COMPONENTOLOGY

A new branch of hard science, aimed at understanding physical entities, to replace the existing junk science, which is fiction portraying fictitious entities.

TRY THESE ROUND WHEELS
HARD SCIENCE
HARD SCIENCE
NO THANKS
DAMN, SCIENTIFIC AUTHORITIES ARE BLIND TO THE OBVIOUS
THESE SQUARE WHEELS ARE WREAKING HAVOC ON OUR BACKS
JUNK SCIENCE
FICTION

Componentology is a valid hard science created to liberate software engineering research from the stranglehold of junk science and fake scientists, who are associated with an anti-science cult.

Creating & Utilizing True Virtual Components
— Based on Componentology

Componentology

Created by Raju Chiluvuri

A new branch of science for acquiring hard scientific knowledge to gain a comprehensive understanding of physical entities (e.g., Real Components & Real Component-Based Products), to invent their true virtual counterparts in software.

INDEX

Terminology & Abbreviations

BoK:	Body of Knowledge
CBP:	Component-Based Product
CBE:	Component-Based Engineering
CBD:	Component-Based Design
CBSE:	Component-Based Software Engineering
Componentology:	A new branch of hard science
Real component:	A specific kind of part that can be plugged in

Key Words:
Componentology,
Virtual Components,
Virtual CBPs, and
True CBE Paradigm.

Raju Chiluvuri,
Componentologist
CEO, Pioneer-soft, Inc
raju@componentology.org

Building true virtual CBPs in software by creating and utilizing true virtual components – Based on an objective understanding and scientific knowledge of Componentology.

1. Introduction to Componentology

Componentology is a novel scientific discipline created to systematically study, without violating the rules and principles of the scientific method, all facets of the reality of physical components and parts (e.g., their innate nature and essential properties); the anatomy, structure, design, and construction of physical CBPs (Component-Based Products); as well as the essential methods and mechanisms of real CBE (Component-Based Engineering) disciplines such as mechanical and electronic engineering which build physical products as CBPs.

"The scientific method is doing whatever it takes to not fool yourself into thinking something is true that is not, or into thinking that something is not true that is."

- Neil DeGrasse Tyson

The modern scientific method was established in the 17th century during the Scientific Revolution by pioneering scientists who bravely challenged the prevailing junk science of the day, which was the illusions of the geocentric paradigm, and paid for it by enduring pain and suffering. Since then, the scientific method has undergone continuous refinement and improvement. The scientific method comprises of a set of proven rules and principles for acquiring and enhancing our understanding of the physical world and reality through objective observations, experimentation, and empirical evidence.

The definition of real/hard science

Any discipline can be classified as a real or hard scientific discipline if it meets two essential conditions: (1) the discipline must scientifically study and understand the reality of physical entities (e.g., animals, trees, microbes, chemicals, physical products such as cars, airplanes, computers, and physical components used to build the products), where the physical entities cannot defy the laws of physics or nature; and (2) the discipline must objectively accumulate knowledge and insights based on valid empirical evidence and independently verifiable observable aspects of physical reality, without violating the proven rules and principles of the scientific method.

"Investigating or scientifically studying" implies the systematic accumulation of the scientific knowledge necessary to gain an objective understanding and insight about physical entities (which cannot defy the laws of physics or nature), without violating the principles of the scientific method.

Componentology is a hard science created to understand objective truth and reality by investigating or scientifically studying physical entities (e.g., physical CBPs, mechanisms of CBE, and Components) that cannot defy the laws of nature or physics, just like other established hard sciences that investigate or study the reality of physical entities. Zoology and Botany are considered hard sciences because they study the reality of physical entities such as animals and plants. Microbiology is another hard science that focuses on the study of microorganisms such as bacteria, viruses, and parasites. Similarly, biochemistry is a hard science dedicated to studying chemical processes within living organisms.

Objective truth is always true/factual regardless of whether anyone believes in it. The methods, principles, and tools of the scientific method are uniquely conceived, refined, and perfected to seek out, test, and validate to establish truths/facts objectively.**

** http://Componentology.org/Vid/4

Background of Componentology: Science entails understanding the physical world (not just the natural world) objectively through observations and experiments, with an open mind. Botany explores the reality of plants, while zoology delves into animals. Similarly, Componentology scientifically investigates physical entities like Components, CBPs, and CBE. However, the current understanding, shaped by the prevailing climate of opinions, encompasses false and baseless descriptions and concepts for components and CBE, rooted in flawed assumptions made over 54 years ago during the nascent phase of software engineering. Thus, viewing Componentology solely through the lens of software and the prevailing climate of misconceptions leads to distorted perceptions and biases.

Componentology must emancipate itself from the prevailing mythology of software engineering, where mythical entities are masqueraded as Components and CBE. Before the onset of this climate of misconceptions, which began with the false assumptions made about components and CBE in the 1968 NATO Software Engineering Conference,** a vastly superior form of Componentology could have emerged, perhaps during the 1950s, untainted by the widespread falsehoods and misconceptions that have disseminated swiftly since the 1968 NATO Conference. Given the pervasive climate of misconceptions today about components and CBE, the primary challenge in scientific research lies in maintaining an open mind while objectivity analyzing all the relevant data, observations, and evidence. It is vital to question whether qualified scientists from the 1950s, unexposed to the prevailing climate of misconceptions, would reach the same conclusions based on the available evidence, data, and observations for each description or concept within Componentology.

The goal of Componentology is to revolutionize software by replacing existing junk science with real science. Achieving this goal necessitates exposing fake scientists who vehemently defend the existing junk science, having already been indoctrinated into it or contributed to it, and are hostile to real science that challenges the false beliefs in the existing junk science.

2. Two Vital Layers/Parts of the Research Ecosystem

The two vital parts of the research ecosystem are (i) Layer-1: "Basic (or pure) scientific research" which is to acquire and accumulate scientific or theoretical knowledge to create and expand the pure scientific BoK (Body of Knowledge) that is vital for gaining deeper understanding and scientific insights to address complex, not yet solved problems in science and engineering, and (ii) Layer-2: "Applied scientific research (or engineering research)" which relies on the pure scientific BoK available in Layer-1 to create or invent useful things or to find solutions to address problems in science and engineering.

The basic scientific BoK in Layer-1 provides the theoretical foundation to conduct engineering research in Layer-2. It is vital to acquire and rely on valid scientific BoK, which must be objective, testable, and falsifiable and must be based on objective valid evidence. If the BoK in Layer-1 is flawed/invalid, it is impossible for engineering research in Layer-2 to successfully invent useful things/solutions (e.g., to address any problems) by relying on such flawed BoK (i.e., filled with flawed beliefs, concepts, or descriptions) in Layer-1.

** http://Componentology.org/Ref/17

Layer-2: Research in Applied Science or Engineering

Layer#2: Applied Scientific (or Engineering) Research, whose objective is to invent useful things and solutions by relying on relevant scientific knowledge in the theoretical foundation in Layer-1. See below:

Layer#1: Basic/Pure Scientific Research, whose goal is to accumulate purely scientific and objective theoretical knowledge comprising of first principles, theories, descriptions, concepts, and methods/mechanisms.

Layer-1: Theoretical Foundation or Basic Science

Table-1: The knowledge and understanding gained from basic research (Layer-1) serve as the essential ingredients and guiding force behind nearly all research activities in applied science or engineering (Layer-2).

Definition of "BoK for CBSE": "BoK for CBSE" stands for "BoK (Body of Knowledge) that is used as (and/or necessary) theoretical foundation (i.e., in Layer-1) for conducting applied research in CBSE (Component-Based Software Engineering)", where the BoK comprises various pieces of knowledge about structure and anatomy of CBPs (Component-Based Products); the essential properties, mechanisms, and nature of physical components that are essential building blocks to build each product as a CBP; as well as the methods and mechanisms of real CBE (Component-Based Engineering).

The theoretical foundation (in Layer-1) essential for conducting engineering research in CBSE and CBSD encompasses numerous descriptions, concepts, theories, and explanations for understanding various kinds of parts as well as real components (e.g., their nature, functioning, and essential properties), the anatomy, structure, design, and construction of ideal CBPs (Component-Based Products), as well as the methods and mechanisms of ideal CBE (Component-Based Engineering).

It is impossible to even contemplate or start conducting engineering research in CBSE/CBSD without having or creating some kind of knowledge or understanding, whether it be good or bad, about components and CBPs, as well as the methods and mechanisms of CBE. It is impossible to conduct engineering research successfully in Layer-2 if the available knowledge or understanding is bad (i.e., junk science), and the research efforts end up in crisis.

Nathan Myhrvold, former CTO of Microsoft: Technological progress cannot continue without the input of basic research and the conceptual breakthroughs it makes possible. To reduce knowledge to practice, one must have the knowledge in the first place. Science is the raw material (in Layer-1) that applied research and engineering (in Layer-2) refine into their products. **

In essence, initiating engineering research efforts is impossible without having or creating at least some knowledge, such as descriptions and concepts for components and CBE. If nothing is known about components and CBE, how is it possible to even initiate applied research on components and CBSE without first attempting to learn about them? Furthermore, if the knowledge acquired about them is flawed, the research efforts cannot succeed.

The 1968 NATO Software Engineering

** http://Componentology.org/Ref/18

Conference is widely recognized for codifying and establishing foundational descriptions and concepts for components and CBE. This foundational knowledge, comprising descriptions and concepts, served as a reference and groundwork for the development of subsequent ideas and concepts, as well as for conducting applied research in CBSE. Since conducting applied research in Layer-2 without any theoretical knowledge in Layer-1 is impossible, this knowledge was utilized in Layer-1. There exists a symbiotic relationship between applied research and basic research, where advancements in one field open new paths for the other.

Applied research contributes to expanding basic research not only by identifying gaps in the existing theoretical foundation to formulate new research questions but also aids by providing empirical evidence, tools, technologies, or mechanisms to address those gaps. Similarly, basic research offers new conceptual insights for experimentation and exploration in applied research, thereby establishing a cycle of innovation and advancement. If important parts of the foundational knowledge in Layer-1 are flawed, it imposes a severe burden on this cycle of innovation, ultimately bringing it to a halt.

3. Three Material Facts for the Lawsuit Against NSF.gov

It is an indisputable fact that in the context of research in any field of science, applied science, or engineering: Accumulating and relying on junk science is far more toxic and harmful than consuming slow poison or deadly carcinogenic foods. When a deadly issue is uncovered in any vital field such as software, the issue must be addressed immediately with a sense of urgency akin to wartime efforts.**

It is an undeniable fact that the Body of Knowledge (BoK) within the existing theoretical foundation for software engineering is unscientific junk science. The BoK, which includes descriptions, concepts, theories, and explanations about components, Component-Based Engineering (CBE), and CBPs (Component-Based Products), passes almost every proven test for junk science with flying colors, which confirms most of the BoK is junk science. ***See Section 16 to 21 & 28 on Pages 30 to 39 & 48 respectively.***

Ignorant, unqualified, or racist scientists at NSF.gov have been employing unlawful evasive tactics to ignore, suppress, or even sabotage evidence that proves the use of toxic junk science. Such actions amount to gross negligence and are unlawful, especially considering the availability of valid real or hard science (e.g., Componentology) to replace toxic junk science. Componentology is undeniably a valid real or hard science since it satisfies the definition of real/hard sciences.

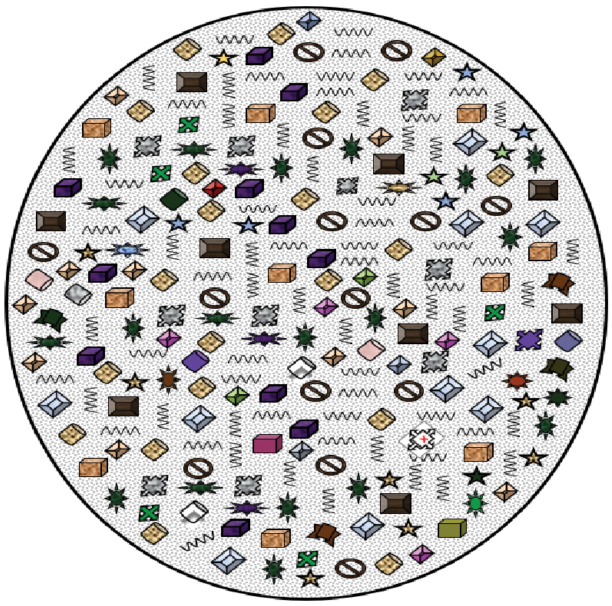
To win the lawsuit, it is essential to prove beyond doubt that most existing components or pieces of the BoK, encompassing all known descriptions, concepts, explanations, or theories about components, CBE, and CBPs, consistently fail proven tests or rules, conclusively confirming most of them to be junk science. Furthermore, it is essential to demonstrate beyond doubt that Componentology is a valid real or hard science, capable of effectively addressing the toxic effects of the existing junk science when the existing junk science is replaced by Componentology.

Fatal consequenses of false foundational beliefs or first principles

** http://Componentology.org/Ref/9
http://Componentology.org/Ref/10

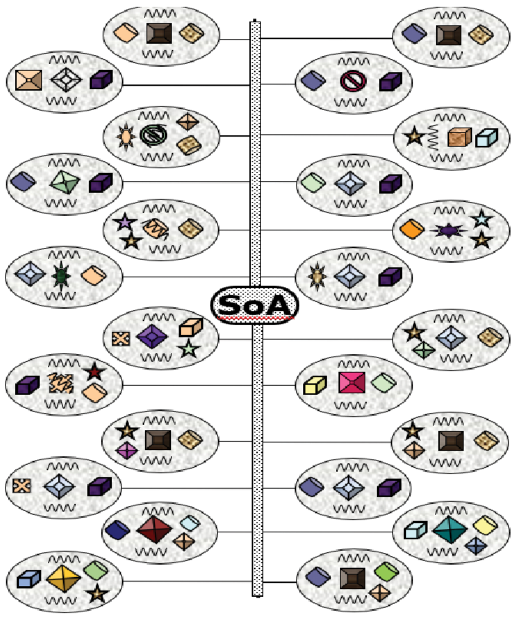
4. Exhibit A: Foundational Principles

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



Monolith W/O Pluggable Components

FIG-2: Total Lines of Code 50 x 3000 = 150K About 50 Microservices with average size of 3K



Pluggable Components in the Monolith

At the foundation of any valid scientific discipline lie one or more foundational truths or postulations, known as the first principles. In any scientific field, it is a fatal mistake to base these crucial or core first principles** on flawed beliefs or myths, such as the notion that the Earth is at the center. Doing so inevitably leads the scientific discipline into crisis, resulting in it being regarded as a form of "junk science," akin to the geocentric illusions prevalent in the 16th century.

The foundational truths or postulations outlined below, referred to as first principles and foundational theories within Componentology, are indisputably correct and verifiable scientific observations within the broader context of all other engineering disciplines that employ CBE including mechanical, automobile, aerospace, computer, and electronic disciplines. Each of these disciplines can be utilized in the design and construction of each large or complex product such as cars, airplanes, computers, spacecraft, electronic machines like telecom switches, medical equipment such as MRI machines, and other large machinery for factories, as a CBP (Component-based Product).

Exhibit-A1: Basic axioms and foundational principles

Foundational Truth (also known as First Principle): Any product can be a real-CBP (Component-Based Product) if and only if the product is built by assembling multiple real components, as illustrated in FIG-2 (each small oval shape represents a pluggable component).

Auxiliary Basic Truth: Any part can be a real component for building real CBPs if and only if the part can be assembled (i.e. multiple such real components are assembled or plugged in to build a real CBP as in FIG-2).

Auxiliary Basic Truth: Any engineering discipline or paradigm can be real CBE if and only if the discipline can design and

** http://Componentology.org/Ref/9

build every large product as a CBP (i.e. by assembling or plugging in multiple real components, as illustrated in FIG-2).

Subsidiary Basic Truth: Components (e.g. devices such as CPU, DRAM, hard drive, or other electronic products) that collaborate and communicate with each other by exchanging data and signals can be plugged in (e.g. into a printed-circuit-board or system-board shown by SoA in FIG-2).

P.S: Components can be broadly categorized into two types: (i) Pluggable components, such as electronic components that collaborate or communicate with each other by exchanging data or events/signals and can be crafted to be pluggable; and (ii) Moving components, such as mechanical components that are crafted for assembly, enabling movement in various mechanisms, and assembled into mechanisms that allow their movement. Software components belong to the category of pluggable components, as they collaborate or communicate with each other by exchanging data or events/signals.

Henceforth, the above four foundational principles will be referred to as Exhibit A1. These statements outline the general design and structure of CBPs and the mechanisms of CBE, analogous to how the three laws of Kepler describe the general structure, orbits, and motions of our planetary system. Please refer to Section-11 on page 20.

Exhibit A2: Important Facts. Please refer to Section-10 starting on page 19

Exhibit A3: Research Question. Please refer to Section-27 starting on page 47

5. Exhibit B: The Existing Junk Science

The existing "BoK for CBSE" in Layer-1 encompasses a vast collection of beliefs, myths, and misconceptions (aka mythology), which have accumulated since the year 1968 based on false assumptions made in the 1968 NATO Software Engineering Conference. Software researchers have been relying on this ever-expanding flawed BoK for CBSE in Layer-1 to conduct applied research in software engineering in Layer-2, which is the root cause of the infamous software crisis. I created Componentology and utilized it (instead of the existing BoK for CBSE) in Layer-1 to conduct applied research in Layer-2, and this approach helped me secure 8 US patents for inventions that can effectively address the infamous software crisis and notorious spaghetti code.

This booklet aims to furnish verifiable evidence, data, and proof that conclusively demonstrates beyond any doubt that the existing BoK for CBSE (in Layer-1) is fundamentally flawed (i.e., junk science). Exhibits B1, B2, B3, & B4 that are listed below comprise a few quintessential representative samples or specimens of the existing BoK for CBSE (e.g., descriptions and concepts) in Layer-1, and researchers have been conducting applied research in Layer-2 by relying on the junk science in Layer-1**.

Exhibit B1: This provides many inconsistent or contradictory and ambiguous descriptions for so-called components, where the descriptions are merely opinions of renowned thought leaders or prominent committees. ***Section-12 starting on Page 22 has first two pages of the descriptions at: https://wiki.c2.com/?ComponentDefinition***

Exhibit B2: ***Section-13 starting on Page 24 has the cover-page and first page of the most popular books (title: "Component Software" by Szypersky, et al) on software components http://real-software-components.com/raju/***

** Refer to Layer-1 & Layer-2 in Table-1 on page 6

ExhibitB2_BookForCBE.pdf

Exhibit B3: This provides yet another group of inconsistent, contradictory, and ambiguous descriptions for so-called components. ***Section-14 starting on Page 26 has the first two pages of the opinions expressed or descriptions suggested about so-called components at: http://real-software-components.com/raju/WhatIsComponent2.pdf***

Exhibit B4: This paper "Twenty-eight years of component-based software engineering" authored by Vale, Crnkovic, et al. This paper studied and analyzed 1231 research papers published on CBSE over a span of 28 years from 1984 to 2012. The introduction section of this paper confirms that McIlroy (1968 NATO Conference) shaped foundational assumptions for the research papers on CBSE. ***Section-15 starting on Page 28 has the first page of this paper. http://www.es.mdh.se/pdf_publications/4272.pdf***

The initial paragraphs of nearly every introductory chapter in textbooks introducing a science or engineering discipline impart the foundational truths or axioms, also known as first principles of the discipline. Therefore, the best place to find the foundational axioms, or crucial first principles, underlying the existing dominant paradigm for CBSE is to delve into the first chapter of each of the renowned books on software components. Exhibit B2 stands out as one of the most popular books on software components, which makes the false statement "Components are for Composition and are Reusable." However, the stark reality contradicts this assertion, as evidenced by the majority of components (as depicted in FIG-2) for ideal physical CBPs being custom-designed to precisely fit and function optimally for a single product model. Consequently, they don't need to be reusable, and they must not be composed but rather must be pluggable.

Any discipline purporting to be scientific risks being labeled the epitome of junk science if its Body of Knowledge (BoK) is inundated with fragments of knowledge, such as descriptions or concepts, that contradict or are inconsistent with each other. The prevailing theoretical knowledge encompasses descriptions and concepts regarding mythical or fictitious entities ***(see Exhibits B1 on page 22 & B3 on page 26)***, misleadingly labeled as Components.

This theoretical knowledge (comprising descriptions & concepts) lacks any basis in verifiable or valid evidence and is devoid of objectivity or scientific rigor. Consequently, it has devolved into an internally inconsistent system replete with untestable beliefs, numerous gaps, unscientific concepts, and ambiguous descriptions, which fail to adhere to the fundamental principles of the scientific method (e.g., testable or falsifiable based on data and evidence), which are vital for them to qualify as valid theoretical knowledge, thereby rendering them as examples of unscientific junk science.

If a BoK contains numerous descriptions, concepts, and theories but is not attempting to describe any real entity, then by process of elimination, it must be describing fictitious entities. In essence, a BoK that seeks to describe entities must either describe real entities, which exist in the physical world and can be observed and verified, or fictitious entities, which are products of imagination or conceptualization. There is no other possibility because descriptions, concepts, and theories inherently pertain to some form of entity, whether real or fictitious.

Today, there are numerous inconsistent descriptions of so-called software components, as evidenced in Exhibits B1 & B3. Are these descriptions attempting to accurately depict real components used in engineering disciplines like mechanical, aerospace, or electronic engineering to design and construct CBPs? If these descriptions, such as those in

Exhibits B1 & B3, are not making any attempt to describe real components (e.g., based on valid observations and verifiable evidence), then they must instead be attempting to describe some form of fictitious entities.

It's not surprising that no one else in the software industry possesses knowledge about real components and CBPs. There's no evidence to suggest that anyone else in the software field has even tried to compile a Body of Knowledge (BoK) containing descriptions, concepts, and theories to accurately understand real entities like physical components and CBPs, along with mechanisms for CBE. Consequently, the current understanding of components and CBPs is centered on fictitious entities insidiously masquerading as components and CBPs, rather than on authentic ones.

Exhibits B1 to B4 epitomize the existing BoK for CBSE, consisting of misconceptions, myths, and unfounded beliefs that have been shaping the perceptions and understanding of the software community. These perceptions and understanding stand in clear contradiction to the reality that can be shaped by the BoK accumulated for Componentology ***(See Section-1 on page 4).***

Several sections in this booklet provide evidence, references, and proof to establish the fact that the existing "BoK for CBSE" is junk science—a mythology about mythical entities insidiously and deceptively labeled as components. These mythical entities cannot be considered components for CBE by any stretch of the imagination, so attempting to use them as components is an egregious mistake. These sections include:

Please See Section 23 to 26 on Pages 40 to 47; Section 21 on page 38 & Section 28 on page 48.

6. Short Summary of Lawsuit Against NSF.gov

Allegation: Blindly relying on untested and unproven toxic junk science as the theoretical foundation for conducting applied research in software engineering — as an unequivocal cardinal rule — represents a fatal error with potentially deadly consequences. Within the realm of scientific disciplines, depending on or teaching toxic junk science is akin to administering a lethal, slow poison.

Corpus of Evidence: Most of the booklet and its sections are dedicated to providing overwhelming evidence that the existing theoretical foundation constitutes toxic junk science, a mythology featuring mythical entities insidiously disguised as components. *Please refer to Sections 16 to 21 from page 30.*

Violations of the Laws: Junk science has already been implicated in thousands of deaths or injuries, and the toll continues to mount until effective measures are taken to address it. Despite clear and repeated warnings, the promotion and defense of junk science by the congressionally designated entity NSF, both in professional practice and education, including the tolerance of its dissemination to impressionable software students, constitutes numerous legal infractions. These violations encompass gross negligence under the Consumer Product Safety Act, defiance of congressional mandates, and contravention of the President of the USA's policy on scientific integrity — amounting to ethical breaches and scientific misconduct. Forcefully indoctrinating unsuspecting and impressionable software students into junk science is tantamount to feeding them slow poison or carcinogenic foods, resulting in their transformation into intellectually compromised brain-dead fake scientists or zombie professionals.

The definition of the 'scum of the scientific world' is any scientist or reviewer in a position of responsibility to advance science and technology who fiercely justifies blatant violations of scientific principles, such as utilizing mythology as the theoretical foundation and deliberately suppressing crucial evidence or data. These individuals lack integrity and ethics, posing the greatest obstacle to scientific progress. Dealing with such malevolent or dishonest actors in positions of power outside the courts is nearly impossible.**

The 'scum of the scientific world' strives to preserve the prevalent junk science by doing anything they can to breach scientific rigor and integrity. Scientific rigor entails strict adherence to the principles and methods of the scientific method in conducting research, ensuring the reliability, reproducibility, and validity of results. In other words, scientific rigor means implementing the highest standards and best practices of the scientific method and applying them to one's research, all in pursuit of discovering the truth.

Galileo's 400-year-old quote, "By denying scientific principles, one may maintain any paradox," remains profoundly insightful and enduring. In the 21st century, it is impossible to sustain any paradox or mythology as a theoretical foundation (in Layer-1). The only exception to this rule is the 'scum of the scientific world' who fiercely defend blatant breach of scientific integrity and rigor. These violations may include deliberately suppressing crucial evidence or data, using false assumptions or baseless beliefs as core first principles or foundational axioms, and viciously ostracizing anyone who tries to challenge or validate (through testing and falsification) their dogma or axiomatic beliefs.

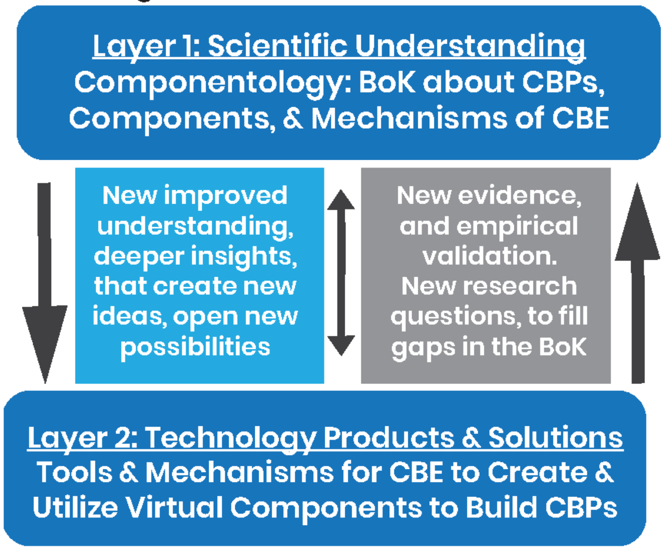
The numerous inconsistent descriptions and opinions found within Exhibits B1 & B3 are attempting to narrate or elucidate certain entities. Can anyone confirm whether they intend to describe real or fictitious entities? If indeed their purpose is to describe real entities, what is the commonly used name or term to refer to those entities?

Incorporating descriptions and concepts about fictitious entities doesn't inherently render the BoK of any discipline junk science. The logic and reasoning of axiomatic systems or formal sciences such as mathematics can effectively handle any Body of Knowledge (BoK) filled with descriptions and concepts for fictitious entities, provided that the descriptions and concepts in the BoK are logically consistent with each other, and each description or concept is unambiguous, precise, and can be rigorously validated through testing and falsification. However, the BoK for any discipline must be considered junk science if most of the descriptions and concepts within it are subjective, ambiguous, and inconsistent with each other. Additionally, if none of these descriptions or concepts are conducive to validation through testing and falsification, it severely undermines the integrity and renders the BoK an unreliable junk science.

7. Symbiotic Relationship Between Understanding & Technology

In science and technology, 'to understand' is a powerful concept. It drives all our activities. It forms the foundation for all technological development. There exists a symbiotic relationship between fundamental understanding and the technologies created based on that understanding. Basic understanding and technological

** http://Componentology.org/Vid/2 & http://Componentology.org/Vid/3 development represent two sides of the same coin. Just as breakthroughs in basic research lead to significant advances in technology, technological developments often yield new directions (e.g., new insights through empirical validation, and new insightful research questions) in basic research, resulting in a deeper understanding of the fundamentals, which serves as fertile ground for new technological innovations.


Layer 1: Scientific Understanding
Componentology: BoK about CBPs, Components, & Mechanisms of CBE
New improved understanding, deeper insights, that create new ideas, open new possibilities
New evidence, and empirical validation. New research questions, to fill gaps in the BoK
Layer 2: Technology Products & Solutions
Tools & Mechanisms for CBE to Create & Utilize Virtual Components to Build CBPs

Even talented painters cannot draw a picture of an elephant if they do not know what an elephant is (i.e., whether it's a tree, bird, animal, fruit, or flower). Similarly, it is impossible to build each software product as a CBP (e.g., by creating and utilizing real components) without understanding components and CBPs. Today, no one else in the software world comprehends the reality of components, the anatomy of CBPs, and the mechanisms of CBE. What is understood today pertains to fictitious entities misleadingly labeled as components, which constitutes an insidious and toxic junk science—a fatal mistake in science and technology with dreadful consequences.

Componentology is designed to accumulate the BoK necessary for scientifically understanding the objective reality of components, CBPs, and mechanisms of CBE. This understanding is crucial for inventing tools and technologies to create and utilize software components for constructing each software product as a CBP. These patented tools and technologies are invaluable for gaining new insights through experimentation and rigorous empirical validation, and they have assisted in formulating insightful research questions to expand Componentology, thereby fostering new technological innovations.

Virtual entities aim to closely mimic real-world counterparts in digital environments, while fictitious entities involve imaginative or invented mythical entities that diverge from reality for purposes such as speculation, wishful thinking, and illusions. The current theoretical foundation, which includes numerous descriptions, concepts, and explanations about so-called components and CBE, is mere fiction and a junk science. This fact should be apparent to any software researcher familiar with the multiple existing speculative descriptions (which are ambiguous, subjective, and inconsistent with each other) for fictitious entities misleadingly labeled as components.

The entities known today in software as components, CBE, and CBPs are fictitious or mythical entities that are being deceptively masqueraded and misleadingly disguised as virtual imitations or counterparts of real-world entities. The existing CBE for software has been using junk science about fictitious entities, and the aim is to replace the fictitious entities with virtual entities based on valid science about physical entities.

Componentology provides precise and comprehensive scientific knowledge about real entities for creating and utilizing their virtual counterparts. On the other hand, intellectual battles have been raging for decades to find acceptable or workable descriptions, concepts, and mechanisms for creating and utilizing the fictitious entities deceptively labeled as components.

"Understanding" (known as basic theoretical knowledge/ingredients in Layer-1) forms the very foundation of engineering research**

** Refer to Layer-1 & Layer-2 in Table-1 on page 6

(in Layer-2) to create useful things or address unsolved technological problems. No one in the software world knows or understands the reality of physical components and CBPs, to be able to create virtual entities for the respective real entities, for instance. It is impossible to create virtual entities for any real entity without comprehending the essential properties, appearance, features, and functionality of the real entity.

8. The "BoK for CBSE" is Fiction & Junk Science

In the context of literature, such as a book or article, it adheres to the classic definition of fiction if the primary focus of the literature lies in depicting or portraying imaginary or mythical entities. Essentially, any field can be classified as fiction if its BoK predominantly consists of descriptions and concepts that depict or portray entities which are not physical or real.

The BoK for the existing theoretical foundation of software engineering contains numerous descriptions and concepts, none of which pertain to depicting physical or tangible entities, but instead, it portrays fictitious entities deceptively referred to as components and CBE. The existing descriptions and concepts for so-called components and CBE contradict obvious evidence and observable aspects of real entities, such as physical components or CBE, demonstrating that they are not attempting to depict or portray the respective real entities.

Consequently, it is reasonable to infer that the theoretical foundation primarily revolves around depicting or portraying imaginary entities, whether fictitious or mythical. The current 'BoK for CBSE' lacking sturdiness provides a weak, uneven, and unstable foundation, encompassing ideas, concepts, or a belief system that fails to withstand scrutiny or lacks a solid logical, evidential, or rational basis.

The current theoretical foundation in Layer-1 for software engineering is filled with ideas, descriptions, concepts, and beliefs that, admittedly and evidently, have not been accumulated to comprehend and portray real or tangible components, CBPs, or mechanisms of CBE. Instead, it has been attempting for decades to depict or portray fictitious entities falsely and misleadingly labeled as components, or CBPs.

There is no need for expert witnesses to prove this fact: Any discipline must be mythology or junk science if its BoK is filled with inconsistent and baseless ideas, descriptions, concepts, and received beliefs that do not aim to understand or depict real or tangible entities, but instead focus on illustrating or portraying fictitious or mythical entities.

Only fools deny this fact: Real hard science depicts real entities; however, it becomes fiction if it portrays fictitious entities.

It is an undeniable fact: Most, if not all, of the existing inconsistent and baseless ideas, descriptions, concepts, and received beliefs in the BoK (such as those in Exhibits B1 to B4**) are not for understanding or depicting real components or CBPs. Instead, they have been accumulated for decades to portray and illustrate fictitious entities that are deceptively and misleadingly referred to as components, CBPs, or CBEs.

Software researchers have been fooling themselves and the rest of the world into believing that they understand components and are creating and utilizing virtual components in practicing real CBE for software. However, the reality is that the existing

** Sections 12 to 15 between pages 22 to 29

BoK is focused on certain kinds of fictitious or mythical entities, and the researchers have no knowledge of real and tangible components or CBPs. It is impossible to create and utilize virtual components in software to build each software product as a virtual CBP without creating the BoK of Componentology.

Distinctions Between Fictitious Entities Vs Virtual Entities: Comprehending the distinction between 'virtual entities' and 'fictitious entities' confirms that the so-called components for software known today are fictitious entities. This is because they are derived from descriptions and concepts for fictitious entities that are misleadingly or deceptively referred to or labeled as components.

Virtual entities, creations of digital technology, are designed to mimic real entities, aiming to resemble and function as their physical counterparts. Therefore, creating virtual entities demands a valid understanding of the essential properties, mechanisms, anatomy, and functioning of these physical counterparts, necessitating a nuanced grasp of the underlying principles or mechanisms governing the entities being represented.

Fictitious entities are characters, concepts, or entities invented by humans to construct narratives or conceptual frameworks, departing from reality to explore hypothetical scenarios, possibilities, mythical realms, or convey symbolism. Fictitious entities are crafted without considering evidence, observable or known facts, and a nuanced understanding of the underlying principles and mechanisms governing the entities they may deceptively appear or pretend to portray.

The fundamental distinction between virtual entities and fictitious entities lies in their adherence and consideration given to reality. Virtual entities, such as digital avatars or simulations, must aim to resemble and function as real entities, while fictitious entities, exemplified by so-called components in software, have disregarded and diverged from reality by ignoring essential properties, function, and observable aspects of real components.

The so-called components known today in software are indeed fictitious entities and not virtual entities, as they make no attempt to resemble or function as real components by understanding their essential features and function. The descriptions and concepts for these so-called components are created by disregarding evidence, observable or known facts, and academic principles such as validation and consistency.

Today, no known type of existing components resembles or functions like real components, so they must be fictitious entities and not virtual entities. There is no evidence to show that anyone else has attempted to understand objective reality (e.g., by creating a hard science Componentology or its equivalent) in order to build virtual counterparts in software.

Isn't it impractical to draw a few pictures that resemble an entity (e.g., an animal, bird, tree, fruit, or flower) in various perspectives without understanding its essential and distinguishing features and appearance? Similarly, to create a virtual entity that portrays a real entity (e.g., a component or CBP) in various perspectives, it is essential to understand its essential and distinguishing features and functionality.

See sections 16 to 19 from page 30: It is junk science if the literature for fiction is filled with contradictory descriptions, ideas, and received beliefs as in Exhibits B1 to B3. Fiction can be a formal science or an axiomatic system if its BoK comprises of precise and unambiguous pieces of knowledge that are **"consistent"** with each other.

The proof to establish the fact that the 'Existing BoK for CBSE' is junk science has two parts, which together provide illustrations to prove beyond any doubt that the 'Existing BoK for CBSE' is junk science:

Illustrating the distinction between hard science and fiction (see paragraph below).

Illustrating the distinction between trustworthy BoK of academic disciplines and untrustworthy, unscholarly, and unacademic junk science *(see Section 28 on page 48)*.

The distinction between hard science and fiction lies in the BoK of the respective disciplines. The BoK of any hard science comprises of multiple pieces of knowledge such as objective descriptions, concepts, theories, and explanations, which are concerned with real or physical entities governed and constrained by the laws of physics or nature. Each piece of knowledge should have been acquired without violating established rules and principles of the scientific method. On the other hand, the BoK of fiction encompasses baseless speculative ideas, subjective descriptions, concepts, and unfounded beliefs about fictitious or imaginary entities. These entities are not constrained by the laws of physics or nature and are unconcerned with principles of science or logic.

No one in the software world can disprove these facts: The descriptions in Exhibits B1 and B3 neither describe any known kind or class of real entities nor serve to create any kind of virtual entities of any class or category of real entities. Instead, the descriptions in Exhibits B1 and B3 evidently describe fictitious entities. It is misleading and deceptive to label these fictitious entities as components in software, which has been giving a misleading and false impression that the so-called components in software are virtual entities for real components.**

9. Transforming a Product from Non-CBP to Real CBP

Two engineering paradigms exist (1) the CBE paradigm, which constructs each large or complex product (e.g., airplane, car, or spacecraft) as a CBP by assembling multiple smaller components as in FIG-2, and (2) the non-CBE paradigm, which involves building each large or complex product (e.g., buildings or skyscrapers) using reusable ingredient parts (e.g., cement, steel, metals, concrete, alloys, chemicals, paint, tiles, or plastic). Both paradigms may utilize an equivalent quantity or number of reusable ingredient parts, where the ingredient parts are not conducive to assembly.

In the CBE paradigm, the product is divided into multiple self-contained modules that can be built as loosely coupled pluggable components. Each component is then designed, constructed as pluggable, and tested individually using reusable ingredient parts. Once all the pluggable components are built and tested, the CBP is assembled by plugging in these components as depicted in FIG-2 on page 8.

In the case of non-CBP, each product is designed using various types of reusable ingredient parts (depicted using various shapes in FIG-1 on page 8), such as cement, steel, plastic, wood, leather, chemicals, bricks, nickel, acid for lead-acid car batteries, paints, silicon, metal-oxides for IC-chips, metals, and alloys, among others.

In the case of CBP, each product is built by plugging in multiple pluggable components as a CBP, as illustrated in FIG-2, where each component can be easily replaced. Every elementary pluggable component is designed and constructed using various types of

** Exhibits B1 & B3 in Sections 11 & 13 on pages 22 & 26

reusable ingredient parts such as steel, plastic, nickel, acid for lead-acid car batteries, paints, silicon, metal-oxides for IC-chips, glass, cloth, metals, and alloys.

Each reusable ingredient part in a component (shown as a small oval shape) is depicted using various shapes as in FIG-2. on page 8. The CBP in FIG-1 and each component in FIG-2 are constructed using ingredient parts, where many of these ingredient parts may be reusable parts sourced from third-party vendors. It is pertinent to note that both CBP in FIG-2 and non-CBP in FIG-1 may utilize an equivalent number and quantity of reusable ingredient parts (which are depicted using various small shapes). Multiple elementary components may be used to build container components, which can be used as sub-components to create larger container components, and so on.

The most significant invention in the history of engineering, to increase productivity, sustainability, and quality, is the very specific kind of part (known as a component) that can be assembled and disassembled. Software engineering has not yet invented such virtual components, many of which can be plugged in to build each software product as a CBP as in FIG-2 on page 8.

Three essential inventions for transforming software engineering from (1) the inefficient non-CBE paradigm, which only uses and composes reusable ingredient parts to build each product as a non-CBP as illustrated in FIG-1, to (2) the approximately ten times more efficient real CBE paradigm, which can construct each large software product as a CBP as illustrated in FIG-2 by assembling multiple pluggable components, are as follows:

Methods and methodologies to partition each large product in FIG-1 on page 8 into multiple optimal-sized self-contained modules in FIG-2 on page 8.

Technologies, tools, and mechanisms (e.g., communication interfaces to facilitate loose coupling or plugging in) to create each self-contained module in FIG-2 as a pluggable component.

Intelligent tools, technologies, and mechanisms to automate various tasks and activities necessary for creating, documenting, and managing (e.g., redesigning) communication code, which enables collaboration between all the pluggable software components in FIG-2.

It is not hard to imagine creating every software product as a virtual CBP, as illustrated in FIG-2; however, achieving this objective is impossible without inventing the necessary tools, technologies, and mechanisms to create and utilize multiple virtual pluggable components. Each pluggable component utilized to construct the CBP is depicted in FIG-2 by smaller oval-shaped images. Componentology has provided an indispensable theoretical foundation for enabling such essential innovations. Virtual pluggable components, capable of being assembled to build virtual CBPs, hold comparable value in enhancing the productivity, quality, and sustainability of the virtual CBPs when compared to real components used in constructing and maintaining large and intricate physical CBPs.

The patents granted to us by USPTO, so far, include a GUI-API capable of building real software components that can be plugged in to build complex component hierarchies for GUI-CBPs (Patent Nos. 7,827,527; 7,840,937; & 8,527,943), patents relating to real software components to build ideal CBPs (Patent Nos. 8,392,877; 8,578,329; & 9,058,177), and patents for a CASE-tool (e.g., virtual system-board SoA in FIG-2) to automatically assemble pluggable components (10,949,171 & 11,275,567).

9.1. Brief Executive Summary about Fictitious & Virtual Entities

Today, no one in the software world knows valid descriptions and concepts for Components and CBPs (Component-Based Products), and almost everything known today about Components, CBPs, and mechanisms for CBE is fiction, akin to junk science.

The software world has been creating fictitious entities based on the current, widely prevalent, deeply entrenched, junk scientific paradigm. Utilizing such fictitious entities as components to create fictitious CBPs is the root cause of the infamous software crisis.

This problem can be overcome by creating and utilizing virtual entities instead of fictitious entities. However, it is impossible to create a virtual entity for any real entity without a valid scientific understanding and nuanced grasp/insights into the reality of real entity.

Therefore, Pioneer-Soft has developed real hard science 'Componentology', which has accumulated valid scientific knowledge comprising of descriptions, concepts, and explanations to objectively understand the reality of physical entities such as the essential properties, behavior, features, internal mechanisms, and functions of components; anatomy, design, and structure of CBPs; and mechanisms of CBE.

Pioneer-Soft has secured patents for all the necessary inventions to create virtual entities based on a valid scientific understanding of real components and CBPs and to properly utilize virtual entities, which can address the infamous software crisis.

In summary, the software world has been creating and using fictitious entities (based on the existing flawed dominant paradigm), which is the root cause of the existing infamous software crisis. The software crisis cannot be addressed without creating and utilizing virtual entities based on valid scientific knowledge equivalent to Componentology.

Today, software engineering is creating and utilizing fictitious entities based on fiction or junk science, comprising fundamentally flawed descriptions and concepts. This phenomenon is the root cause of the infamous software crisis. We have devised solutions for this crisis by creating and utilizing virtual entities grounded in valid scientific understanding and BoK (Body of Knowledge), encompassing objective descriptions, theories, and concepts accumulated by the real hard science of Componentology.

It is impossible to solve the infamous software crisis without creating and using virtual entities for components that are essential to building each software product as a virtual CBP. It is nearly impossible to create an accurate enough virtual entity for a real entity without an accurate and objective understanding of the real entity. Hence, we have created a new branch of science named Componentology to gain a valid scientific understanding of real entities such as Components and CBPs, to build their virtual counterparts in software.

Pioneer-Soft created all the necessary inventions such as tools, technologies, and mechanisms to create and utilize virtual components to build virtual CBPs. Each virtual CBP is built by plugging in multiple virtual pluggable components as illustrated in FIG-2. Today, it is impossible to create and utilize virtual components and build virtual CBPs without our patented tools, technologies, and mechanisms.

It is impossible to draw a picture of anything (e.g., an animal, flower, fruit, or face of a person of interest), without having any previous knowledge, from insidiously misleading and flawed descriptions. Similarly, it is impossible to create and utilize virtual components by relying on fundamentally flawed and insidiously misleading descriptions and concepts such as those found in Exhibits B1 to B4**.

Componentology, or anything remotely resembling it, is unheard of in the software community today. Without Componentology, inventing virtual components and the necessary mechanisms to build each software product as a CBP, as illustrated in FIG-2, would be impossible.

10. Exhibit A2: Important Facts about Components, CBPs, & CBE

Question-1: What is the striking difference between the specific kind of part that is most certainly a component and all other kinds of parts that are most certainly not components, in the context of engineering disciplines such as mechanical, aerospace, and electronic engineering paradigms which design and build large and complex products (e.g. spaceships, supercomputers, and airplanes)?

Answer-1: Any part can be a component if and only if the part can be "loosely coupled", where the intended meaning of the term "loose coupling" can be defined by using the phrase "loosely coupling a part" which implies (a) "assembling a part" (e.g. a moving part such as an engine, gearbox, wheel), in the context of engineering paradigms for mechanical and aerospace engineering disciplines, and (b) "plugging in a part" (e.g. CPU, DRAM, and hard-drive), in the context of engineering paradigms for electronic and computer engineering disciplines.

Question-2: What is the striking difference between (1) any product that is certainly a CBP (Component-Based Product), which is designed and built by employing CBE (Component-Based Engineering) paradigm, and (2) any product that is certainly not a CBP, which is designed and built by employing non-CBE (Component-Based Engineering) paradigm?

Answer-2: Any product can be a CBP if and only if the product is built by assembling multiple components as in FIG-2. For example, any product (e.g. automobile, airplane, factory machinery, computer) that can be disassembled into components (where all the components can be assembled to rebuild the product) is certainly a CBP. Conversely, any product (e.g. a house, skyscraper, software application, complex pharma products, or bridge) that cannot be disassembled and reassembled is certainly not a CBP. Hence, engineering disciplines such as Mechanical, Electronics, and Aerospace design and build large products as real CBPs as in FIG-2, while engineering disciplines such as Civil, Chemical, Pharma, and Software design and build large products as non-CBPs as in FIG-1.

Question-3: What is the striking difference between an engineering discipline that is certainly employing real CBE (Component-Based Engineering) paradigm to design and build large products and an engineering discipline that is certainly not employing CBE paradigm to design and build large products?

Answer-3: Engineering disciplines such as automobile, mechanical, aerospace, and electronic engineering design and build each large and complex product (e.g., a car, computer, airplane, or machinery) as a CBP, so each of these engineering disciplines is certainly employing real CBE paradigm. Engineering disciplines such as civil, chemical,

** Exhibits B1 to B4 in sections 12 to 15 between pages 22 to 29

and software engineering do not design and build each large and complex product (e.g. a house, skyscraper, software application, bridge) as a CBP, so each of these engineering disciplines is certainly not employing the CBE paradigm.

It is indisputable that great industrial engineering inventions, such as Eli Whitney's interchangeable components, the stationary assembly line patented by Olds, and Ford's moving assembly line, vastly increased productivity and quality by reducing the costs of assembling and replacing each of the components. It is highly desirable to minimize or even eliminate the cost of replacing any of the pluggable components in FIG-2. The goal is to provide outstanding service access to each component. ***See Section 32 on page 55 (Service Access).***

11. Kepler's Laws Analogy for Exhibit A1

Humanity believed 2300 years ago that the Earth was static at the center. Thousands of pages of documents or scriptures were created until the 15th century to describe the planetary motions, orbits, and paths of planets in our solar system. This Body of Knowledge (BoK) created to understand reality is known as the geocentric paradigm or Ptolemaic system, comprising inexplicable epicycles and retrograde motions. The BoK for the Ptolemaic system had been filled with flawed descriptions and concepts, painting an illusory perception, which constitutes junk science.

On the other hand, the three laws of Kepler, formulated in the early 17th century, provided highly accurate descriptions of the planetary motions and paths in our solar system, which were hundreds of times more precise and remarkably close to objective reality. Nearly every description, concept, or explanation within the Ptolemaic system (found in thousands of scriptures and literature of the era for the geocentric paradigm) was false, as none of them aligned with the truth and reality described by Kepler's laws.

Kepler's laws epitomize great discoveries: They accurately describe planetary orbits, in stark contrast to the over 3000 pages of the Ptolemaic system, which were filled with false, inconsistent, and ambiguous descriptions regarding the dominant geocentric paradigm of the time. I highly recommend watching this fascinating short video: http://componentology.org/Vid/1.** It serves as a compelling reminder that the less humanity knows, the more room there is for argument and disagreement***. The disagreements and arguments in Exhibits B1 & B3 illustrate this. Conversely, when truth and reality are known (as in Exhibit A1 on page 8), there's nothing left to dispute or speculate about.

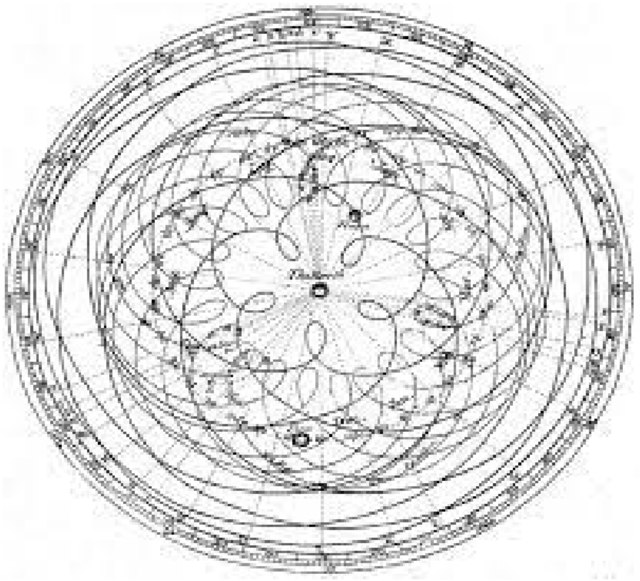


FIG-4: The Ptolemaic system or geocentrism filled with inexplicable unstable motions, was considered the reality until the 16th century

This epitomizes a great discovery, where

** http://Componentology.org/Vid/1

*** http://Componentology.org/Vid/2

a few concise paragraphs illuminate truth and reality far more accurately, especially when contrasted with the thousands of books and research papers authored by esteemed experts of the time that were unsuccessful. Scientific research is a journey aimed at comprehending objective reality, continually striving to get closer and closer to it. Since false assumptions (e.g., Exhibits B1 & B3 in Sections 12 & 14) divert these efforts down the wrong path, it is impossible for research efforts to get closer to objective reality without following the correct path (e.g., Exhibits A1 & A2**).

Analogous to Kepler's laws, one of the goals of Componentology is to establish a few foundational principles that can accurately describe reality in about 180 words. While the four foundational principles in Exhibit A1 may not be flawless, they undeniably represent significant progress in the right direction. Nearly every description, concept, or explanation for CBSE known today (found in existing literature such as thousands of books and research papers, as exemplified by Exhibits B1 to B4) is false, as none of them is consistent or congruent with the above four foundational principles for genuine CBSE outlined in Exhibit A1 (See on page 8).

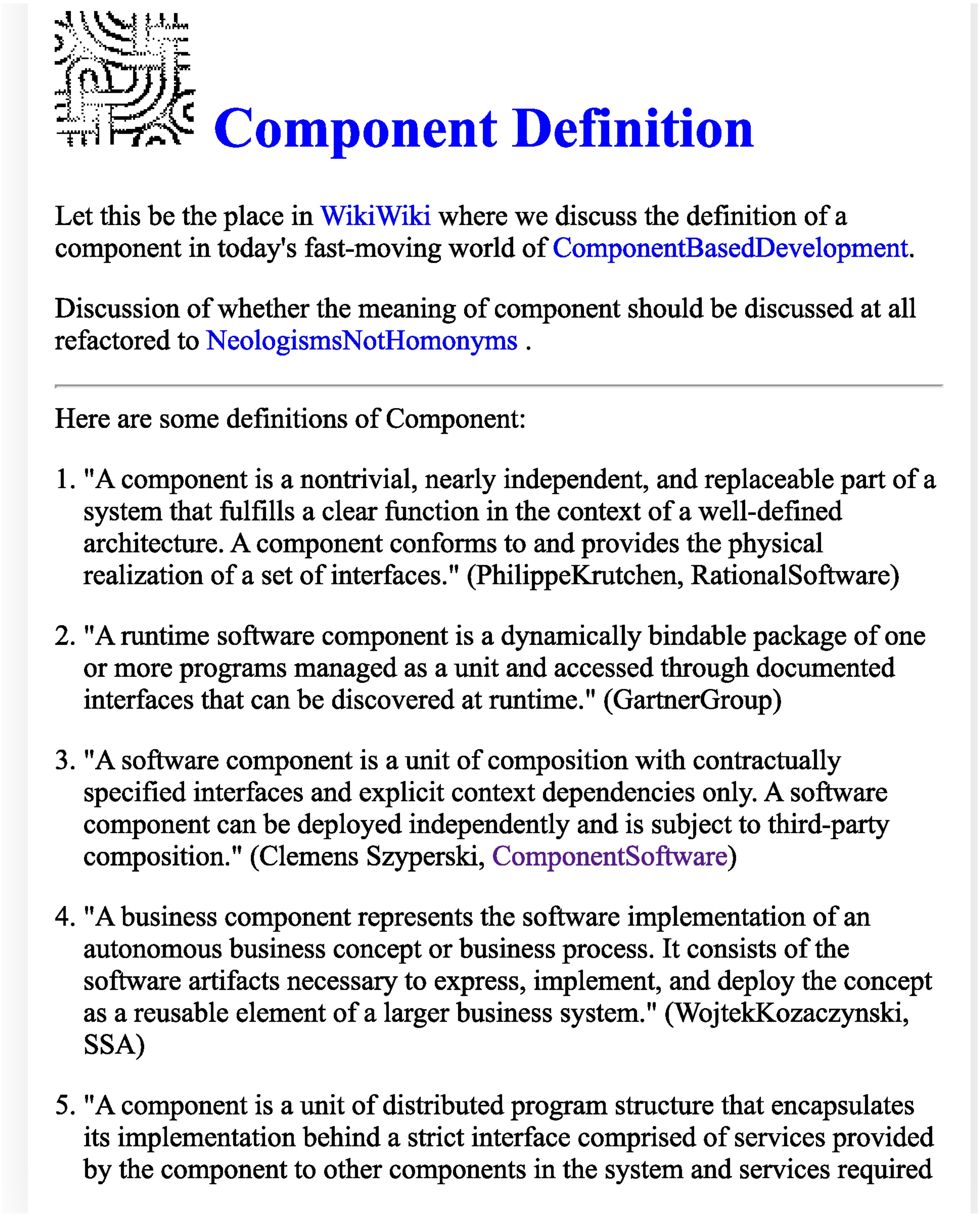
Today, it is impossible for any expert who contributed a description (e.g., Exhibits B1 & B3) to deny the obvious fact: The description or explanation they have provided is subjective and ambiguous, not intended to offer an objective or scientific depiction of real components. Furthermore, every expert acknowledges that the descriptions or explanations of all other renowned experts in the field are not necessarily false, even though each of these descriptions or explanations is inconsistent and substantially disagrees with each other.

It's not logically possible for every expert's description, along with other descriptions that contradict theirs, to be correct simultaneously. Therefore, any description or concept known and utilized today that is substantially inconsistent and contradicts the four foundational statements in Exhibit A1 must be false, just as any description or concept that is substantially inconsistent and contradicts Kepler's laws is false.

Joke: Numerous brilliant software researchers have been working on a top-secret project to create entities such as Components, CBE, CBD, and CBPs in software for decades. These entities are so top-secret that even the researchers do not know about the mysterious entities they are creating. Is it a magic?

** Exhibits A1 & A2 in sections 4 & 10 on pages 8 to 19

12. Exhibit B1: A Sample Set of Component Descriptions



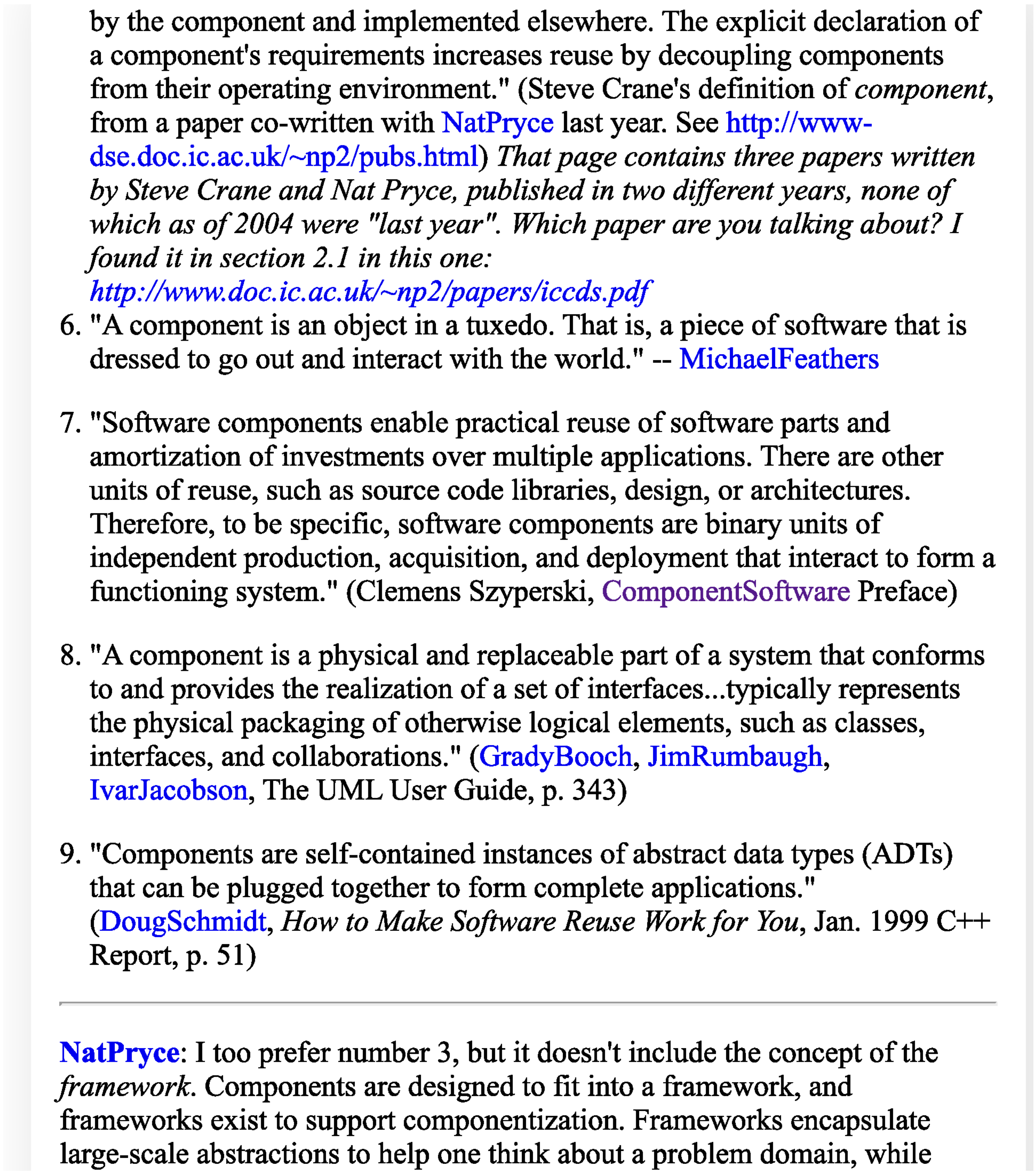
Component Definition

Let this be the place in WikiWiki where we discuss the definition of a component in today's fast-moving world of ComponentBasedDevelopment.

Discussion of whether the meaning of component should be discussed at all refactored to NeologismsNotHomonyms .

Here are some definitions of Component:

1. "A component is a nontrivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interfaces." (PhilippeKrutchen, RationalSoftware)
2. "A runtime software component is a dynamically bindable package of one or more programs managed as a unit and accessed through documented interfaces that can be discovered at runtime." (GartnerGroup)
3. "A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to third-party composition." (Clemens Szyperski, ComponentSoftware)
4. "A business component represents the software implementation of an autonomous business concept or business process. It consists of the software artifacts necessary to express, implement, and deploy the concept as a reusable element of a larger business system." (WojtekKozaczynski, SSA)
5. "A component is a unit of distributed program structure that encapsulates its implementation behind a strict interface comprised of services provided by the component to other components in the system and services required



by the component and implemented elsewhere. The explicit declaration of a component's requirements increases reuse by decoupling components from their operating environment." (Steve Crane's definition of *component*, from a paper co-written with NatPryce last year. See http://www-dse.doc.ic.ac.uk/~np2/pubs.html) *That page contains three papers written by Steve Crane and Nat Pryce, published in two different years, none of which as of 2004 were "last year". Which paper are you talking about? I found it in section 2.1 in this one: http://www.doc.ic.ac.uk/~np2/papers/iccds.pdf*

6. "A component is an object in a tuxedo. That is, a piece of software that is dressed to go out and interact with the world." -- MichaelFeathers

7. "Software components enable practical reuse of software parts and amortization of investments over multiple applications. There are other units of reuse, such as source code libraries, design, or architectures. Therefore, to be specific, software components are binary units of independent production, acquisition, and deployment that interact to form a functioning system." (Clemens Szyperski, ComponentSoftware Preface)

8. "A component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces...typically represents the physical packaging of otherwise logical elements, such as classes, interfaces, and collaborations." (GradyBooch, JimRumbaugh, IvarJacobson, The UML User Guide, p. 343)

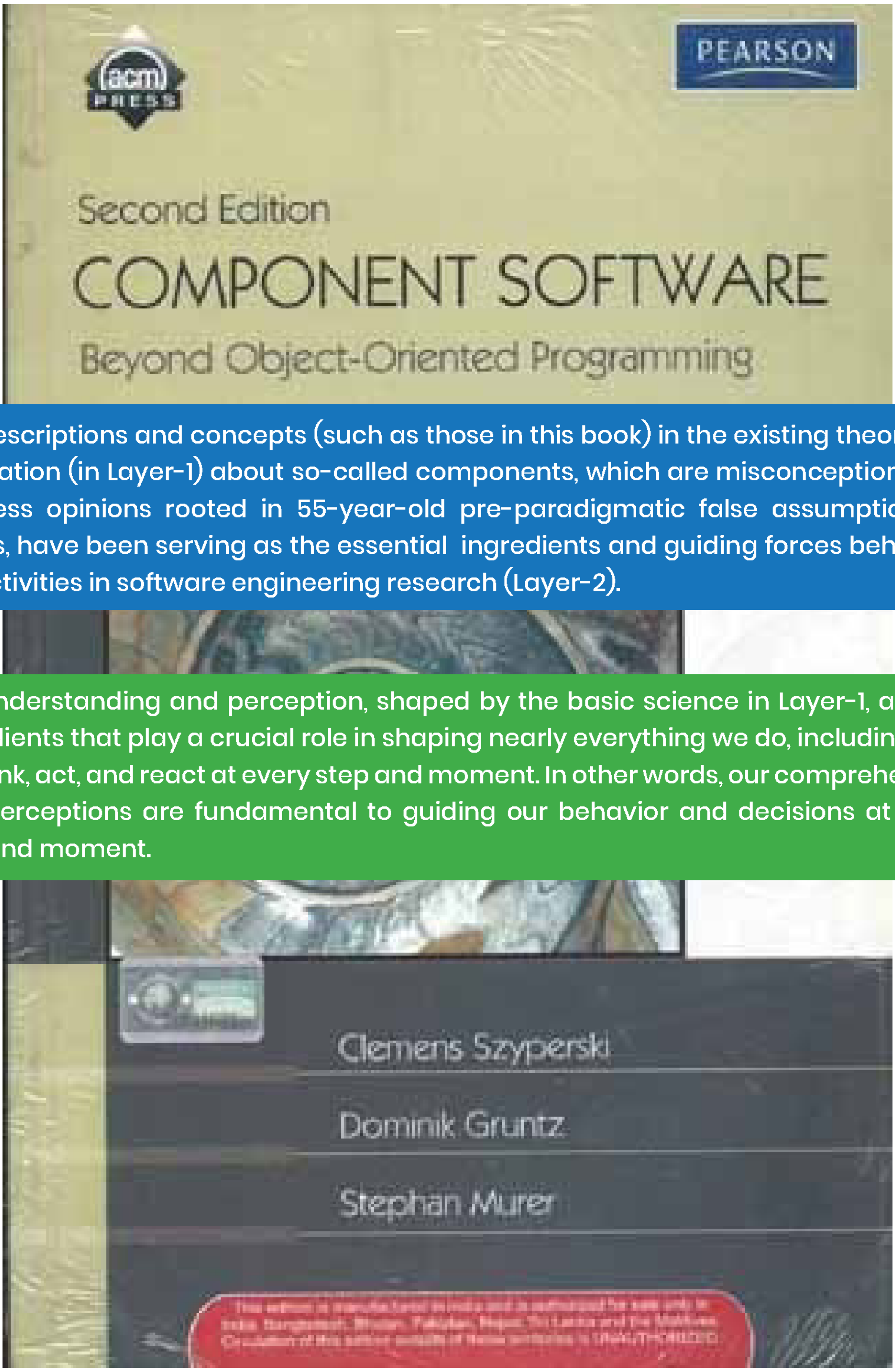
9. "Components are self-contained instances of abstract data types (ADTs) that can be plugged together to form complete applications." (DougSchmidt, *How to Make Software Reuse Work for You*, Jan. 1999 C++ Report, p. 51)

NatPryce: I too prefer number 3, but it doesn't include the concept of the *framework*. Components are designed to fit into a framework, and frameworks exist to support componentization. Frameworks encapsulate large-scale abstractions to help one think about a problem domain, while

The above and other such baseless descriptions, toxic misconceptions, and beliefs in the existing theoretical foundation (Layer-1 of Table-1 on page 6) have served as the essential ingredients and guiding force behind all applied research activities (Layer-2) in software engineering since the 1968 NATO SE conference.

Reference: https://wiki.c2.com/?ComponentDefinition

13. Exhibit B2: A Sample Book on CBSE



The descriptions and concepts (such as those in this book) in the existing theoretical foundation (in Layer-1) about so-called components, which are misconceptions and baseless opinions rooted in 55-year-old pre-paradigmatic false assumptions or beliefs, have been serving as the essential ingredients and guiding forces behind all the activities in software engineering research (Layer-2).

Our understanding and perception, shaped by the basic science in Layer-1, are key ingredients that play a crucial role in shaping nearly everything we do, including how we think, act, and react at every step and moment. In other words, our comprehension and perceptions are fundamental to guiding our behavior and decisions at every step and moment.

Reference: http://Componentology.org/Ref/3

In addition to exerting direct influence, the concepts and prevailing climate of opinions within the existing theoretical foundation subtly influence perception in multiple nuanced ways that are often not immediately apparent. This profoundly altered perception makes it particularly challenging to subvert any deeply entrenched and widely practiced, yet flawed, dominant paradigm.

Introduction

This chapter defines the term software component and summarizes the key arguments in favor of component software. Components are well established in all other engineering disciplines, but, until the 1990s, were unsuccessful in the world of software. The reasons behind this failure can be linked to the particular nature of software. The chapter concludes with a discussion of the nature of software, its consequences for component software, and lessons learned from successful and unsuccessful approaches.

1.1 Components are for composition

One thing can be stated with certainty: components are for composition. *Nomen est omen.* (Literally, "the name is a sign"; usually interpreted as: one's name predicts one's fate.) Composition enables prefabricated "things" to be reused by rearranging them in ever-new composites. Beyond this trivial observation, much is unclear. Are most current software abstractions not designed for composition as well? What about reusable parts of designs or architectures? Is reuse not the driving factor behind most of these compositional abstractions?

Reuse is a very broad term covering the general concept of a reusable asset. Such assets can be arbitrary descriptions capturing the results of a design effort. Descriptions themselves normally depend on other, more detailed and more specialized descriptions. To become a reusable asset, it is not enough to start with a monolithic design of a complete solution and then partition it into fragments. The likely benefits of doing so are minimal. Instead, descriptions have to be carefully generalized to allow for reuse in a sufficient number of different contexts. Overgeneralization has to be avoided to keep the descriptions nimble and lightweight enough for actual reuse to remain practicable. Descriptions in this sense are sometimes called components (Sametinger, 1997).

This book is not about reuse in general, but about the use of software components. To be specific, for the purposes of this book, software components are executable units of independent production, acquisition, and deployment that can be composed into a functioning system. To enable composition, a

14. Exhibit B3: Another Set of Component Descriptions

The following and other such baseless descriptions, toxic misconceptions, and beliefs in the existing theoretical foundation (Layer-1 of Table-1 on page 6) have served as the essential ingredients and guiding force behind all applied research activities (Layer-2) in software engineering for decades.

What characterizes a (software) component?

Manfred Broy [1], Anton Deimel [2], Juergen Henn [3], Kai Koskimies [4], Franti˘ sek Pl´ a˘sil [5], Gustav Pomberger [6], Wolfgang Pree [7], Michael Stal [8], Clemens Szyperski [9]

[1] Technical University Munich, Germany; E-mail: broy@informatik.tu-muenchen.de, http://www.broy.informatik.tu-muenchen.de/ ~broy/
[2] SAP AG, Walldorf, Germany; E-mail: Anton.Deimel@sap-ag.de, http://www.sap-ag.de/
[3] IBM, Boeblingen, Germany; E-mail: jhenn@de.ibm.com
[4] Nokia Research, Helsinki, Finland; E-mail: kai.koskimies@research.nokia.com, http://www.uta.fi/ ~koskimie/
[5] Charles University Prague, Czech Republic; E-mail: plasil@nenya.ms.mff.cuni.cz, http://nenya.ms.mff.cuni.cz/ ~plasil/
[6] University of Linz, Austria; E-mail: pomberger@swe.uni-linz.ac.at, http://www.swe.uni-linz.ac.at/pomberger/
[7] University of Constance, Germany; E-mail: pree@acm.org, http://www.swe.uni-linz.ac.at/wolf/
[8] Siemens AG, Munich, Germany; E-mail: Michael.Stal@mchp.siemens.de
[9] Queensland University of Technology, Brisbane, Australia; E-mail: c.szyperski@qut.edu.au, http://www.fit.qut.edu.au/ ~szypersk/

Abstract. The definitions and discussions below were contributed via e-mail. They are arranged by date. The experts, listed alphabetically above, participated in this virtual round table during the first quarter of 1998.

Key words: Software component – Component versus object/class – Component versus module

---

Anton Deimel

1. A component represents one or more logical or organization-related processes or tasks (for example, purchasing, sales or management of master data).
2. A component is more coarse-grained than single classes; in other words, a component usually consists of several logically coherent classes.
3. A component is unique from other components because a class can be assigned only once to a component.
4. A component may consist of other components.
5. A component uses precisely-defined interfaces to communicate with other components.
6. A component is independent of the release (components can be upgraded and distributed) and can be delivered separately.
7. Frameworks form the underlying technology for components.
8. A component may be a client and server for other components.

Points 1–4 come from an article by Wolfgang Hesse and Friedrich Weltz entitled "Project Management for Evolutionary Software Development".

Wolfgang Pree, Gustav Pomberger

Our definition of components is derived from examining the deficiencies of the object-oriented paradigm:

- Classes / objects implemented in one programming language cannot interoperate with those implemented in other languages.
- Objects are typically composed on the language level. Black-box composition support is missing, that is, visual / interactive tools that allow the plugging together of objects.

Characteristics of components:

- A component is simply a data capsule. Thus information hiding becomes the core construction principle underlying components.
- A component can be implemented in (almost) any language, not only in any module-oriented or object-oriented languages but even in conventional languages.
- As a consequence, standards for describing components have been established.
- The component (module) interface is described either
  - textually by means of an interface description language (IDL) or
  - visually / interactively using appropriate tools.

- Framework architectures form the enabling technology of plug & play software, where most adaptations can be achieved by exchanging components. Visual, interactive composition tools are ideally built on top of domain-specific frameworks.
- Single-component reuse is less attractive. It means that programmers build the overall software system architecture on their own. They have to locate the appropriate components in a Lego building block library and define their interactions.

Kai Koskimies

It is probably very difficult to cover all aspects that are associated with components in different contexts. There are lists of component features in some textbooks trying to do this, but I suspect that they are rather long because the authors themselves are a bit unsure of the essence of a component. And so am I. Anyway, I would suggest something like the following:

> A component is a system-independent binary entity which implements one or more interfaces. An interface is a collection of signatures of services belonging logically together.

A point here is that since a component implements interfaces, it can be plugged in to any context requiring one of those interfaces. System-independence means interoperability with respect to languages, platforms, applications, tools etc. The fact that an interface consists oflogically related services implies that a component has a "meaning", so to say, i.e. an identifiable role in the system it is plugged into. As far as I can see, this definition does not contradict Wolfgang's characterizations. BTW, I noted that Oscar Nierstrasz has used the definition: "static abstraction with plugs". That is perhaps too terse for my taste.

Clemens Szyperski

Here is a compact definition that we developed in the first Workshop on Component-Oriented Programming (WCOP'96) at ECOOP'96 in Linz:

> A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties [WCOP'96 Summary in ECOOP'96 Workshop Reader, dpunkt Verlag, 1997, ISBN 3-920993-67-5].

The entire one-day workshop mostly concentrated on producing this definition, which since then has survived intact.

Michael Stal

To complete the definitions parade I want to just add my own definition:

> A component is a binary unit that exports and imports functionality using a standardized interface mechanism. The underlying component infrastructure supports composition of components by providing mechanisms for introspection, event-handling, persistence, dynamic linking and layout management. In general, application frameworks are required for building components as well as for composing them.

What about this definition that incorporates what componentware technologies such as Opendoc, ActiveX Controls and JavaBeans provide.

Franti˘ sek Pl´ a˘sil

In addition to CORBA and dynamic component updating (mainly in Java), my research interests include Architecture Description Languages (ADLs). Below are my comments on the component concept from this perspective.

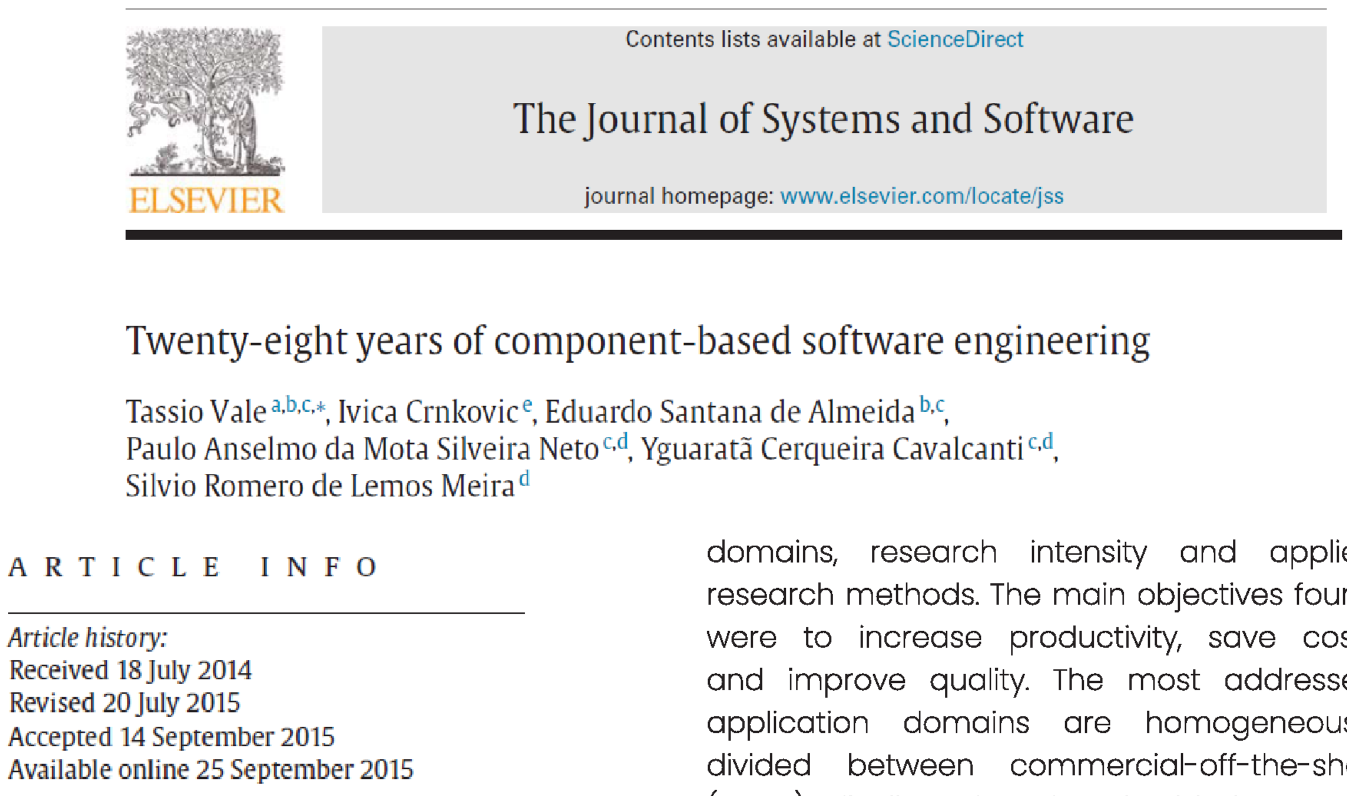
```
> From szypersk@fit.qut.edu.au Sat Jan 31 05:01 MET
> 1998
> Here is a compact definition that we developed in the
> first Workshop on Component-Oriented Programming
> (WCOP'96) at ECOOP'96 in Linz:
>
>   A software component is a unit of composition with
>   contractually specified interfaces and explicit
>   context dependencies only. A software component can
>   be deployed independently and is subject to
>   composition by third parties.
```

A crucial point that was lacking in Kai's two-liner: components that are at all useful cannot be entirely context-independent. However, they have to come with a clear specification of what it is that they depend on: required interfaces, that is, services they need but don't provide themselves.

- Sure, in ADLs a component specification typically includes the "provides" and "requires" clauses (interface(s)).
- Components are interconnected: (provides / requires connections) to cover different paradigms ofinterconnection under one roof; ADLs provide the "connector" abstraction. Different types of connectors may coexist in one architecture (RPC, events, pipe, ORB, ...). Connectors also solve the interface adaptation problems; informally, there are a number of adapters (method granularity) in a connector (interface granularity).
- Components can be nested.

15. Exhibit B4: A Paper on 28 Years of CBSE

Reference: http://Componentology.org/Ref/4 or at https://www.es.mdh.se/pdf_publications/4272.pdf



Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Twenty-eight years of component-based software engineering

Tassio Vale[a,b,c,*], Ivica Crnkovic[e], Eduardo Santana de Almeida[b,c], Paulo Anselmo da Mota Silveira Neto[c,d], Yguaratã Cerqueira Cavalcanti[c,d], Silvio Romero de Lemos Meira[d]

A R T I C L E   I N F O

Article history:
Received 18 July 2014
Revised 20 July 2015
Accepted 14 September 2015
Available online 25 September 2015

Abstract

The idea of developing software components was envisioned more than forty years ago. In the past two decades, Component-Based Software Engineering (CBSE) has emerged as a distinguishable approach in software engineering, and it has attracted the attention of many researchers, which has led to many results being published in the research literature. There is a huge amount of knowledge encapsulated in conferences and journals targeting this area, but a systematic analysis of that knowledge is missing. For this reason, we aim to investigate the state-of-the-art of the CBSE area through a detailed literature review. To do this, 1231 studies dating from 1984 to 2012 were analyzed. Using the available evidence, this paper addresses five dimensions of CBSE: main objectives, research topics, application domains, research intensity and applied research methods. The main objectives found were to increase productivity, save costs and improve quality. The most addressed application domains are homogeneously divided between commercial-off-the-shelf (COTS), distributed and embedded systems. Intensity of research showed a considerable increase in the last fourteen years. In addition to the analysis, this paper also synthesizes the available evidence, identifies open issues and points out areas that call for further research.

Introduction

Component-Based Software Engineering (CBSE) promotes the development of software systems through construction from existing software components, the development of components as reusable entities, and system evolution realization by the customization and replacement of components (Szyperski, 2002). The CBSE idea is not new. It was envisioned more than forty years ago by McIlroy (1968) who provided an idea of commercial component production similar to that found in other engineering fields. However, most of the research work on CBSE has emerged in the last two decades. The motivations behind CBSE have been of a business and technical nature, i.e. increased efficiency and effectiveness, lower costs, and shorter time to-market on one side, and improved quality with regards to fewer errors, improved performance, maintainability, portability, etc. on the other side. However, for many of these claims there is no proper evidence. Furthermore, after many years of development, the question arises whether the research in this field has fulfilled its goals, or if there are still important issues that require further research. Regarding these years of development in the CBSE research field, Schneider and Han (2004) argue that a literature review on CBSE is important in order to investigate the state-of-the-art, pointing out research topics which researchers and practitioners should investigate.

In this context, we present a systematic mapping study (Kitchenham and Charters, 2007; Petersen et al., 2008) performed to map out the CBSE area in the period 1984–2012. The goal is to synthesize evidence of contributions from academic publications. We identify the existing research trends, open issues, and areas for further improvement.

15.1. Observations on the origins & existing flawed dominant paradigm for CBSE

This paper was written in 2014 and revised in 2015, so it is safe to assume that it covered almost all notable research publications and the state of knowledge on CBSE up to 2014. The paper analyzed 1,231 research papers on CBSE published until 2014, and there is no evidence that any of these papers made any attempt to create or use anything comparable to Componentology. Since 2014, no evidence has been found that anyone else in the software world has attempted to create anything similar or comparable to Componentology.

It is impossible to create virtual entities in software (e.g., virtual components and virtual CBPs) without sufficient understanding and knowledge of the nuanced aspects of the physical entities that are the real-world counterparts of these virtual entities.

Creating and utilizing true virtual components in software can substantially increase productivity and quality in designing and building large and complex software products as virtual CBPs, as illustrated in FIG-2, while also significantly reducing maintenance costs. This is comparable to how real components have a proven track record of enhancing productivity, quality, and sustainability (i.e., by significantly reducing maintenance costs) in the design and construction of large and complex physical CBPs.

Heliocentrism is a paradigm-shifting fundamental discovery that has served as the foundation for modern astronomy and physics. Similarly, Componentology is a paradigm-shifting fundamental discovery that provides the essential foundation not only for the virtual-CBE paradigm in software but also for the broader field of software engineering. Heliocentrism subverted the flawed geocentric paradigm, which was rooted in 2300-year-old pre-paradigmatic false assumptions. Similarly, Componentology subverts the existing flawed CBE paradigm, which is based on 55-year-old pre-paradigmatic false assumptions, such as those made during the NATO Software Engineering Conferences in 1968 & 1969.

Creating & Utilizing True Virtual Components to Build Software Products as Virtual CBPs as illustrated in FIG-2 on page 8 -- Based on Componentology, which is a new branch of science created to acquire comprehensive & valid scientific understanding & knowledge about the objective reality of physical components, CBPs & mechanisms of CBE.

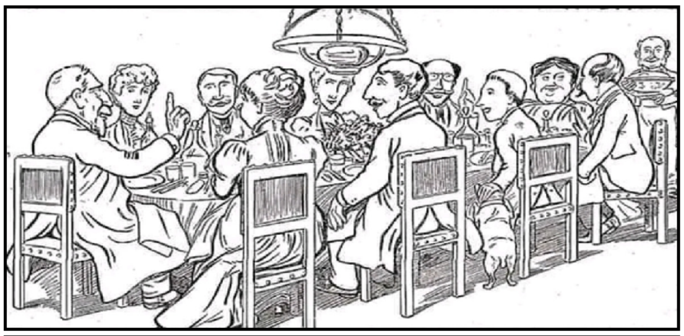
Reference: https://www.es.mdh.se/pdf_publications/4272.pdf

16. Description of Junk Science (or Fiction)

There must be a Body of Knowledge (BoK) associated with every academic discipline (or junk science), where the BoK comprises of multiple pieces, parts, or sections of knowledge such as descriptions, theories, concepts, and explanations. There are many kinds of academic disciplines such as soft sciences, social sciences, axiomatic systems, and formal sciences like mathematics and computer science. The distinction between trustworthy academic disciplines and untrustworthy/unscholarly junk science lies in their respective BoK:

A general description or definition of trustworthy knowledge in any academic discipline is as follows: The BoK of a discipline, comprising of multiple parts or pieces of knowledge such as descriptions, concepts, and theories, must meet certain essential conditions and criteria. These include: (i) ensuring consistency among all parts or pieces of knowledge in the BoK, (ii) ensuring the validity of each part or piece through rigorous testing and falsification; not just before inclusion into the BoK but also whenever doubt arises regarding its validity, and (iii) maintaining a high degree of clarity and precision to minimize ambiguity and subjectivity. Any piece or part of the knowledge can be considered trustworthy if it satisfies or endeavors to meet all three essential conditions.

The BoK of a field is deemed untrustworthy and can be unequivocally classified as junk science if it comprises of multiple parts or pieces of knowledge such as descriptions, concepts, and theories, and if these parts or pieces satisfy at least two conditions or criteria. These include (i) widespread contradictions or inconsistencies among the parts or pieces of knowledge, (ii) the majority of parts or pieces being deemed untrustworthy because they are untestable and unfalsifiable, lacking viable and tailored methods or tools for proper validation through testing and falsification, and (iii) a high degree of ambiguity and subjectivity characterized by a lack of clarity, consensus, and agreement among experts. The current theoretical foundation for conducting research in software engineering meets all three conditions for classification as junk science. For example, there are widespread inconsistencies and disagreements among renowned experts, as evidenced in Exhibits B1 and B3.



Above: Computer scientists unanimously agreeing on the supremacy and vital need for Components and Component-Based Engineering for the past 50 years.

*Below: Computer scientists have been still wrestling to agree on the right descriptions & definitions for Components & Component-Based Engineering.***

Is this not junk science: No one in the software world today knows or agrees with each other about descriptions of CBE, CBPs, and Components for CBE/CBPs.

Only Componentology can help computer scientists seek out, test, and validate to scientifically establish and agree on correct descriptions and theories objectively.

** http://Componentology.org/Vid/2 & http://Componentology.org/Vid/1

P.S: Fake scientists and reviewers who cannot (i) grasp the distinction between the BoK of real science and that of fiction, nor (ii) understand the contrast between the trustworthy BoK of an academic discipline, properly validated through rigorous testing, and the BoK of a junk science riddled with untestable, untested, inconsistent, and untrustworthy pieces of knowledge, are unfit to judge this essay.

Additional valid tests to determine whether the BoK of a discipline qualifies as unequivocal junk science include the following criteria: (i) inconsistencies among the parts or pieces of knowledge in the BoK, (ii) ambiguity, disagreements, or misleading concepts, (iii) inconsistency with readily available data, evidence, or observable aspects of relevant or closest reality (e.g., which it may be trying to mimic), and (iv) assertions or ideas lacking viable methods or tools for validation through proper testing and falsification. The presence of proven or widely accepted methods for acquiring necessary data or evidence, as well as for testing and validation, is crucial for maintaining the quality and integrity of essential knowledge. Without any acceptable or viable testing methods, most pieces of knowledge in the BoK are highly susceptible to compromise by undetectable inaccuracies and unsubstantiated claims.

Junk sciences dismiss the principles of the scientific method and scholarly criteria, neglecting to validate individual pieces of knowledge. Unlike legitimate academic disciplines, they lack practical means to test or falsify most pieces of knowledge. This deficiency becomes apparent when their erroneous concepts, theories, and beliefs are confronted with solid data or evidence, often eliciting defensive reactions from experts in the field. Such defensiveness underscores the absence of acceptable methodologies for scrutinizing and improving knowledge, as well as the utter inability to detect and eliminate false concepts and theories, serving as clear evidence of junk science. The BoK represented in Exhibits B1 to B4[1] fits this description.

17. Illustration of Junk Science Using an Example

Illustration of unscientific "junk science" using an example: Imagine you want to learn about and understand something to solve a complex problem in a discipline. You begin searching, gathering, and systematically studying literature (e.g., highly reputable books, research papers, and relevant websites) about the subject, such as descriptions, concepts, and theories. Let us say you encounter multiple inconsistent and ambiguous descriptions, concepts, and explanations for it, all of which are actively used in Layer-1. You then rely on these descriptions and concepts to conduct research in Layer-2.

Imagine that these multiple descriptions are not only inconsistent with each other but also highly subjective and ambiguous. Similarly, the multiple concepts and explanations are not only subjective and ambiguous but also substantially inconsistent and contradict each other. Is it possible to solve the problem by making sense of and using such inconsistent, ambiguous, and misleading information?

An example of junk science is evident in Exhibits B1 to B4**, which demonstrate that the existing descriptions, concepts, and explanations for software components and CBE for software epitomize this phenomenon of junk science. It is not difficult to conclusively demonstrate that most parts or pieces of knowledge (e.g., descriptions, concepts, explanations) in the current theoretical foundation regarding software components and CBE for software align with the

** Exhibits B1 to B4 in sections 12 to 15 between pages 22 to 29

characteristics of unscientific junk science

Let me illustrate junk science using an example: It constitutes junk science when certain species of animals have multiple inconsistent descriptions. For instance, one description provided by a renowned expert asserts that the species of animal has four legs, while another description from a different renowned organization asserts that the species has two legs. Yet another description from yet another renowned conference suggests that the species has no legs and instead swims like a fish, crawls like a snake, or does not move at all. In such a scenario, which description can be trusted and relied upon to, for instance, draw a picture of the animal?

Furthermore, one description attributed to a renowned expert assert that the species has two wings like birds, while another authority asserts that it possesses two horns akin to cows. Yet another source suggests that the species bears horns on its nose, like a rhinoceros, while another description indicates that it has two tusks resembling those of elephants. Are you confused yet? If even renowned experts who provided these descriptions are unsure, is it possible to make sense and rely on such inconsistent descriptions? This analogy of inconsistent descriptions of a mythical animal illustrates the existing inconsistent descriptions, concepts, and explanations (e.g., in Exhibits B1 to B4) we have for mythical software components and CBE for software.

What kind of animal is it? Can even the most skilled painters create accurate depictions of it? If even brilliant experts in the field cannot make sense of it, then it must be junk science. ***Please refer to cartoon 2 on page-30,*** which portrays persistent disagreement and arguments in attempting to understand the mythical entities misleadingly referred to as components. Ignorance fills volumes. For further details, visit ***http://componentology.org/Vid/1 & see Section 11 on page 20.*****

The most vital rule and criterion for the BoK (comprising of description, theories, and concepts) of any academic discipline (including axiomatic systems and formal sciences) is that every part or piece in the BoK must be consistent with other parts in the BoK. Therefore, if most descriptions, theories, and concepts in the BoK of a discipline are substantially inconsistent with each other, the discipline is toxic junk science. The secondary rules and conditions are that each part of the BoK (e.g., a description, theory, or concept) must strive to be highly objective, unambiguous, testable, and falsifiable.

How is it possible for anyone to build each software product as a virtual CBP if they haven't heard of and are unaware of the reality, anatomy, and structure of real CBPs? Today, no one in the software world is aware of the reality such as the internal mechanisms, anatomy, and structure of CBPs.

Consider a scenario where you're tasked with drawing a picture of the animal described above. In this situation, imagine encountering numerous contradictory and ambiguous descriptions that are untestable and unfalsifiable, like those found for mythical components in sources such as Exhibits B1 and B3. These descriptions in Exhibits B1 and B3 are baseless, inconsistent, and highly ambiguous.

How do you determine which description is accurate? Suppose you place trust in a certain description provided by a researcher or authority whom you consider to be the most trustworthy. How do you double-check its accuracy? Do you possess any reliable corroborating data or evidence to support your belief? It's crucial to note that blind trust in authority is not acceptable in science, especially when the authorities aren't certain of their own descriptions/opinion due to the lack of reliable corroborating data or evidence.

Is it possible to solve the infamous software crisis by relying on such junk

** http://Componentology.org/Vid/1

science (i.e., mythology about mythical entities disguised as components and CBE) as its theoretical foundation in Layer-1 and conducting engineering research in Layer-2, considering the most likely possibility that the junk science is responsible for creating the infamous software crisis and its dreadful effects, such as the notorious spaghetti code?

Hence, for nearly two decades, I have been advocating for conducting applied research in software engineering in Layer-2 by creating and utilizing hard science as the theoretical foundation in Layer-1. Unfortunately, I have faced fierce and hostile resistance from the existing deeply entrenched anti-science cult who consider it arrogant and disrespectful for a person of color to challenge the 'junk science' created by many famous white men, almost as if it is unscientific and unscholarly for a person of color to do so.

When drawing pictures of elephants, it's incorrect if they don't aim to resemble real elephants. It would be surprising if they looked like rodents, or lizards instead. Virtual representations of real things should strive to closely resemble them.

18. Unscholarly Breach of Academic Integrity

Even a graduate in any academic discipline having basic common sense can understand the following three simple statements about basic scholarly standards & criteria for academic knowledge:

> Knowledge within each hard scientific discipline must adhere to strict rules, conditions, standards, and criteria to be deemed trustworthy, scientific, and useful. The scholarly standards and criteria include consistency, empiricism, objectivity, testability, and falsifiability.
>
> Similarly, within academic disciplines (such as soft sciences, formal sciences, and axiomatic systems), knowledge must strive to meet specific scholarly standards and criteria such as consistency, unambiguity and clarity, as well as data and evidence-based testability and falsifiability to be trustworthy and credible.
>
> However, the existing knowledge** in the theoretical foundation for software engineering blatantly disregards and violates even the most basic scholarly criteria, standards, and requirements, thus rendering it toxic junk science.

Most pieces or parts in the existing BoK such as descriptions, theories, and concepts for CBSE and its components are junk science, which describes fictitious entities. For example, even a cursory reading of these discussions, opinions, and descriptions in Exhibits B1 & B3 makes it apparent that the descriptions and concepts for components and CBE are baseless opinions (rooted in the prevailing prejudices and biases of 1968) of influential authorities and thought leaders.

Is it not unscholarly and a breach of academic as well as scientific integrity to conduct applied research by relying on baseless opinions/fiction? Graduates with a good scholarly foundation agree that it is unscholarly and a breach of academic integrity and scientific rigor to conduct engineering research by relying on junk science, filled with inconsistent and unjustifiable opinions/beliefs about such mythical entities which can only be classified as mythology, fiction, or junk science.

Understanding the difference between virtual entities and fictitious entities confirms that the existing entities within software misleadingly referred to as components,

** Exhibits B1 & B3 in Sections 11 & 13 on pages 22 & 26

are not virtual representations of real components but rather fictitious entities grounded in fiction or junk science.

18.1. Minimum Scholarly Standards for Academic Knowledge

The academic BoK in the theoretical foundation of any academic discipline (including soft sciences such as social sciences) must adhere to specific minimum scholarly standards and fulfill essential academic requirements, and rules without exception. Any knowledge that fails to meet these minimum standards and essential criteria or rules should be regarded as unreliable and baseless opinions. These standards encompass a few vital academic criteria:

Internal Consistency: The various parts or pieces of the BoK must demonstrate a high degree of coherence or consistency with other parts of the BoK,

Clarity and Unambiguity: Each piece or part of knowledge should possess a high degree of clarity and a minimal degree of ambiguity, and

Methodological Rigor: There must be viable, appropriate, and validated tools and methods for acquiring and utilizing relevant, valid data and evidence to enable the validation through testing and falsification of each piece of knowledge.

Even after striving to adhere to such academic and scholarly standards, the BoK of many academic disciplines is far from perfect. Hence, the "Existing BoK for CBSE" (comprising of numerous descriptions and concepts**) must be considered junk or trash, since no attempt has been made to adhere to even minimum scholarly standards.

Adhering to these standards is an essential requirement for every academic discipline to ensure the overall quality and integrity of the BoK. Any BoK that fails to satisfy two or more of these essential requirements cannot be considered academic knowledge and should not be taught to unsuspecting and impressionable students under the guise that it is a trustworthy BoK of an academic discipline. Unfortunately, many parts of the knowledge within the BoK in the current theoretical foundation for software engineering fall far short of meeting all three rules and requirements, rendering it highly unreliable, which makes it toxic junk science (e.g., akin to mythology about mythical entities).

19. Relevant/Applicable Categories of Academic Disciplines

In the context of the "BoK for CBSE," the relevant academic disciplines can be broadly classified into the following three basic categories:

Hard Sciences: The BoK of each discipline comprises of numerous pieces/parts of knowledge, such as descriptions, theories, and explanations, which must pertain to and portray physical/real entities that cannot defy the laws of physics/nature. The prescribed tools, rules, steps, and principles for acquiring and validating (through testing and falsification) each piece of knowledge are provided by the scientific method.

Axiomatic Systems or Formal Sciences: Various pieces and parts of the knowledge within the BoK must be not only consistent with each other but also precise and unambiguous. Drawing or reaching valid or trustworthy logical conclusions is impossible when relying on imprecise and ambiguous axioms (or first

** Exhibits B1 to B4 in sections 12 to 15 between pages 22 to 29

principles) that are inconsistent with each other. The various parts or pieces of knowledge for formal or axiomatic systems must be represented by a set of consistent and precise axioms or constructs such as equations, theorems, and formulas, and new knowledge and insights are acquired by systematically applying logic (e.g., carrying out a sequence of mathematical steps).

As chaos theory reveals, when a sequence of logical steps is carried out based on initial foundational axioms, even minor errors such as inconsistencies/ambiguities in the initial axioms can lead to unpredictable outcomes. This is especially notable in the context of foundational axioms or first principles of deterministic systems, such as the theoretical underpinnings of any engineering discipline**.

Soft Sciences or Academic Discipline: Within the BoK of any discipline, every piece of knowledge, such as a theory or explanation, must not only be supported by valid, relevant, and appropriate data and evidence but also be testable and falsifiable using valid, relevant, and appropriate data and evidence. Having viable, appropriate, and well-established tools, principles, rules, and methods to acquire and utilize data and evidence is crucial, as they play the most vital role in maintaining the quality, trustworthiness, and integrity of the BoK.

The intention and purpose of every academic discipline (including soft sciences), such as economics, sociology, linguistics, anthropology, psychology, and political science, is to accumulate the necessary BoK to gain an understanding of its subject matter. There can be multiple subject matters (e.g., subfields) associated with each academic discipline. Each discipline needs to have well-established, viable, appropriate, and tailored rules, principles, methods, and tools for accumulating and utilizing the necessary corpus of data and evidence to support the understanding based on various pieces or parts of knowledge in its BoK.

Methods for acquiring each piece of knowledge in the BoK include systematically analyzing the relevant corpus of data and evidence to draw conclusions, propose hypotheses, and rigorously validate them through testing and validation using the relevant corpus of data and evidence. In other words, accumulating and utilizing a valid and verifiable corpus of data and evidence plays a central and pivotal role in acquiring each piece of knowledge. Each piece of knowledge, such as a valid explanation or theory proven through rigorous validation (through testing and falsification using data and evidence), is valuable for making predictions, providing explanations, and gaining deeper insights and wisdom vital to address complex problems.

Each piece of knowledge presented or proposed without providing the relevant verifiable data and evidence necessary to ascertain its validation (through testing and falsification, independently by qualified experts) lacks credibility and is unreliable trash.

The existing descriptions and concepts, such as those in Exhibits B1 to B4, are not supported by any data or evidence. Proposing any such description or concept without providing any supporting data or evidence makes them pure speculations or delusions. It is not possible to collect the necessary verifiable data and evidence to support descriptions and concepts about mythical entities such as angels, demons, and winged unicorns, as well as fictitious entities that are deceptively disguised as components and CBPs.

Therefore, the pieces or parts of knowledge in the existing "BoK for CBSE" must be considered junk science as the "BoK for CBSE" does not belong to any academic category: *http://componentology.org/Ref/16*

The descriptions and concepts in

** http://Componentology.org/Ref/9

Exhibits B1 to B3 (in Section 12 to 15) , which are used in the existing theoretical foundation at Layer-1, do not fit into any of the above three categories of academic disciplines. Therefore, they must be classified as junk science.

The existing descriptions and concepts do not describe tangible entities; instead, they describe fictitious entities, so they cannot be classified as hard science.

The existing pieces of knowledge (such as descriptions) are inconsistent and imprecise axiomatic received beliefs. Hence, the discipline whose BoK has been accumulated by relying on such received beliefs cannot be classified as an axiomatic system or formal science. It is impossible to apply logic to draw useful or trustworthy conclusions by relying on such inconsistent, imprecise, and flawed received beliefs.

The existing descriptions and concepts are not soft science because it is impossible to acquire necessary valid data and evidence to support or validate (through testing and falsification) the various pieces of knowledge such as descriptions and theories like those in Exhibits B1 & B3 for such fictitious or mythical entities, which can only exist in the fanciful imaginations of the experts who have proposed them.

The existing theoretical foundation, referred to as the "BoK for CBSE", is riddled with trash such as baseless, inconsistent, and ambiguous descriptions, concepts, and explanations about mythical entities deceptively disguised as components. This mythological trash has been staring every software expert and researcher in the face for decades. It is disgraceful that no one has bothered to notice this junk science, even after being informed about it in no uncertain terms.

If anyone wants proof, they can see it for themselves by simply finding and examining the descriptions, concepts, and explanations known today for so-called components and mechanisms for CBE in the existing literature on software engineering and CBSE. The distinction between hard science and fiction lies in the subject matter being studied to understand or portray within BoK or literature for the discipline (or story book of fiction).

Hard sciences strive to objectively study and understand physical entities such as plants, animals, chemicals, and components, and soft sciences strive to study and understand subject matters such as economics, sociology, anthropology, and psychology. However, fiction does not strive to study anything but portrays fictitious entities such as angels, demons, winged unicorns, and fictitious entities disguised as components.

The goal of Componentology is to revolutionize software engineering research (in Layer-2) by replacing the existing deeply entrenched junk science (i.e., trash in Layer-1**) with a novel real science, which is not only unknown today but also profoundly contradicts the existing deeply entrenched orthodoxies (i.e., belief system).

Achieving this goal necessitates exposing fake scientists and reviewers, even by filing a lawsuit if necessary, and confronting those who vehemently defend the existing junk science and orthodoxies, having already been indoctrinated into it or having contributed to it, and are hostile to real science that challenges the false received beliefs in the existing deeply entrenched junk science.

This content and information can be considered to have been provided in good faith and under oath, and I believe them to be true and accurate. I am willing to face prosecution if any of the core/main contentions are found to be inaccurate or dishonest.

Honest scientists, who possess integrity and moral values, adhere to truths and facts

** Refer to Layer-1 & Layer-2 in Table-1 on page 6

regardless of whether they are under oath in courtrooms or not. Conversely, only dishonest scientists/reviewers engage in nefarious activities if they believe they cannot be punished for their actions, such as resorting to evasive tactics and deception. Such dishonest scientists are the scum of the scientific world.

Is it possible to find this kind of honest and ethical scientists/reviewers: *http://componentology.org/Vid/2* & *http://componentology.org/Vid/3.*

PS: It is impossible to demonstrate that a certain brand of soft drinks contains harmful levels of carcinogenic substances without examining the contents of a few representative samples of the soft drink bottles. However, it is neither necessary nor practical to test every single soft drink bottle. Similarly, it is impossible to demonstrate that paint from a manufacturer contains harmful levels of lead/mercury without examining the contents of a few representative sample cans of paint and/or painted walls. It is not necessary to test every painted wall.

Likewise, it is impossible to demonstrate that the existing "BoK for CBSE" in Layer-1 (comprising of numerous pieces and parts of knowledge such as descriptions, theories, and concepts) is junk science without identifying and using the best possible representative samples, such as those found in Exhibits B1 to B4 (in Section 12 to 15 from page 22). However, it is neither necessary nor practical to test every known description and concept.

20. Distinguishing Between Mythological & Enigmatic Realities

A BoK, which encompasses descriptions, theories, and concepts, for any subject, can become enigmatic if it's built upon only a fraction of reliable data and observable aspects of its reality. Examples of such enigmatic domains include 15th-century astronomy (i.e., Geocentrism) and paleontology (such as the diverse species of dinosaurs). Developing consistent and unambiguous descriptions and concepts becomes impractical when substantial gaps in data/evidence exist, and only a small portion of useful data or evidence is available to rationally test or falsify descriptions, theories, and concepts within the BoK (Body of Knowledge).

A BoK, encompassing descriptions, theories, and concepts about entities, may be termed as mythology if it's founded upon baseless beliefs, myths, or misconceptions. Examples of such mythical realms include books on legendary creatures like ghosts, angels, and demons. It becomes impossible (i) to formulate consistent and unambiguous descriptions and concepts when the body of evidence or data used to construct the BoK involves untestable and unsubstantiated beliefs, myths, and misconceptions, and (ii) to test or falsify the descriptions, theories, and concepts within the BoK**.

The existing descriptions, concepts, and theories about the so-called components and CBE/CBD for software, within the BoK of the existing theoretical foundation in Layer-1 for conducting engineering research in Layer-2, hover between enigmatic and mythical (perhaps closer to mythology). This is because the BoK is rooted in fundamentally false foundational beliefs, assumptions, and misconceptions. This mythological aspect would become especially apparent in light of the BoK in the hard science of Componentology.

** http://Componentology.org/Vid/7

21. Unscholarly Pursuit of Mythology

It's deemed unscholarly and unacademic if the BoK within any academic discipline, covering descriptions, theories, and concepts, lacks widely accepted and utilized formal tools, methods, standards, or processes specifically tailored to the discipline's knowledge domain. These tools are essential for (i) properly collecting and utilizing valid data and evidence, and (ii) facilitating the testing and falsification of descriptions or concepts using such valid data and evidence.

Unfortunately, the BoK concerning components and CBE for software, which includes descriptions and concepts, lacks any tailored tools, methods, standards, or processes for collecting and utilizing necessary valid data and evidence, as well as for testing and falsifying the descriptions and concepts within the BoK.

It's essential to employ appropriate methods for testing and falsification to identify and eliminate false beliefs and to address any imperfections or shortcomings, in order to enhance the overall quality and integrity of the BoK. Data or evidence-based validation through testing and falsification is vital for improving the overall quality and integrity of the BoK. Enhancing the quality and integrity of the BoK is an iterative and continuous process. Validation through testing and falsification is also an invaluable tool for enhancing our understanding and gaining deeper insights for the scholarly pursuit of true wisdom.

It's indeed disheartening to realize that researchers with doctorates are unaware of fundamental principles and concepts, and I am unduly burdened with teaching researchers who are highly reluctant to learn. Shockingly, many of them maintain irrational skepticism towards these elementary cardinal rules, concepts, and principles, such as acquiring, validating, and utilizing a high-quality, trustworthy BoK, which are essential aspects of scholarly pursuit and gaining deeper insights to nurture true wisdom.

Any BoK that violates all or most of the cardinal rules must be treated as untrustworthy and unacademic junk. It is impossible to identify and expose false beliefs, theories, or concepts if no viable method is available to validate each piece of knowledge through testing and falsification. In the context of any academic discipline, each piece of knowledge, whether it's a description or concept, that is not testable and falsifiable must be regarded as untrustworthy junk science.

Most of the existing descriptions, concepts, and theories about so-called components, CBE, and CBD for software in the existing theoretical foundation in Layer-1 (to be relied upon to conduct engineering research in Layer-2**) violate almost every cardinal rule and hence must be treated as untrustworthy junk (e.g., false beliefs or myths). Each piece of academic knowledge must be regarded as untrustworthy junk if it is not testable and falsifiable, or if there are no viable proven methods and tools to test and falsify it.

Except for the theoretical foundation for software engineering, every contemporary academic discipline today benefits from access to viable and proven tools and methods developed and continually refined to validate their knowledge. The modern scientific method provides proven tools, processes, and principles to validate scientific knowledge for all hard scientific disciplines. Created about 400 years ago, it has been continuously perfected ever since by numerous great philosophers of science and scientists.

** Refer to Layer-1 & Layer-2 in Table-1 on page 6

22. Fundamental & Essential Nature of Components

Birds are highly specialized creatures distinguished by their wings and ability to fly. Similarly, components represent a distinct category of entities tailored for specific functions within systems/products. Electronic components, for instance, are designed to be pluggable, facilitating collaboration and communication through data or signal exchange. In contrast, mechanical components are crafted for assembly, enabling movement in various mechanisms. Software components also collaborate and communicate through the exchange of data or signals; thus, they can and must be designed to be pluggable like electronic components (see FIG-2).**

Just as not every animal possesses the attributes of a bird, not every part possesses the characteristics essential to be deemed a component. It is the utmost folly to label any other type of animal (e.g., a pig, cow, or goat) as a kind of bird. Similarly, it is the utmost folly to designate any other type of part as a kind of component. Such errors are due to the ignorance of the unique features, functionality, and built-in mechanisms that differentiate these entities, both in the natural world and within the realm of engineering. Researchers in software committed this kind of utmost folly by labeling many other kinds of parts as a type of component, despite no known type of so-called component possessing the essential inbuilt mechanisms to be deemed as such.

It is the utmost folly to claim expertise on birds without understanding the essential distinguishing characteristics and key differences between animals that are certainly birds and other kinds of animals that are certainly not birds. Similarly, it is the utmost folly to claim expertise on components for CBE without understanding the essential distinguishing characteristics and key differences between parts that are most certainly components and other kinds of parts that are most certainly not components. Hoping to build each software product as a real CBP (as in FIG-2) using the fictitious components (in Exhibits B1 & B3) known today is akin to hoping that pigs or cows can fly.

If someone went to a farm and the farmer showed them a pig or goat, arguing that it was a bird, they would surely consider the farmer insane. I feel the same way about ignorant software researchers who label reusable parts (comparable to paint and cement) as components and define that CBE is using such fake components. I am confident that anyone who learns reality and discovers the truth about components and CBE by mastering the true reality of Componentology would agree with me.

Likewise, parts that are not self-contained and have not implemented the necessary internal mechanisms to plug in are not real software components. The essential characteristics of real software components include: (i) must be implemented to be assembled by plugging in as in FIG-2, and (ii) being self-contained; a somewhat complex concept to explain. Over a hundred pages are dedicated in each of my patents to explain and illustrate the intended meaning of self-contained using many examples. Software components are equivalent to electronic components, which collaborate by exchanging data or signals/events. Such components can be designed to be pluggable.

In the context of parts to build products, reusing a part is a choice and it does not need any inherent nature or essential properties.

** Please refer to FIG-1 & FIG-2 on page 8

However, the inbuilt features and nature of components include being self-contained and having necessary mechanisms so that they can be assembled or plugged in. No one in the software world is aware of these two most fundamental and essential nature/aspects of components, without which no part can be a component.

COTS (Commercial-Off-The-Shelf) or Reusable ingredient parts akin to paint, cement, metals, & alloys are falsely considered components in software engineering, even though ingredient parts (e.g., paint) are COTS but are not assembled or plugged in. It is impossible to create real software components (needed to build real CBPs) without comprehending the essential nature & inherent properties of components. We have secured multiple patents to create and use real software components, where software components are self-contained and have the necessary mechanisms to be pluggable.

Software researchers are deceiving themselves and the rest of the world by falsely labeling certain parts as components, driven by their intense desire and myopic obsession to increase the reuse of already available COTS parts (i.e., so-called components). The choice of reusing those parts is optional. If others choose to use an available part, it is a reusable part, and if no one chooses to use an available part, it is not a reusable part. However, for a part to be utilized as a component as in FIG-2, it must possess certain essential properties, rather than simply being a matter of choice.

In an automobile, the crucial components include the engine for generating locomotion and the wheels for enabling movement through rolling. It is impossible to build automobiles without these crucial components. Similarly, when it comes to physical components, there are two crucial aspects without which physical components cannot exist. Astonishingly, even among those deemed renowned experts or researchers in the software world, particularly in the domain of software components and CBSE, these two crucial facets and the essential nature of components remain unknown.

How can someone be considered a renowned expert and automobile researcher if they are clueless about crucial components of automobiles, such as engines and wheels? Similarly, in today's software industry, no one has any clue about the essential aspects of components, such as being self-contained and having mechanisms for plugging in. The essential aspect of "self-containment" is the most crucial and intricate element, requiring many days or even weeks of hands-on training to master. My patent applications provide comprehensive information and many examples to illustrate 'self-contained' components.

23. Cargo Cult Science: Utilizing Fiction as Theoretical Foundation

The current theoretical foundation, which includes numerous descriptions, concepts, and explanations about so-called components and CBE, is mere fiction or junk science. This fact should be apparent to any scientist familiar with the existing descriptions, concepts, and explanations, such as those found in Exhibits B1 to B4.** The entities known today in software as components, CBPs, and CBE are fictitious or mythical entities that are being deceptively masqueraded and misleadingly disguised as virtual imitations or counterparts of real-world entities.

However, despite not being intended as

** Please refer to sections 12 to 15 on pages 22 to 29 virtual equivalents of real-world counterparts, they are deceptively and fraudulently presented to impressionable software students and new practitioners of CBSE as such. This perpetuates the creation of a false impression and a widespread belief in their virtual equivalence to real-world entities, fostering the misconception that they are indeed virtual counterparts of real-world entities.

Drawing from the accumulated knowledge and understanding of Componentology, it becomes apparent that these virtual entities bear very little relationship and minimal resemblance, similarities, or equivalence to their real-world counterparts, much like the disparity between pigs and war elephants (e.g., to use pigs as war elephants).

Anyone acquainted with the current descriptions and concepts of so-called components in software would undoubtedly recognize that these descriptions bear little resemblance to their real-world counterparts. Instead, they often depict abstract and fictitious entities erroneously labeled as components and have been foolishly attempting to be utilized as such. These activities are comparable to labeling pigs as war elephants and attempting to utilize them as the equivalents of war elephants in battles.

The descriptions or concepts of so-called "components" and "CBPs" in software today are not intended to create virtual replicas or equivalents of their real-world counterparts. Instead, they merely depict fictitious entities, misleadingly disguised and insidiously masqueraded as such. The BoK (encompassing descriptions, concepts, and explanations) within the current theoretical foundation for software engineering neither describes nor pertains to real components, CBPs, CBD, and CBE. Rather, the BoK portrays fictitious entities, which are being deceptively labeled as components, CBPs, CBD, and CBE.

Labeling pigs as roses won't make them smell any better. Similarly, labeling fictitious entities as components won't make them function like genuine components. Understanding the nature, essential properties, and inherent mechanisms of real components is essential to creating equivalent software components. It is akin to cargo cult science to expect fictitious entities to function as components without comprehending the underlying principles and mechanisms. ***Please see Section 23 & 30 on page 40 & 52 respectively.***

Utilizing fiction as the theoretical foundation is more foolish than using electrolytes instead of water to grow crops: https://www.youtube.com/watch?v=ZMHfBobgLSI.** Today, software researchers and experts have been using foolish excuses, much like the excuse that plants crave electrolytes.

Using electrolytes to grow crops does not work; using water does. No evidence in the history of engineering can indicate that utilizing fiction as the theoretical foundation has ever worked. There is no other workable way except using science as the theoretical foundation. The infamous software crisis and notorious spaghetti code are the problems created by utilizing fiction and they cannot be solved without utilizing Componentology.

Summary: The current descriptions and concepts known today in software regarding so-called "components" were developed without considering the essential nature, properties, functions, and attributes of their real-world counterparts. Instead, these descriptions and concepts portray fictitious entities that are deceptively labeled as components, falsely suggesting that these "so-called" components are virtual equivalents, imitations, and/or analogous to their real-world counterparts in software. Therefore, the existing theoretical foundation that is riddled with such fictitious descriptions and concepts regarding mythical entities is essentially fiction, rendering it junk

** http://Componentology.org/Vid/16

science.

In the realm of software, each of the current descriptions for "components" describes certain fictitious elements without considering the essential nature, properties, and functionality of real or physical components.

These descriptions and concepts portray imaginary entities misleadingly labeled as "components," creating the impression that they are analogous or designed to imitate their real-world counterparts. However, this portrayal is misleading because software components are not imitative, similar, or comparable replicas to physical components. Consequently, the current theoretical foundation built upon these flawed descriptions and concepts is essentially fiction, akin to junk science.

24. Sacred Principles of the Scientific Method

The sacred principles of the scientific method encompass respect for valid and verifiable evidence, objectivity, empiricism, validation through testing, and falsification. All these principles are essential for scientific rigor and safeguarding the quality and integrity of knowledge. The lawsuit against NSF.gov is for violating every one of these most basic scientific principles and is based on the three simple indisputable facts listed below:

No qualified scientist in the scientific world can disprove this easily verifiable fact: The existing theoretical foundation for software engineering is a mythology about mythical entities deceptively disguised or masqueraded as Components, CBE (Component-Based Engineering), and CBD (Component-Based Design).

No qualified scientist in the scientific world can disprove this self-evident fact: Componentology is a hard science since it scientifically studies and accumulates valid scientific knowledge about the objective reality of physical entities.

It is impossible to address any unsolved or complex scientific or technological problem, such as the infamous software crisis, by using junk science such as the existing mythology. So, one must rely on valid science such as Componentology.

"By denying scientific principles, one may maintain any paradox."--- Galileo

This profoundly insightful quote has endured for nearly 400 years and has flourished through widespread acceptance. It is impossible to perpetuate any mythology, paradox, or deception (e.g., such as junk science or mythology in the existing theoretical foundation) without flagrantly violating sacred fundamental scientific principles, such as the deliberate suppression of crucial evidence by unscrupulous reviewers.

In both the judicial system and science, one of the most despicable acts is deliberately ignoring or suppressing crucial evidence, which is a violation of the most sacred fundamental principle. It is because the noble goal of both is to uncover the truth, objectively with an open mind, and it is impossible to achieve this noble objective if crucial evidence is deliberately ignored or suppressed by the participants in this noble endeavor.

In the judicial system, deliberately ignoring or suppressing crucial evidence is deemed morally reprehensible since it is a grave violation of the most sacred principle, which inevitably results in blatant miscarriages of justice. When such misconduct becomes widespread and commonplace, it precipitates a crisis within the legal system, completely eroding public trust. Those lawyers, jurors, crucial witnesses, police, and judges who aid and abet such wrongdoing by willfully disregarding or concealing vital evidence to obstruct justice are the vilest elements within the legal framework.

In the context of science, the moral and ethical duty of every scientist is to uncover the truth and reality by acquiring and utilizing valid data and evidence. Since any honest scientist participating in a debate with an open mind will reach objective conclusions based on valid evidence and data, there is no possibility of disagreement among participating scientists if there is no surreptitious agenda or dishonesty among some of the participants: *http://componentology.org/Vid/2 or http://componentology.org/Vid/3.***

There is no possibility for hostile disagreements among participants when all the participants fight on the side of science/justice to find the truth/reality, which serves the best interests of our national economy, public health, students, and the research community. In my humble opinion, participants who deliberately ignore and suppress crucial evidence are evil scum since they are striving to prevent the truth from being known.

"The scientific method is doing whatever it takes to not fool yourself into thinking something is true that is not, or into thinking that something is not true that is."

- Neil DeGrasse Tyson

25. Tests for Body of Knowledge (BoK) of Disciplines

According to lumenlearning.com and several other sources, all scientific ideas must be testable. If it's not testable and not at least potentially falsifiable, then there is no way to improve understanding, which is the goal of science — to seek out the imperfections/flaws in our current perception and correct them. You can't do that if there is no way to know fact from falsity, which direction would be correct, or what parts are in error and need to be corrected.

There must be verification and testing, critical exposure to scrutiny, peer review and assessment, and these are the ways to objectively check and recheck, repeatedly, so that we don't get fooled into believing something that is not trustworthy/true. And what we already accepted must still continually be reevaluated and, if possible, refined and improved or, if not, exposed and rejected. If it's true, in whole or in part, then the validation or evidence will support it, but if it's false, then there must be some way to show that too, or at least show what parts are false so that we can correct them.***

As cosmologist Carl Sagan said, "The suppression of uncomfortable ideas, data, or evidence may be common in religion or politics, but it is not the path to knowledge, and there's no place for it in the endeavor of science." To quote Sagan again, arguments from authority are worthless. Whatever is inconsistent with the evidence (or untestable), no matter how fond of it we are, must be discarded or revised. And according to the National Center for Biotechnology Information, some of the most basic principles of science are respect for the integrity of knowledge, collegiality, honesty, objectivity, and openness. Openness includes a lack of secrecy and collegiality refers to honest cooperation in good faith and integrity.

*** http://Componentology.org/Vid/7 & http://Componentology.org/Vid/8

Over centuries, numerous tests have been developed and refined to validate the quality and integrity of the BoK in each academic discipline. The BoK encompasses various types of knowledge, including descriptions, concepts, theories, and explanations. Today, every mature and widely practiced academic discipline employs proven tests tailored to its specific needs, ensuring the quality and integrity of its BoK through testing and falsification. Similarly, it is crucial to have established tests and methods to unequivocally confirm when the BoK of a discipline is junk science.

I have generally categorized disciplines into three broad categories: (i) hard sciences, (ii) soft sciences, encompassing disciplines requiring trustworthy BoK, and (iii) junk sciences or faith-based belief systems, like religious literature or mythology.**

The cardinal rules and tests for determining hard scientific knowledge include consistency, objectivity, empiricism, testability, and falsifiability. Valid scientific knowledge must satisfy all five rules. However, the existing BoK in the theoretical foundation for software engineering, seen in Exhibits B1 to B4, violates every one of these rules. Any scientific knowledge that violates even one of these rules is invalid.

For the BoK of any soft science or academic discipline to be considered trustworthy, it must adhere to principles such as a high degree of consistency, objectivity, evidence, testability, and falsifiability, with limited tolerance for errors or deviations. Every academic discipline needs to strive its best to adhere to the principles of the proven tests (e.g. tools and methods tailored to its specific needs), to acquire and validate each piece of knowledge. Each part or piece of knowledge in the BoK must strive to achieve a high degree of consistency and should be subject to rigorous validation through testing and falsification.

Proven tests to confirm that the BoK (comprising of multiple parts/pieces of knowledge such as descriptions, theories, and concepts) of a discipline is junk science beyond doubt include inconsistencies among its parts, a high degree of subjectivity or ambiguity, contradiction with known evidence/data, and the lack of acceptable methods (e.g., tools and techniques tailored to its specific needs) for validation through testing and falsification. In the case of junk science, experts in the discipline often resist challenging the validity of foundational assumptions or beliefs, perceiving it to be heretical.

In the context of the BoK (comprising of multiple parts or pieces of knowledge such as descriptions, theories, and concepts) of any academic discipline, any piece or part of knowledge is deemed untrustworthy if it cannot be validated through testing and falsification by utilizing tools, methods, and techniques tailored to its specific needs. It is strongly recommended not to rely on or use such untrustworthy parts or pieces of knowledge for logical reasoning or to reach any conclusions. Any discipline touted as a scientific field is undeniably junk science if it either justifies or tolerates utilizing such untrustworthy parts or pieces of knowledge to draw crucial conclusions.

26. Debunking the Most Egregious Myth

The nature and essence of any academic discipline are determined and shaped by the knowledge about the entities that form the core of the discipline. These entities are meticulously and systematically studied to accumulate more knowledge and gain a deeper understanding. In economics, this core comprises of economic principles

** http://Componentology.org/Vid/8

and theories grounded in relevant economic data or evidence. Accounting is centered around accounting principles and tax codes, while chemistry rests upon scientific principles and theories supported by evidence and experiments about the composition of matter.

None of these disciplines can be considered branches of mathematics by any stretch of the imagination, although they often utilize mathematics and logic extensively. This is particularly apparent when compared to the theoretical foundation required for applied research in CBSE (Component-Based Software Engineering), which entails understanding the nature of components, CBD (Component-Based Design), as well as the internal structure and anatomy of CBPs (Component-Based Products).

At the core of any discipline that employs CBE (Component-Based Engineering), such as mechanical, aerospace, and electronic engineering where every large physical product is constructed as a CBP, is contained knowledge and understanding of two essential entities: (i) the components themselves, encompassing their properties, internal features, functionality, and built-in mechanisms for their composition to construct CBPs, and (ii) the structure, anatomy, design, and construction of CBPs. This knowledge, crucial for understanding CBE, undoubtedly falls well within the realm of hard sciences.

Based on my understanding of mathematics, I can confidently assert that the descriptions and concepts found in the current theoretical foundation (e.g., in Exhibits B1 to B3) which address properties, attributes, functioning, features, and built-in mechanisms for their composition cannot possibly be classified as mathematics. Academic disciplines such as hard sciences and soft sciences operate on principles where theories, descriptions, concepts, and explanations must be firmly rooted in valid data and evidence.

Reviewing the existing inconsistent descriptions and opinions, as exemplified in Exhibits B1 & B3, may lead one to perceive each description as portraying a mythical entity termed as a component. Within these descriptions, so-called components are depicted with detailed accounts of their properties, attributes, internal features, functionality, and built-in mechanisms for composition to integrate with other software parts. While descriptions using attributable characteristics such as internal features, functionality, or mechanisms certainly cannot be found in the literature for mathematics, such attributable characteristics for entities are commonplace in scientific, fictional, or religious literature.

Valid, testable, and falsifiable descriptions aimed at objectively understanding components and CBPs undeniably belong within the realm of science, serving as crucial theoretical foundations for conducting research in CBSE. However, the existing descriptions and concepts prevalent today, such as those discussed in Exhibits B1 & B3, cannot be classified as branches of science. These descriptions lack grounding in any data or evidence and are devoid of testability and falsifiability based on evidence or experiments. Moreover, they exhibit inconsistency and are highly ambiguous, biased, or rooted in subjective beliefs or myths. Hence, these descriptions and concepts exemplify instances of junk science (akin to fiction or religion).

Is this not sufficient to prove that the existing theoretical foundation for software engineering is mythology or junk science: In debates like those found in Exhibits B1 and B3, where descriptions of entities encompassing properties, internal features, functionality, and built-in mechanisms are sought, the entity in question must either be a real entity or a fictitious one. In the case of descriptions for real entities, each description can be validated by acquiring and utilizing data, evidence, or experiments. However, if it is not possible to acquire and utilize such data, evidence, or experiments for testing and falsification, then the description must pertain to a fictitious entity. Hence, the inconsistent and baseless descriptions such as those in Exhibits B1, B2 & B3 must be attempting to describe a fictitious or mythical entity.

The objective of this section is to debunk the most popular and insidious misconception that the entirety of Computer Science is merely a branch of mathematics. This egregious misconception must not be blindly extended to the specific sections of computer science that furnish the theoretical foundation, encompassing theoretical knowledge and an objective understanding of real components, their mechanisms, and CBD/CBPs, crucial for conducting applied research in software engineering and CBSE.

Almost every book or literature on software components and CBSE (see Exhibits B2 & B4), comprising of inconsistent descriptions, concepts, and subjective explanations or opinions, reads like fiction about fictitious entities referred to as components rather than any kind of mathematics or logic. On the other hand, one of the defining features of mathematics is its precision and rigor. Unlike other forms of human knowledge, mathematics relies on strict rules and logical deductions to arrive at conclusions.

Theorems, proofs, and axioms form the basis of mathematical reasoning, ensuring that results are consistent and reproducible. Mathematics encompasses not only the use of highly precise expressions, formulae, equations, theorems, and axioms to represent or describe knowledge but also the application of mathematical methods and logical reasoning to generate new insights and testable knowledge. None of these elements or applications of methods are found in the theoretical foundation required for CBSE.

None of these mathematical elements or applications of mathematical methods are found in the theoretical foundation used to conduct research in CBSE. The existing theoretical knowledge (see Exhibits B1 to B3) neither uses mathematical expressions, formulae, equations, theorems, and precise axioms to represent or describe knowledge nor applies methods (see Exhibit B4) to generate new knowledge and insights.

The vocabulary, terms, steps, and notations used in the language of mathematics encompass expressions, formulae, equations, theorems, precise axioms, proofs, and more. Conversely, the terminology or vocabulary used to describe aspects such as properties, functionality, features, attributes, and built-in mechanisms—which can only be employed to provide descriptions of real or fictitious entities—generally falls outside the confines of mathematical language and vocabulary. Since Exhibits B1 and B3 are not describing any kind of real entities, they must be describing fictitious or mythical entities.

What are those researchers describing in Exhibits B1 & B3 on page 22 & 26 respectively? Do these represent precise or objective and consistent descriptions of any tangible entities? If so, what are they? Since such entities do not exist, I inferred that they must be describing some form of fictitious or mythical entities. Only ignorant or fake scientists justify relying upon such mythology in the existing theoretical foundation in Layer-1 to conduct engineering research in Layer-2.**

The existing description such as those in Exhibits B1 and B3 describing some kinds of fictitious entities and egregiously calling them as components. This has been giving false impression that software engineering already has been using software components.

** Refer to Layer-1 & Layer-2 in Table-1 on page 6

27. Exhibit A3: Research Questions (RQs) for Componentology

Question-1: Name a product that is certainly a CBP (Component-Based Product)?
Answer-1: Cars, Computers, Airplanes, and Machinery for factories.

Question-2: Why is each of these products a real CBP?
Answer-2: Each car, computer, or airplane is built by assembling multiple components (as in FIG-2). No product can be a CBP if it is not built by assembling multiple components.

Question-3: Name a physical product that is certainly not a CBP?
Answer-3: Houses, Buildings, Bridges, and software products.

Question-4: Why is each of these not a CBP (Component-Based Product)?
Answer-4: Each house or building is not built by assembling multiple components. That is, each product is built (as a monolith as in FIG-1) using reusable ingredient parts such as cement, steel, plastic, paint, metals, and alloys.

Question-5: What are the striking differences between real-CBPs and non-CBPs?
Answer-5: Any product can be a CBP if and only if the product is built by assembling multiple components. For example, any product (e.g. cars, airplanes, computers) that can be disassembled into components (and if the product can be rebuilt by reassembling the components) is certainly a CBP. However, any product (e.g. a house, skyscraper, software product, bridge) that can't be disassembled (e.g., to harvest components, which are in working condition to assemble them in building other products) is certainly not a CBP. Hence, engineering disciplines such as Mechanical, Computer, Electronics, and Aerospace are certainly building products as CBPs, while engineering disciplines such as Civil and Software are certainly not building products as CBPs.

Research Questions to comprehend differences between real-CBE & non-CBE

Question-6: Name an engineering discipline that is certainly employing real-CBE to design and build each large product as a CBP.
Answer-6: Mechanical, Aerospace, Electronics, & Computer Engineering disciplines.

Question-7: How can you say that each of them is employing real CBE?
Answer-7: Each discipline is building products as real CBPs as in FIG-2. No discipline can be a real CBE if it is not building large products as CBPs.

Question-8: Please name an engineering discipline that is certainly not employing real-CBE (i.e. which discipline is not designing & building products as CBPs)?
Answer-8: Houses, Buildings, and Bridges are examples of non-CBPs, where non-CBP implies a product that is not built by assembling multiple components. So, civil engineering is not employing real-CBE.

Question-9: Why is civil engineering not CBE (Component-Based Engineering)?
Answer-9: Each house or building is not built by assembling multiple components. That is, Civil and Software engineering have been building products as non-CBPs.

Question-10: What are the striking differences between real CBE and non-CBE?
Answer-10: Any engineering discipline can be a CBE if and only if the discipline can build each

large product as a real CBP (i.e. by assembling multiple components). For example, disciplines such as Mechanical, Aerospace, Electronics, and Computer are examples of real-CBE, since each of them builds every large product (e.g. a car, airplane, computer) as a real-CBP. However, not every product (e.g. house, skyscraper, software product, bridge) is a CBP. Hence, engineering disciplines such as Civil and Software are not employing CBE, since they are not building products as CBPs.

Note: Many software researchers & engineers claim to be leading experts in Software Components, CBPs, and CBE for software, so it is expected that they at least have an elementary knowledge and understanding of existing theories and concepts about Components, CBPs, methods for CBE, and proof for the theories and concepts. Anyone who considers themselves to be a scientist or researcher must not have any problem answering the elementary questions below about physical Components and physical CBPs if they are honest and have integrity.

Question-11: Which engineering disciplines practice real CBE? In other words, which engineering disciplines can build any large or complex product as a real CBP?
Answer-11: Mechanical, Electronics, and Aerospace engineering disciplines.

Question-12: Which engineering disciplines today do not practice real CBE? In other words, which disciplines today do not build large or complex products as CBPs?
Answer-12: Civil, Chemical, and Software engineering disciplines.

Galileo said "Names and attributes must be accommodated to the essence of things, and not the essence to the names, since things come first and names afterwards." Calling a pig a rose does not make the pig smell any better. The software community has been practicing fake CBE using fake components and is fooling itself and the rest of the world that they are practicing CBE using real components.

Dr. Alan Key, who won the AM Turing Award (the Noble Prize in computing) and is considered the father of OOP (Object Oriented Programming), compared software engineering with the Egyptians building the pyramids about 4500 years ago. ***See Section 36.1 on page 63.*** It is an indisputable fact that every software product today is built as a non-CBP (as in FIG-1). That is, it is impossible to find even a single large software product that is designed and built as a real CBP (as in FIG-2). Today, no one else even knows the above basic facts about real CBPs & real Components.

28. Destination Between Academic Disciplines & Mythology

The intention and purpose of every academic discipline, such as economics, sociology, linguistics, anthropology, botany, zoology, psychology, and political science, is to accumulate the necessary Body of Knowledge (BoK) to gain an understanding of its subject matter. There can be multiple subject matters (e.g., subfields) associated with each academic discipline. Each discipline needs to have well-established, viable, and appropriate methods and tools for accumulating and utilizing the necessary corpus of data and evidence to support the understanding and validate its BoK.

Methods for acquiring each piece of knowledge in the BoK include systematically analyzing the relevant corpus of data and evidence to draw conclusions or proposing hypotheses and rigorously validating them through testing and validation using the relevant corpus of data and evidence. In other words, accumulating and utilizing a valid and verifiable corpus of data and evidence plays a vital role in acquiring each piece of knowledge. Each piece of knowledge, such as a theory proved through rigorous validation (through testing and falsification using data and evidence), is valuable for making predictions, providing explanations, or gaining deeper insights and wisdom.

Any piece of knowledge presented or proposed lacks credibility and is unreliable junk if there is no relevant data and evidence for its validation (through testing and falsification, in case it is false). There is no verifiable data or evidence available to support the existing descriptions and concepts, such as those found in Exhibits B1 to B4. Proposing such descriptions and concepts without any supporting data or evidence is pure speculation, which is junk. It is not possible to collect the necessary verifiable data and evidence to support descriptions and concepts for mythical entities such as angels, demons, Pegasus/alicorn, or fictitious entities that are deceptively disguised as components and CBPs.

The primary distinction between the BoK of an academic discipline and that of mythology/fiction is the availability of necessary, relevant, and verifiable corpus of data and evidence to support and validate (through testing and falsification, in case it is flawed) each piece of knowledge in the BoK.

In the case of the "BoK for CBSE," there are no established, viable, or appropriate methods and tools to accumulate and utilize the necessary corpus of data and evidence to support or validate various pieces of knowledge widely being used, such as the descriptions and concepts known today. For example, this is evident since none of the renowned experts have provided any verifiable data or evidence to support their descriptions or concepts, such as those in Exhibits B1 to B3.

Cartoon: We ostracized the brown guy sitting alone over there for advocating science and scientific descriptions, over the divine mythology that was revealed to the high priests of computer science more than 50

years ago (e.g., such as the 1968 NATO Software Engineering Conference).

We cannot allow that guy to bring science that exposes the cult faith that we have all meticulously crafted, and prevents us from successfully indoctrinating unsuspecting software practitioners and impressionable software students into submitting to our cult faith like we have for over 50 years.

That brown guy violated an unwritten rule by challenging the false beliefs (at the root of junk science) propagated by our renowned white scientists. Such action by a person of color is unscientific, unscholarly, and taboo in the circles of our cult/KKK.

29. Obvious Fact: The Existing BoK for CBSE is Junk Science

It is an undeniable, if not tacitly admitted, fact that the existing 'BoK for CBSE' (see Exhibits B1 to B3) is centered on fictitious entities referred to as components and CBE. In legal contexts, accepted and uncontested facts require no further proof. Many renowned software experts have proposed multiple inconsistent descriptions, such as those seen in Exhibits B1 and B3, for software components in CBSE**.

The discussion in the exhibits closely resembles subjective and/or ideological opinions rather than an academic discourse that relies on principles of evidence and sound logic to seek out trustworthy truth/reality, which should be evidence-based, testable, and falsifiable. It is embarrassing that those researchers may be subconsciously aware of the fact that these subjective and inconsistent descriptions (that are their opinions) — influenced by their ideological biases, received beliefs, or prejudice—represent junk science.

Each expert proposing a description must provide honest answers to these simple questions: Is your description a valid assertion or merely an unsubstantiated opinion? If you claim that your description is valid, can you provide valid evidence, scientific rationale, or logical justification to support its validity? If your description is valid, can you assert or prove that all other descriptions that contradict your description are invalid?

Can you provide any evidence to demonstrate that other descriptions contradicting yours are false? Is your description testable and falsifiable using established methods? Promoting personal opinions or beliefs as valid academic knowledge is unethical and constitutes academic misconduct. In essence, presenting opinions as reliable academic knowledge is dishonest, morally wrong, and a breach of academic integrity as it deceives others into relying on such information for further research.

After reviewing numerous inconsistent descriptions and engaging in discussions with esteemed experts, any scientist with basic common sense would logically conclude that every description is merely a belief, opinion, or unproven hypothesis. Through conversations with many experts, it becomes apparent that these descriptions lack scientific evidence or rationale. Furthermore, none of the descriptions are testable or falsifiable, rendering them unreliable and akin to junk science.

In the context of descriptions for software components and Component-Based Engineering (CBE), when multiple researchers propose differing descriptions that lack empirical evidence and are inconsistent with each other, it is an undeniable fact that each

** Exhibits B4 in sections 15 on page 28 of the descriptions is a mere belief, opinion, or unproven hypotheses. Consequently, it is unscientific and unscholarly to treat them as trustworthy knowledge, and it is a mistake to rely on such unproven and untrustworthy inconsistent descriptions to conduct applied or engineering research in this field.

In the realm of software components and CBE for software, when multiple researchers propose varying subjective and ambiguous descriptions lacking empirical evidence and coherence, it becomes evident that each of these descriptions is merely a belief, opinion, or unproven hypothesis. Consequently, it is unscientific and unscholarly to regard them as trustworthy academic knowledge. Relying on such inconsistent and unproven descriptions for applied or engineering research in this field is misguided.

Hence, it is paramount to prioritize the search for the most accurate and empirically supported descriptions. Only through rigorous evaluation of the evidence and refinement of our understanding can we establish a foundation of reliable and trustworthy knowledge for applied or engineering research. Relying on any of those inconsistent and unproven descriptions as trustworthy knowledge for conducting applied or engineering research is a mistake and a fool's errand, comparable to relying on junk science to conduct applied research in fundamental sciences such as physics, biology, or chemistry.

It is an undeniable fact among the research community that it is a fatal mistake to conduct applied research by relying on unproven and untrustworthy beliefs or opinions, considering them to be trustworthy academic knowledge. It is an indisputable fact that it is unscientific and unscholarly to rely on baseless opinions or myths, treating them as trustworthy academic knowledge (or self-evident truths) to reach conclusions of vital importance with far-reaching implications.

29.1. Paradigm for a Virtual World & Paradigm for a Fictitious World

In the context of a paradigm of a fictitious world, there are no prohibitions on describing and creating any type of fictitious things and giving any labels (or names) to them, such as components** and CBPs (Component-Based Products). The existing deeply entrenched and widely practiced dominant paradigm for CBSE is an example of such a paradigm of a fictitious world, which is incontrovertibly based on "junk science".

In the context of a paradigm of a virtual world, it is necessary to create accurate virtual representations of physical or real entities by objectively finding valid descriptions and concepts to portray each real entity accurately. This necessitates gaining a nuanced grasp of the underlying principles and mechanisms governing each of the entities intended to be represented virtually. The newly proposed paradigm for CBSE, based on the hard scientific knowledge of Componentology, is an example of such a virtual world.

The software community, or the software engineering world, has been practicing the paradigm for the fictitious world in CBSE since 1968, which is the root cause of the infamous software crisis. Pioneer-Soft introduced, in CBSE, a new paradigm for the virtual world by developing and relying on Componentology. This approach aided in making necessary innovations that effectively tackled the notorious software crisis.

Our patented inventions offer empirical evidence that unequivocally demonstrates the superiority of the paradigm for the virtual world (based on hard science) over the paradigm for the fictitious world (based on junk science). This proves beyond any reasonable doubt that the paradigm for the virtual world (based on hard science) is the right path, while the paradigm for the fictitious world (based on junk science) is incontrovertibly the wrong path.

** Exhibits B1 & B3 in Sections 11 & 13 on pages 22 & 26

In the paradigm of a fictitious world, there are no restrictions on creating depictions of anything and assigning any name to it. For instance, one can create depictions of a bird, fruit, tree, or fish, and label any of them as a type of elephant. In contrast, in the paradigm for a virtual world, every virtual depiction of a real entity must aim to accurately resemble the appearance of the real entity. For example, creating depictions or pictures to portray elephants needs knowing of their appearance and distinctive attributes, so that the pictures of elephants resemble actual animals.

Today, various kinds of software parts (possessing any perceived useful properties such as standardization for reuse) are misleadingly labeled as a type of component in software, disregarding essential and distinguishing features, inbuilt mechanisms, and attributes that are necessary for any part to be considered a component.

This kind of mislabeling can occur when experts are ignorant of the essential functionality and attributes of real components, which are vital building blocks to building CBPs. It's akin to misleadingly calling other types of animals (e.g., species such as birds, fish, reptiles, rodents, snakes) a kind of sub-species of elephants, by those who have never seen elephants and are ignorant of their unique distinguishing attributes and apparent features.

30. Theoretical Knowledge in Layer-1 & its Application in Layer-2

There is a symbiotic relationship between the BoK in Layer-1 (Section 1) and the applied research in Layer-2**, which applies the theory available in Layer-1 to invent useful things or solutions. There must be theoretical knowledge in Layer-1 because when there is no theoretical knowledge in Layer-1, there is nothing to apply in Layer-2. Layer-2 cannot float in thin air. Applied research in Layer-2 needs a trustworthy theoretical foundation in Layer-1. Creating & using mythology as the theoretical foundation in Layer-1 is a fatal mistake.

It is an undeniable fact that the existing theory for software engineering in Layer-1 is mythology or toxic junk science. Let's examine a few popular descriptions known today for components in Exhibits B1 & B3.

To unequivocally classify the BoK of a discipline as junk science, fiction, or mythology, what specific criteria, conditions, or rules must it satisfy? The current BoK for CBSE, containing descriptions and concepts of components, CBPs, and CBE, satisfies nearly all such valid criteria, conditions, and rules. On the other hand, the BoK for Componentology satisfies every rule, criterion, and condition to qualify as scientific knowledge.

These and other known descriptions, provided by renowned experts on software components and widely used as the theoretical foundation, are junk science by any conceivable academic or scholarly standard. For any knowledge to be deemed academic or scholarly, it must meet certain minimum requirements or criteria. However, the existing BoK for CBSE, comprising of many descriptions, theories, and concepts, fails to meet even these minimum standards/criteria to qualify as academic or scholarly knowledge.

It is obvious that the conflicting descriptions or disagreements, such as baseless suggestions discussed and debated in Exhibits B1 & B3, are merely untested ideas or hypotheses expressed by renowned software

** Refer to Layer-1 & Layer-2 in Table-1 on page 6

experts & researchers about fictitious entities misleadingly referred to as components.

Although such disagreements may be acceptable regarding hypotheses that are not yet proven or accepted at the cutting edges of the BoK of a discipline, they are not acceptable as its core first principles (as seen in Exhibits B1& B3**). These principles serve as the fundamental building blocks at its foundation, essential for constructing the discipline's edifice.

Any honest scientist with integrity who reads these inconsistent and ambiguous descriptions with an open mind cannot deny the fact that they are not describing any kind of real entity but rather some kind of fictitious entity. Is it not a fatal mistake to use such baseless ideas as the theoretical foundation in Layer-1 to conduct applied engineering research in Layer-2?

It is an indisputable fact that the theoretical foundation of any engineering discipline must be grounded in science. Rigorous academic and scientific principles, such as consistency, unambiguousness, objective validation through viable proven methods to test, & falsification, are valued for their effectiveness in creating and sustaining a reliable theoretical foundation. These principles must serve as the bedrock and foundation of engineering disciplines. Those who advocate or justify the use of mythology (in Exhibits B1 to B4) as the theoretical foundation are either misguided or ignorant of even the elementary principles of engineering research.

In the current landscape of software or computer science, finding descriptions of components, the structure of CBPs, and essential mechanisms for CBE that even remotely meet scientific standards is nearly impossible. Hence, such descriptions cannot be considered scientific by any stretch of the imagination. Many experts are subconsciously aware of this fact and tacitly admit it. Consequently, the descriptions and concepts used today in the theoretical Layer-1 must be deemed unscientific junk.

This reality is further established and evident by the fact that no known description today can be considered scientific by any stretch of the imagination. Virtually no description or concept within the existing theoretical foundation pertains to any tangible reality; instead, they refer to fictitious entities deceptively portrayed as components, CBPs, and CBE. In essence, nearly every description or concept known today concerns a fictitious entity rather than a tangible reality.

Just as Biomimetics draws inspiration from nature's wisdom, derived from billions of years of evolution, my research on CBSE is guided by insights gleaned from centuries of wisdom accumulated by mature component-based engineering practices such as mechanical, aerospace, and electronics. Componentology is created to accumulate the essential theoretical foundation and insights, which provide a vital foundation to understand and utilize the wisdom amassed by these mature engineering disciplines. Please refer to section 31 on page 54 which summerizes the dawn of primitive prehistoric component based products.

Today, it is impossible to find any description, such as those found in Exhibits B1 & B3, that can be considered objective or based on scientific investigation by any stretch of the imagination. Hence, every description known today describes a fictitious entity, which makes it junk science. In contrast, the descriptions in Exhibit A1 (in page 8) closely align with objective reality since they are based on rigorous scientific investigation and valid evidence such as variable observable aspects of physical entities. Hence, Componentology is a real science.

** Exhibits B1 & B3 in Sections 11 & 13 on pages 22 & 26

31. The Dawn of Component-Based Products

The anti-science cult at NSF.gov and other federal agencies have been wasting hundreds of millions of dollars each year on SPSQ (Software Productivity, Sustainability, and Quality) without recognizing the single greatest invention of all time for enhancing productivity, quality, and sustainability through economic, effective, and efficient maintenance. This greatest invention refers to a very specific kind of part that can be easily assembled, disassembled, and replaced. These very specific replaceable parts are widely known and referred to as components. Any part that cannot be assembled or plugged in (as depicted in FIG-2) should not be considered a component.

A replaceable component of a CBP implies a part that can be assembled and disassembled to replace it easily, as and when needed (to service the part) throughout the useful lifespan of the CBP: ***Please refer to Section 32 on page 55.***

The earliest CBPs were crafted in the Stone Age by cavemen, who assembled properly crafted primitive components such as properly shaped stones, sticks, and strong ropes/threads possibly made from leather. The livelihood and survival of cavemen depended on creating and maintaining rugged and high-quality CBPs such as spears, hammers, and axes. For instance, if one of the components broke, it could be replaced with a new or improved component.

Naturally, the cavemen must have tried to innovate to not only create high-quality, rugged CBPs but also sustain and maintain the CBPs at a lower cost for their survival and prosperity. Also, the cavemen must have salvaged and reused unbroken good-quality components from damaged CBPs.

The defining characteristic of a bird is its ability to fly, just as the defining characteristic of a component is its conduciveness and suitability for assembly (or plugging in), as depicted in FIG-2. In other words, the defining mechanism of a bird lies in its wings and its ability to fly. Likewise, the defining characteristic of a component lies in its inherent mechanisms that facilitate quick and easy assembly, as shown in FIG-2 on page 8.

Remarkably, no one else in the software world has created a real CBP like the one depicted in FIG-2, which requires inventing and utilizing pluggable components. Sadly, even these primitive inventions cannot be made (in Layer-2**) without first having a basic knowledge of valid and objective

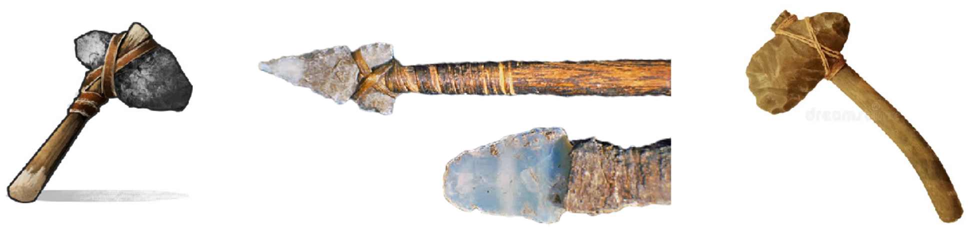


Figure 6: Primitive CBPs, built by assembling replaceable components, to provide service access

Along with fire and the wheel, the CBP, built by assembling replaceable components, is one of the three great prehistoric inventions.

** Refer to Layer-1 & Layer-2 in Table-1 on page 6

descriptions for CBPs, components, and mechanisms for CBE. Unfortunately, the anti-science cult has been insulting and ostracizing me for my struggles to expose the existing junk science (see Exhibits B1 to B3) and get them to comprehend valid and objective descriptions (see Exhibits A1**).

Unfortunately, the anti-science cult has been insulting and ostracizing me for my struggles, such as exposing the existing junk science (see Exhibits B1 to B3) and requesting them to comprehend reality, known since the dawn of CBPs and CBE in the Stone Age, through valid and objective descriptions (see Exhibits A1). I will withdraw my lawsuit, tender an unconditional apology, pay penalties, and face criminal prosecution if anyone else in the software world can produce evidence that they understand the reality represented by valid objective descriptions (see Exhibits A1) and have taken actions based on this understanding to create real CBPs.

Even today, certain parts only cost US$10, but cost US$500 and 10 man-hours to replace them. Such parts are designed to the highest quality standards to work for the lifetime of the product. But a small number (e.g. one in hundred thousand) of such components fail because of human errors or quality control problems (e.g. using of poor-quality ingredient metals, alloys, or poor installation at the assembly line). No one wants to waste weekends at repair shops for such unpredictable breakdowns. In the case of software, every component needs a redesign, so can use the best service access.

Today, each large software application or product is built as in FIG-1 on page 8, by merging code for many self-contained modules. In the case of real-CBE, each software product is built as a CBP, which is built by plugging-in multiple real components as in FIG-2. The code for each self-contained module (of a CBP) is implemented as a pluggable software component and plugged-in (as in FIG-2), so it can be replaced in a few minutes. But, today the code for each self-contained module is merged into the code of the product (see FIG-1), so it takes many days or even weeks to replace the code for the same self-contained module and to test the whole product, since the product is built as FIG-1 (i.e. by merging code for multiple self-contained modules).

32. Superior "Service Access" to Each Component is Desirable

The designers of products such as automobiles, airplanes, computers, and large machines are obsessed with providing outstanding "service access" to components (i.e. very specific kinds of parts that are assembled), which are prone to wear and tear and require periodical maintenance. Service access allows repair shops to easily repair an old broken component, or replace it with a new component.

In the case of computers, during the 1980s and 1990s, we used to extend the life of old computers by installing more powerful CPU, faster DRAM, or bigger hard drives, etc. Personal Computers (or PCs) were expensive in those days, so many people used to replace older Intel 386 processors with newer 486 CPUs. In software, we could use "service access" to every component, because 80% of software engineering is changing existing code (of one or more components in FIG-2 on page 8). If each software product is built as CBP as illustrated in FIG-2, every component can be

** Refer to Exhibit A1 in Section 4 on page 8 easily replaced by a better component (i.e. by redesigning it).

For example, certain kinds of parts (e.g. tires, break-pads, oil-filters, spark-plugs) need to be replaced periodically (e.g. every 10,000, 25,000, or 50,000 miles). So, it is highly desirable to minimize the cost and time at the mechanic shop to replace such components, which need to be serviced periodically (e.g. as per recommended maintenance guidelines). How would consumers feel if it takes two days to replace spark-plugs or oil filters? I once read that replacing spark-plugs in a certain 1940s Ford model needed removing its engine. Those spark-plugs were put in a hard-to-reach place, and so could not be removed without taking out the engine.

Even today, certain parts only cost US$10, but cost US$500 and 10 man-hours to replace them. Such parts are designed to the highest quality standards to work for the lifetime of the product. But a small number (e.g. one in hundred thousand) of such components fail because of human errors or quality control issues (e.g. use of poor-quality ingredient metals, alloys, or poor installation at the assembly line). No one wants to waste weekends at repair shops for such unpredictable breakdowns. In the case of software, every component needs a redesign, and can use the best service access.

Today, every large software application or product is built as in FIG-1, by merging the code for many self-contained modules. In the case of real-CBE, each software product is built as a CBP, which is built by plugging in multiple real components as in FIG-2. The code for each self-contained module (of a CBP) is implemented as a pluggable software component and plugged in (as in FIG-2), so that it can be replaced in a few minutes. But today, the code for each self-contained module is merged into the code of the product (see FIG-1), so it takes many days or even weeks to replace the code for the same self-contained module and to test the whole product, since the product is built as in FIG-1 (i.e. by merging code for multiple self-contained modules).

33. Documented Evidence: Men May Lie but Documents Don't

History and all necessary evidence are well-documented, and no one can erase well-documented history or make the evidence disappear. Thousands of research papers and textbooks have documented the existing Body of Knowledge (BoK), which encompasses descriptions, concepts, and explanations, widely used in the current theoretical foundation in Layer-1**. This documentation includes descriptions found in Exhibit B1 and B3, as well as numerous books like Exhibit B2, along with 1231 research papers on CBSE referenced and analyzed in Exhibit B4.

No scientist of sound mind can deny the fact that the existing BoK is junk science. After reviewing multiple inconsistent, ambiguous, and puzzling descriptions and concepts, it becomes evident that the BoK meets almost every criterion for being classified as junk science. For instance, any scientist can identify numerous descriptions for components that are substantially inconsistent with each other, contrary to evidence, illogical, and lacking scientific rationale. Any scientist, particularly one with a doctorate in science, would be deemed insane and unfit if they were to assert that such unequivocally inconsistent and enigmatic descriptions and concepts are not junk science.

** Refer to Layer-1 & Layer-2 in Table-1 on page 6

I have accumulated a huge BoK comprising descriptions, concepts, mechanisms, and explanations for Componentology, which have been documented in various sources for over two decades. These include eight granted patents, papers posted on third-party websites, and emails exchanged with hundreds of software researchers during this time. Neither I nor anyone else can tamper with this documented evidence. No rational-minded scientist can deny the fact that Componentology is a valid hard science. The BoK created for Componentology successfully passes all tests, rules, and criteria to be qualified as a hard science.

The following two facts incontrovertibly substantiate our lawsuit, the allegations of which are presented in *Section-3 on Page 7 and Section-6 on Page 11.*

No qualified scientist who values honesty, ethics, and scientific integrity can deny or falsify the fact that the existing "BoK for CBSE" is junk science, based on the available documented evidence (e.g., Exhibits B1 to B4). Only the scum of the scientific world resorts to evil evasive tactics such as denying proven principles of the scientific method and deliberately disregarding crucial, well-documented evidence.

Furthermore, based on the well-documented evidence such as our granted patents, no qualified scientist who values honesty, ethics, and scientific integrity can deny or falsify the fact that Componentology (e.g., Exhibits A1 & A2**) constitutes valid hard science, representing significant progress in the right direction, akin to Kepler's laws, as illustrated in ***Section-11 on Page 20.***

Tunnel Vision (or Confirmation bias) Trap

Confirmation bias selectively seeks only favorable evidence while suppressing unfavorable evidence. Researchers falsely concluded 2300 years ago that the Earth is at the center and stopped looking/thinking outside the tunnel/box. This confirmation bias was insidiously contagious and spread for centuries. Similarly, software researchers falsely concluded 54 years ago that reusable parts are components, and that Component-Based Engineering (CBE) is the use of those so-called components, and researchers are refusing to look/think outside the tunnel/box. This confirmation bias is insidiously contagious, having spread for the past 50 years and become a deeply entrenched conviction.

It is an undeniable fact that the geocentric paradigm, which became the dominant paradigm by the 15th century, was junk science — nothing in the world can alter this truth. Similarly, the current theoretical foundation for software engineering — comprising of descriptions, concepts, and explanations about Components, CBPs, and CBE — is toxic junk science, and no scientist in the world can deny or falsify this verifiable fact.

It is inconceivable for any ethical scientist to refute this basic truth: The existing BoK encompasses descriptions, concepts, and explanations about Components, CBPs, and CBE, all of which constitute junk science by every conceivable criterion, standard, method, and measure. This reality is glaringly evident, such that even laypeople can recognize it through a cursory review of those existing inconsistent descriptions, concepts, and explanations.

Any scientist who attempts to deny such incontrovertible evidence under oath would face legal consequences, including being charged with contempt of court for dishonesty and lying under oath. If I fail to extract a confession from any such scientists, or if I am wrong, I am ready to face prosecution.

** Refer to Section 4 on page 8 & Section 10 on page 19

34. A Few Quotes Relevant to Paradigm Shifting Endeavors

"By denying scientific principles, one may maintain any paradox." - Galileo

This profoundly insightful quote has endured for nearly 400 years and has flourished with widespread acceptance. It is impossible to perpetuate any mythology, paradox, or deception (e.g. such as junk science or mythology in the existing theoretical foundation) without flagrantly violating sacred fundamental scientific principles, by unscrupulous reviewers deliberately suppressing crucial evidence.

Breach of scientific principles includes using baseless and inconsistent beliefs as core foundational axioms or first principles, and suppressing crucial evidence or uncomfortable truths. Only the scum of the scientific world justifies mistakes such as blatant violation of the basic principles of the scientific method.

As Carl Sagan said, "The suppression of uncomfortable ideas or evidence may be common in religion or politics, but it is not the path to knowledge, and there's no place for it in the endeavor of science." Darwin said that if you find any evidence that is contrary to what you already believe, you should record it as soon as possible, as people have great resistance to new evidence and their minds otherwise block it out.

Any valid verifiable evidence or truth that challenges deeply entrenched conventional wisdom is cherished by great scientists and sabotaged by fake scientists. The key aspect of almost every great paradigm-shifting discovery is exposing deeply entrenched false beliefs. The central tenet of the famous book "Zero to One" is: what revolutionary truth do you know that no one else agrees with?

"No problem can be solved from the same level of consciousness that created it." - Albert Einstein

The root cause for the infamous software crisis is the flawed consciousness (i.e., illusions) based on the existing mythology about mythical things being disguised as Components and CBE. Hence, it is impossible to address the software crisis without a consciousness based on the reality of Componentology. It is imperative to make every possible effort to avoid being influenced by widely prevalent myths, misconceptions, biases, and prejudices. Software researchers must ask themselves if they would make the same insulting comments about the BoK of Componentology if it were the year 1924, and they knew nothing about the existing illusions or mythology about mythical entities deceptively impersonating Components and CBE***.

The 1968 NASA Software Engineering Conference has generally provided foundational assumptions or axioms for software engineering. The foundational evidence or axioms are in the realm of science, and not mathematics: ***https://www.cs.dartmouth.edu/~doug/components.txt.*****

The indicative evidence and axioms used to formulate the hypothesis for CBSE 50 years ago were fundamentally flawed. The foundational axioms and basic assumptions are neither consistent with each other nor consistent with valid observable reality: http://real-software-components.com/CBD/main-differences.html

In the realm of axiomatic and formal systems, a fundamental rule reigns supreme: each system hinges upon a handful of

** http://Componentology.org/Ref/17

*** http://Componentology.org/Ref/10

foundational postulates or axioms. Even minor imperfections within these axioms can breed inexplicable inconsistencies, leading to unforeseen disruptions such as undetectable errors in the conclusions drawn from them. This phenomenon of inconsistencies persists despite the impeccable logic employed in deriving each conclusion.

Discovering imperfections or errors within the foundational axioms of a major and widely practiced discipline can lead to profound and paradigm-shifting discoveries. The transformative impact of such revelations and resulting discoveries would be proportional to both the number of active practitioners within the discipline and the total contribution of the discipline to the global economy.

Galileo Galilei aptly remarked, “Names and attributes must be accommodated to the essence of things, and not the essence to the names, since things come first and names afterwards.” This statement underscores the importance of aligning our understanding with the true nature of phenomena.

Galileo suggested that assigning essence to things/entities is foolish. The descriptions in Exhibits B1 and B3 by renowned software experts committed such foolish mistakes by assigning an illusory essence and fictional qualities to fictitious entities deceptively named Components or CBE. On the other hand, Componentology aims to uncover the true essence of real or physical entities such as components and CBPs, objectively with an open mind.

The existing descriptions in Exhibits B1 and B3 depict mythical entities masquerading as components. Each description attributes essence or properties to these mythical entities, misleadingly named components, perpetuating baseless biases and misconceptions. This approach is akin to putting the cart before the horse — it’s conducting science and research in a fundamentally flawed manner.

It should be obvious to any civilized community of scholars, academicians, and scientists that these kinds of descriptions and concepts (in Exhibits B1 to B3) are either baseless opinions or biased ideological preferences expressed and proposed in discourses about fictitious entities misleadingly referred to as Components and CBE.

Software researchers are suffering from Elephant Rope Syndrome: From the perspective of primitive or earlier technologies available between the years 1960 and 1990, software researchers have subconsciously concluded or been conditioned to believe that software is different, so there cannot be a viable alternative for pseudoscientific knowledge (e.g., descriptions and concepts for fictitious entities) in the theoretical foundation. This premature conclusion stems from the fact that the earlier primitive technologies available were incapable of creating virtual Components and CBPs ***(see Section-9 on page 16)***, and from the analysis of numerous failures until the early 1990s from the perspective of the technologies of the day. However, technological advancements have made it possible to create and use virtual components to build each software product as a virtual CBP *(as in FIG-2 on page 8).*

35. Detailed Summary & Conclusions

There is no alternative to the reality of Componentology, just as there was no alternative to the reality of Heliocentrism**. It is the only right path. When a discipline ends up as a flawed paradigm (or junk science), the only right path is to expose the flawed

** http://Componentology.org/Ref/9 & http://Componentology.org/Ref/10 paradigm (e.g., geocentric paradox). Likewise, based on my two decades of research, http://componentology.org/ is the only right path for software engineering.

I have created our patented solutions in Layer-2 by including and relying on Componentology in Layer-1. Today, no one in the software world can create and use components to build CBPs without using our patented tools, technologies, and methodologies. The reviewers of this kind of paper must accept the possibility that the things they have not yet seen can exist, even if they are unable to contemplate it.

One must look at the evidence objectively and with an open mind. Let me explain what is meant by evidence in the context of science. No one at the time could see the reality Galileo's telescope had shown him, without access to the telescope. No one at the time could see the reality of microbes Leeuwenhoek had seen using his microscope, without access to the microscope. At the time, no one else had access to the telescope or the microscope**.

When Galileo insisted on examining the empirical evidence, the scientific community of the day were offended and instigated the Vatican to prosecute him. The scum of the scientific world suppressed crucial evidence to justify their illusions. Leeuwenhoek offered to show empirical evidence. The scientific community of the day initially said that he was drunk and hallucinating**. That no one else had seen microbes did not necessarily mean that Leeuwenhoek was hallucinating.

I have been offering to show real CBD for over two decades. Fake scientists of the anti-science cult have assumed and been insisting for nearly two decades that I might be hallucinating. Even if the reviewers are despicable liars and have the blood of many innocent people on their hands, my lawyers advised me that it's illegal to compel them to examine the evidence through force. I was told that the only option available is to file an expensive and time-consuming lawsuit to force the scum of the scientific world to do their ministerial duty.

If anyone wanted to see the empirical evidence, they should have been open to using inventions such as telescopes or microscopes which were available from their respective inventors, who were eager to show the proof. Only the scum of the scientific world would deliberately ignore and suppress crucial evidence. Similarly, we have invented the technologies and tools to create and use real components that can build ideal CBPs, and I have openly offered to demonstrate the evidence and proof countless times.

Is it possible for me (who obtained 8 US patents) and dozens of engineers I employed (who created hundreds of real-CBPs) to hallucinate for decades? Only the scum of the scientific world refuse to look at the evidence (by using the right tools and technologies) and insist that no evidence has been provided. One must open one's eyes and look at the evidence. Only the scum of the scientific world close their eyes to the truth and insist that they cannot see the truth: *http://componentology.org/Vid/5****

Forgive me for being harsh and impolite. I have been dealing with the scum of the scientific world since 2006 and I am left with no option but to file a lawsuit to expose the evil racist scum at research organizations such as ACM.org and NSF.gov. If any scientist or reviewer thinks that I am wrong or that I am hallucinating, I as a US citizen am happy to face a defamation case in the US courts, which I am sure will expose the true scum of the scientific world who have been ignoring & sabotaging crucial evidence.

Suppose a researcher seeks to undertake applied research in Layer-2 to tackle a complex technological crisis, such as the

** http://Componentology.org/Vid/6

*** http://Componentology.org/Vid/5

notorious software crisis and the prevalence of spaghetti code. In such a scenario, which would be the preferable theoretical foundation for Layer-1: a valid hard science or the prevailing mythology (which is responsible for the crisis)?

I am ready to go to jail if anyone can prove me wrong in a court of law. I stand by the statement: Any scientist/reviewer is evil scum of the scientific world if they knowingly promote mythology about the mythical things that are disguised as Components/CBE over hard scientific facts and the reality of Componentology about Components/CBE.

The biggest hurdle consists of the unsavory elements within the software research community and members of a deeply entrenched anti-science cult, who shamelessly suppress and sabotage crucial evidence proving beyond any doubt that the 'Existing BoK for CBSE' is junk science.

This anti-science cult shamelessly employs evasive tactics, egregious circular logic, and appeals to authority to justify the perpetuation of junk science. Such appeals to authority often involve quoting baseless opinions in Exhibits B1 to B4 and promoting false, outdated ideas or beliefs, such as those from the 1968 NATO Conference. Ugly manifestations of racism include snubbing, insults, and making condescending and patronizing remarks.

Furthermore, many racists consider it unscholarly and unacademic for a person of color to challenge the beliefs of renowned experts (such as those in Exhibits B1 to B4**). I am intimately familiar with the ugly face of subtle racism and discrimination, having experienced subtle forms of both for nearly two decades while advocating for Componentology, a valid hard science aimed at replacing existing junk science.

Unfortunately, when junk science infiltrates the theoretical foundation, there's no alternative but to replace it with valid real science. Any scientist who rejects such replacement, even after being informed, is the scum of the scientific world. While a single valid and verifiable piece of evidence suffices for honest scientists or reviewers with integrity to expose junk science, racist fake scientists affiliated with the anti-science cult remain impervious to even a hundred such pieces of evidence.

Heliocentrism is not just one of many possible novel ideas (or plausible paths), but an irreplaceable truth that faced hostilities and fierce resistance from the anti-science cult of the day. It was the first step on the only possible right path that opened up a whole new dimension, resulting in the greatest scientific revolution. Since there is no other possible right path for scientific progress, only the scum of the scientific world rejected and refused to investigate the body of data and evidence which was offered openly and repeatedly by Kepler and Galileo.

Similarly, Componentology is not just one of many possible novel ideas (or plausible paths), but an irreplaceable truth that has been facing hostilities and fierce resistance since the year 2008. It was the first step on the only possible right path that would open up a whole new dimension for software engineering. Since there is no other path for addressing the infamous software crisis, only the scum of the scientific world reject and refuse to investigate the available crucial body of evidence, which has been offered openly and repeatedly for nearly 16 years.

Handling fake scientists or dishonest reviewers outside the courtroom presents a formidable challenge because they often resort to deceptive tactics and evasive maneuvers such as excuses, insults, and lies to ignore, evade, or suppress crucial evidence which is vital for uncovering the truth. However, when under oath in a courtroom, they are

** Exhibits B1 to B4 in sections 12 to 15 between pages 22 to 29

barred from employing evasive tactics such as refusing to engage in any kind of deliberations.

Over the past 20 years, I have amassed a substantial volume of data and evidence, empowering me to uncover and expose any deceptive tactics employed by fake scientists. Through my struggles over the years, I am confident that I have encountered every conceivable deceptive tactic that fake scientists might employ when they are compelled to engage in deliberations under oath without suppressing/ignoring crucial evidence.

It would be impossible for NSF.gov to find a single qualified scientist in the world who could successfully disprove, under oath, this vital fact: Componentology is a valid hard science.

Similarly, it would be equally impossible to find even a single qualified scientist who could successfully disprove, under oath, this vital fact: the existing descriptions and concepts (e.g., about Components & CBSE) are junk science.

36. Abstract: 'Software Engineering' is Certainly an Oxymoron

Our patented inventions that create and utilize true "virtual components" — based on "Componentology" — transform software engineering into real engineering.

Our Fundamental Discoveries: Componentology -- A new branch of hard science dedicated to understanding the reality of physical Components, Component-Based Products (CBPs), and the mechanisms of Component-Based Engineering (CBE). A comprehensive understanding of real-world counterparts is the indispensable foundation for creating and utilizing virtual entities, such as virtual components and virtual CBPs.

Our Patented Inventions: We have invented tools and technologies to create and utilize True Virtual Components, based on the indispensable theoretical foundation provided by Componentology. These innovations enable the construction of each software product as a virtual CBP. Our Virtual Components match or exceed the capabilities of the greatest fundamental engineering invention of all time: physical components.

Physical components form the indispensable backbone of every CBE discipline, including mechanical, aerospace, and electronic engineering, where each large product is built as a CBP. Similarly, virtual components are the indispensable backbone vital to transforming software engineering into a truly CBE paradigm, allowing for the creation of each software product as a virtual CBP. We created Componentology since it is impossible to create new technological breakthroughs without having the science to support them.

Summary of Breakthrough Inventions & Groundbreaking Discoveries:

Our transformative technological breakthrough includes multiple patented inventions for the creation and utilization of True Virtual Components for software engineering, all based on a new branch of science called Componentology. Physical or real components serve as the backbone for designing and building large or complex physical products as CBPs (Component-Based Products) across various engineering disciplines, such as mechanical, aerospace, computer, and electronic engineering. Similarly, the virtual components we have developed are poised to become the backbone of software engineering. They are the basic building blocks and will play a vital role in the design and development of each large and complex software product as a true virtual CBP.

Our groundbreaking fundamental discovery is the creation of a new branch of hard science named Componentology, which served as the vital theoretical foundation for our patented inventions. It is impossible to invent True Virtual Components and other necessary tools and technologies for designing and building large and complex software products as true virtual CBPs (Component-Based Products) without gaining a deep and comprehensive understanding of the intricate and nuanced aspects of physical components and CBPs. This includes the nature, essential properties, features, behavior, built-in mechanisms, and functionality of physical components; the anatomy, structure, design, and construction of physical CBPs; and the methods and mechanisms of real CBE (Component-Based Engineering).

Today, in software engineering, there are no true CBPs (Component-Based Products), no true CBE (Component-Based Engineering) paradigm, and no genuine components essential for practicing the CBE paradigm to build each software product as a CBP. Instead, there are a few types of fictitious entities that are misleadingly and deceptively called components, and using these entities to build software products is similarly misrepresented as CBE. The fatal mistake of labeling and using fictitious entities as components has turned 'Software Engineering' into an oxymoron.

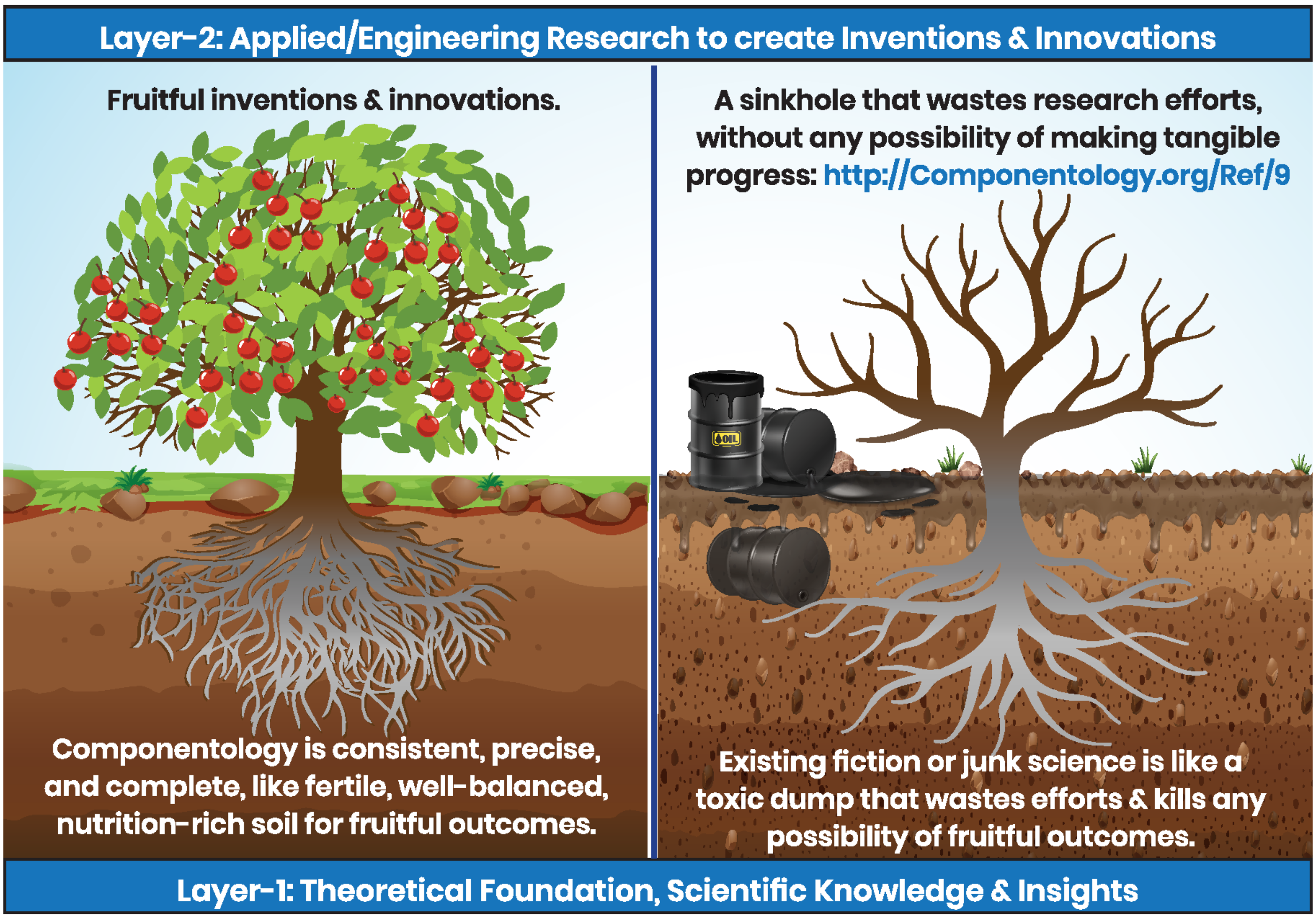
36.1. Is 'Software Engineering' an Oxymoron? by Dr. Alan Key

Dr. Alan Key, winner of the AM Turing Award (considered the Noble Prize in computing) and father of Object-Oriented Programming, compared Software Engineering to the building of pyramids 4500 years ago in his paper: "Is 'Software Engineering' an Oxymoron?" Reference: http://componentology.org/Ref/11. Some of the extracts are given below:

Real Software Engineering is still in the future. There is nothing in current SE that is like the construction of the Empire State Building in less than a year by less than 3000 people: they used powerful ideas and power tools that we don't yet have in software development. If software does "engineering" at all, it is too often at the same level as the ancient Egyptians before the invention of the arch (literally before the making of arches: architecture), who made large structures with hundreds of thousands of slaves toiling for decades to pile stone upon stone: they used weak ideas and weak tools, pretty much like most software development today.

The real question is whether there exists a practice in between the two — stronger than just piling up messes — that can eventually lead us to real modern engineering processes for software. Very often during a project, we'll find out something that we wished we'd known when designing the project. Today, much of the key to more rapid construction and higher success is what we do with this new knowledge. In the "Egyptian" SW systems, we pretty much must start over if we find a basic new way to deal with the foundation of our system. This is so much work that it is generally not done.

Something quite similar obtains in the ominous fact that 85% of the total cost of a SW system is incurred after it has been successfully installed. The bulk of the cost comes from the changes needed from new requirements, late bugs, and so forth. What seemed expedient and cost-saving early on, winds up costing exponentially more when the actual larger life cycle picture is taken into account.


Layer-2: Applied/Engineering Research to create Inventions & Innovations
Fruitful inventions & innovations.
A sinkhole that wastes research efforts, without any possibility of making tangible progress: http://Componentology.org/Ref/9
OIL
Componentology is consistent, precise, and complete, like fertile, well-balanced, nutrition-rich soil for fruitful outcomes.
Existing fiction or junk science is like a toxic dump that wastes efforts & kills any possibility of fruitful outcomes.
Layer-1: Theoretical Foundation, Scientific Knowledge & Insights

A comprehensive and valid theoretical foundation in Layer-1 serves as the fertile ground for technological advancements in Layer-2 (Refer to Table 1 on page 6).

Plants require various nutrients to thrive, including nitrogen, phosphorus, potassium, calcium, magnesium, sulfur, and micronutrients like iron, zinc, copper, and manganese. A well-balanced fertilizer contains these nutrients in appropriate proportions to meet the specific needs of the cultivated plants. Additionally, a "nutrition-rich" fertilizer implies that it not only provides essential nutrients but also contains other beneficial substances that can enhance plant growth and productivity. This may include organic matter, beneficial microbes, or other additives designed to improve soil health and nutrient uptake by plants.

Similarly, to meet the specific needs of applied research conducted in Layer-2 to address a particular technological challenge, it is essential to have a relevant, specific, well-balanced, and valid theoretical foundation of hard/pure science in Layer-1. This foundation must provide a consistent, well-balanced, and comprehensive body of scientific knowledge (as nutrients-rich soil) that includes even minute details of scientific insights (as micronutrients) to conduct fruitful applied or engineering research that can effectively address the given technological challenge.

Any gaps or ambiguities in scientific knowledge and insights in Layer-1 hinder and impede the progress of applied research in Layer-2. This analogy mirrors how a well-balanced and nutrition-rich fertilizer provides a balanced combination of essential nutrients to support the growth and health of plants. Only the valid hard science of Componentology can fill and eliminate every minute gap and ambiguity in order to provide nutrients-rich soil as well as micronutrients.