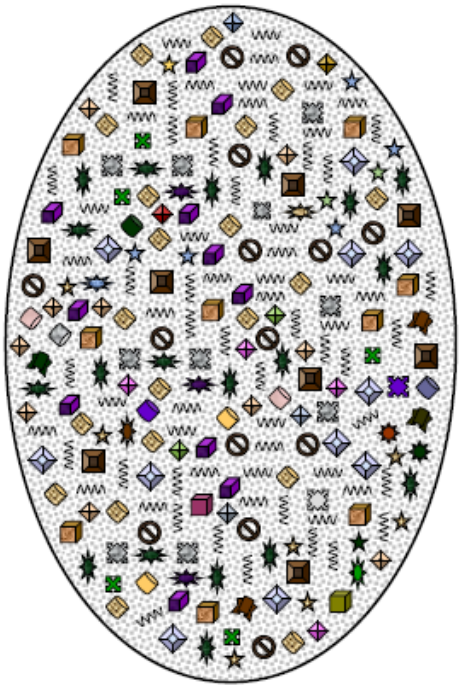
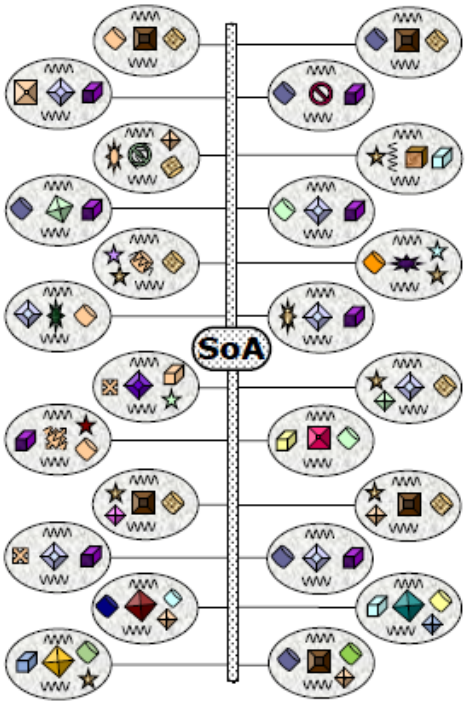
Heliocentrism was the true and objective reality 2000 years ago, remains the true and objective reality in the 21st century, and will continue to be the true and objective reality 2000 years from now. The geocentric paradigm, which was widely accepted until the 16th century, has been conclusively proven to be a grand pseudoscientific illusion.

Similarly, Germ Theory was the true and objective reality 500 years ago, remains the true and objective reality in the 21st century, and will continue to be the true and objective reality 500 years from now. The Miasma Theory, which was widely accepted until the 18th century, has been conclusively proven to be a grand pseudoscientific illusion.

In the same manner, **Componentology** has been the true and objective reality for at least 200 years, remains the true and objective reality today, and will continue to be the true and objective reality 200 years from now. In contrast, the existing theoretical foundation for Component-Based Software Engineering (CBSE), which has been widely accepted for many decades, is most certainly a grand pseudoscientific illusion.

Componentology constitutes the valid theoretical foundation that underpins every other engineering discipline—such as mechanical, aerospace,

and electronic engineering—where large products are constructed as *Component-Based Products (CBPs)*. Each CBP is designed and built by assembling multiple real, physically distinct components, with each component being a specific type of part that can be independently assembled, disassembled, and replaced.

FIG-1:Total Lines of Code for Monolith is 150K (After using every possible reusable library/API)	**FIG-2**: Total Lines of Code 50*3000 (=150K) About 50 Micriservices of average size of 3K
	
Monolith W/O Pluggable Components	Pluggable Components in the Monolith

Our research, spanning the past 25 years, has revealed that the existing theoretical foundation for software engineering is, in fact, a grand pseudoscientific illusion — a root cause of the infamous software crisis. Our patented inventions, which are based on the principles of Componentology, offer effective solutions to address this long-standing software crisis. This demonstrates that software engineering, like other engineering disciplines, can

be grounded upon the same valid and objective theoretical foundation—namely, **http://Componentology.org**—which enables the construction of large products as CBPs.

The following foundational principles of **http://Componentology.org/Fly/Booklet2.pdf** are indisputably true 100 years ago, remain the true and objective reality today, and will continue to be the true and objective reality 100 years from now:

1. **Foundational Truth (First Principle):** Any product qualifies as a real *Component-Based Product (CBP)* if and only if it is constructed by assembling multiple real components, as illustrated in FIG-2 (where each small oval shape represents a pluggable component).

2. **Auxiliary Basic Truth 1:** Any part qualifies as a real component for building real CBPs if and only if it can be assembled—i.e., multiple such real components are assembled or plugged in to form a CBP as illustrated in FIG-2.

3. **Auxiliary Basic Truth 2:** Any engineering discipline or paradigm qualifies as real *Component-Based Engineering (CBE)* if and only if it can design and build every large product as a CBP, by assembling or plugging in multiple real components, as shown in FIG-2.

4. **Subsidiary Basic Truth:** Components (e.g., devices such as CPUs, DRAM, hard drives, software modules, or other electronic devices) that

communicate or collaborate with each other by exchanging data or signals can be plugged in (e.g., into a printed circuit board or system board, shown by SoA in FIG-2).

Note: Components can be broadly categorized into two types:

i. *Pluggable components*: such as electronic components, which collaborate or communicate by exchanging data or events/signals and can be crafted for plug-in assembly; and
ii. *Moving components*: such as mechanical components, which are designed for assembly and motion within mechanical structures.

Software components belong to the category of *pluggable components*, as they communicate and collaborate through the exchange of data or events/signals.

It is deeply regrettable that the scientific and research communities continue to suppress or ignore these indispensable insights, despite their fundamental importance in achieving an objective scientific understanding—an understanding that forms the essential theoretical foundation required to meaningfully and effectively address the longstanding software crisis.