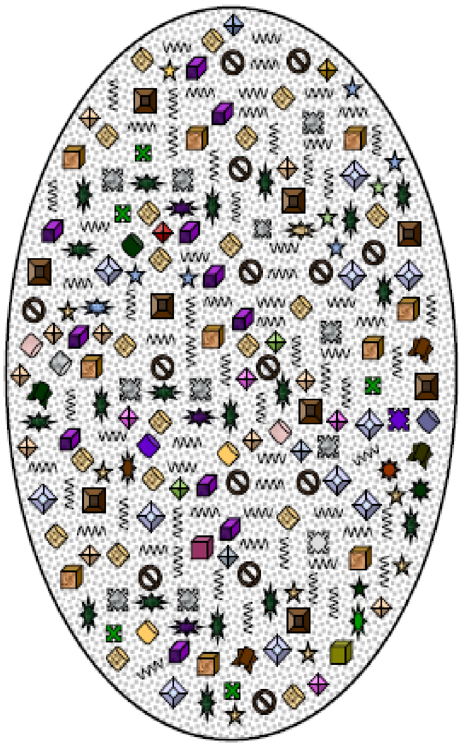
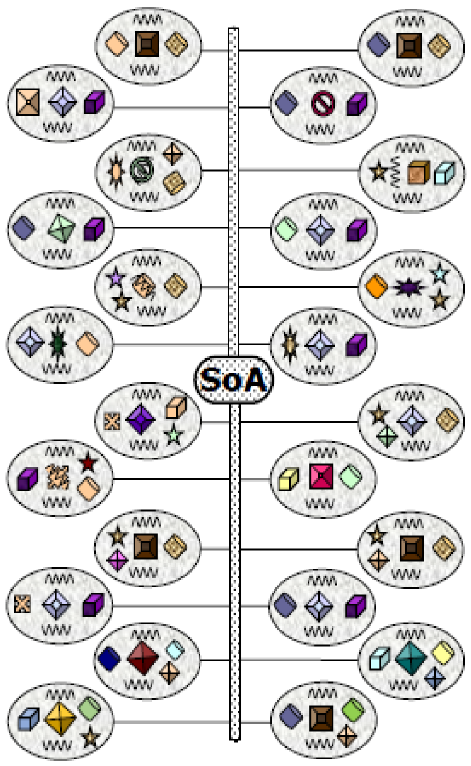
What is a CBP (Component Based Product)?

There will be a revolution in Software Engineering (by subverting the existing flawed dominant paradigm that is rooted in primordial, unproven, and flawed dogma) if the software community investigates facts and evidence scientifically to learn valid answers and descriptions, scientifically, honestly, and with integrity (by adhering to proven scientific principles), for these 2 simple questions in the context of all other engineering disciplines such as Mechanical, Electronic, Computer, Civil, & Aerospace:

1. What is a real CBP (Component-Based Product)? In other words, what is the structure and anatomy of CBPs (Component-Based Products)?

2. What are real Components (i.e. a very specific kind of parts) that are essential to building real CBPs?

<u>**FIG-1**</u>:Total Lines of Code for Monolith is 150K (After using every possible reusable library/API)	<u>**FIG-2**</u>: 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

I have been requesting real scientists who are honest and have integrity to validate my answers scientifically. It is the least general public expects from the scientific community. I am hoping that real scientists who are honest and having integrity validate my discoveries and patented inventions.

Understanding the following two simple facts scientifically subverts existing flawed dominant paradigm for Software Engineering by exposing myths in the foundation of the paradigm. Do these two simple facts require any proof: (1) No product can be a CBP if the product is not built by assembling multiple components (e.g. as illustrated in FIG-2). (2) It is essential to invent new kind of components (e.g. objects instances) that can be assembled, by inventing mechanisms and tools (e.g. such as a SoA as in FIG-2) for example, to build a product by plugging in components: *http://real-software-components.com/raju/Briefs/WhatIsStructureOfCBP.pdf*

Once valid answers and descriptions for CBPs and Components are found scientifically (without violating proven scientific principles), it is not difficult to make inventions that are necessary to build every large software products by plugging in multiple optimal-sized components as in FIG-2. My objective is to prove these 3 facts:

1. CBE (Component-Based Engineering) implies designing and building Component-Based Products, where a CBP (Component-Based Product) implies a product built by assembling multiple components as illustrated in FIG-2, and where the components are a very specific kind of parts that can be assembled.

2. Today, it is impossible to find even a single software product that is designed and built by assembling real software components as illustrated in FIG-2. Therefore, Software Engineering is not employing real CBE paradigm today, since real software components (i.e. a very specific kind of parts that can be assembled) are not yet known and Software Engineering is incapable of building CBPs.

3. It requires three kinds of inventions to transform Software Engineering from the inefficient non-CBE paradigm to the ten times more efficient real CBE paradigm that can design and build every software CBP as illustrated in FIG-2 by assembling multiple optimal-sized software components.

Transforming **From** *non-CBE-paradigm* **To** *CBE-paradigm*

There are two engineering paradigms, which are (**1**) CBE (Component-Based Engineering) paradigm that builds each large or complex product (e.g. airplane, car, or spacecraft) by assembling multiple smaller components, and (**2**) non-CBE paradigm implies building each large or complex product (e.g. buildings or skyscrapers) by using reusable ingredient parts (e.g. cement, steel, metals, concrete, alloys, chemicals, paint, tiles or plastic). Both paradigms use equivalent quantity of reusable ingredient parts (that are not conducive to be assembled). In case CBE-paradigm, each of the components is designed and built by using reusable ingredient parts and also each component is designed so that the component can be easily assembled and disassembled: http://real-software-components.com/raju/TwoKindsOfParadigms.pdf

Three inventions that are essential for transforming software engineering **From** (1) inefficient non-CBE paradigm (as illustrated in FIG-1) that only uses and composes reusable ingredient parts to build each product; **To** (2) about ten-times more efficient real CBE-paradigm (as illustrated in FIG-2) that can build each large software product by assembling multiple pluggable components, where each software pluggable component is built by using reusable parts or modules, include:

1. Methods and methodologies to partition each large product in FIG-1, into multiple optimal size self-contained modules in FIG-2,

2. Technologies, tools and mechanisms (e.g. communication interfaces that can be plugged-in or loosely coupled) to create and use real software components, and

3. Intelligent tools and mechanisms to automate various tasks and activities that are necessary to create, document and manage (e.g. redesign) communication code to plug-in software components, where the communication code allows collaboration between all the software components that are plugged in.