Axiomatic Cloud Computing Architectural Design

John Thomas[1*], *Pam* Mantri [1]

[1]Cognitive Tools Ltd. LLC, P.O. Box 695; 255 North Ave; New Rochelle, NY 10801, USA

Abstract. Modern cloud computing makes available a plethora of scalable cloud computing offerings. The cloud is increasingly becoming the backbone of the highly complex modern knowledge-economy that includes Social, Mobile, IoT, Big-Data and AI. Knowledge-based products and services follow fat-tail distributions such as the power-law that poses major opportunities and challenges for the designer. The Axiomatic Designer is uniquely positioned in designing for the de-novo situations that the fat-tailed distributions expose. Also, the cloud frees-up the architectural decision-making away from the legacy compatibility-burden, and towards various cloud-native (i.e., de-novo/solution-neutral) as well as hybrid (on-prem/cloud & cloud/cloud) architectures. Furthermore, the competitive landscape around the cloud is not static; it is adaptive and evolving rapidly. Here again, Axiomatic Design (AD) is uniquely positioned in rising up to the various de-novo challenges This, however, requires contributions from frameworks such as Knowledge-as-Heterarchically-Hierarchical (KA|h|H), Stigmergy, Complex Adaptive Systems (CAS), Cynefin, Boyd's OODA-Loop Theory of asymmetric fast-transients, Axiomatic-Maturity-Diagram (AMD), as well as Weick's Loose-Coupling approach to help unify and strengthen the Axiomatic approach. This paper unifies the above approaches in order to tackle the architectural challenges of cloud computing.

Keywords: Cloud Computing; Axiomatic Design; Axiomatic-Maturity-Diagram; Power-Law; Knowledge-as-an-Heterarchic-Hierarchy; Cynefin; Stigmergy; Complex Adaptive System; OODA; Loose-Coupling.

1 Introduction

The Cloud is fundamentally disruptive. If its strategic import is properly understood, it has immense potential to help scale businesses in space (geographical space, product/service space, governance/regulation space, etc.) & time (start-up to IPO, industry business cycle, market seasonality, etc.).

With hindsight, it is understandable that the web would bankrupt a successful legacy business model such as the Borders bookstore that in the mid-90s, mistakenly invested heavily on brick-and-mortar stores across the globe, failed to develop an e-reader, and outsourced its online sales (in 2001) over to Amazon. By the time (i.e., in 2008) Borders realized its online outsourcing error and retracted, it was too late. It filed for bankruptcy on February 16, 2011 [1].

In hindsight, it is easy to see the strategic mistakes that the management at Borders bookstore had made. But from a mathematical perspective, what killed Borders was an inadequate appreciation of the power-law [2] that is operative in the modern knowledge-based, network economies. Human knowledge is a network of concepts which has been aggregating across millenniums. It has a certain shape, structure and overall dynamic. But most significantly, it has a certain directionality in its growth patterns, as dictated by the power-law which results from the phenomenon of preferential-attachment (colloquially known as the Matthew-Effect or the Rich-Get-Richer effect). In other words, what is popular becomes even more popular by virtue of the fact that it is better known. Thus, a brick-and-mortar bookstore that tries to give equitable shelf-space to the mega-successful Harry Potter books as well as the also-rans, simply cannot compete with an online store that only incurs remote-location storage cost that is comparatively cheap. Note that shipping costs are borne by the buyer who pays for it for the convenience of shopping from his/her home. Along with lower inventory costs, an astute vendor such as Amazon also gets to harvest critical insights about the knowledge growth patterns of the buyer [3]. When this is aggregated across the population dimension that spans cities, states, and nations--it provides deep strategic insights about the meristematic growth patterns in the overall economy, and which may be gainfully tapped into. Note that it is not just the purchase of books that provide insights about the struggles and aspirations of a nation that is logging in to browse and purchase. Each item that is being browsed and purchased has aspects of knowledge that went into its design and manufacture. The 5-star rating that captures the *wisdom-of-the-crowd* [4] is the engine that propels the preferential-attachment and the resultant power-law. These strategic insights help the company outsmart its competition.

The power-law distribution of knowledge-based products is fundamentally different from the familiar Gaussian-Normal distribution. Power-law based products exhibit fat-tails that are often underestimated when mistakenly framed using the Gaussian distribution. Fat-Tails force the designer into the uncharted de-novo space. And it is here that the Axiomatic Design approach has a unique role to play. This paper helps trace the potential of Axiomatic Design in the context of de-novo situations that fat-tailed distributions expose in the cloud.

Cloud computing provides various benefits, including agility, scalability, cost reduction, mobility, disaster recovery, etc. Also, the cloud ecosystem is fairly complex. Designing an adaptive architecture that can withstand the onslaught of change is fairly challenging. This paper holistically unifies a smorgasbord of architectural concepts and frameworks that help unify and strengthen the Axiomatic Design approach for tackling cloud computing design. These include the following:

i. History of Cloud Computing

*Corresponding author: johntom@cogtools.com

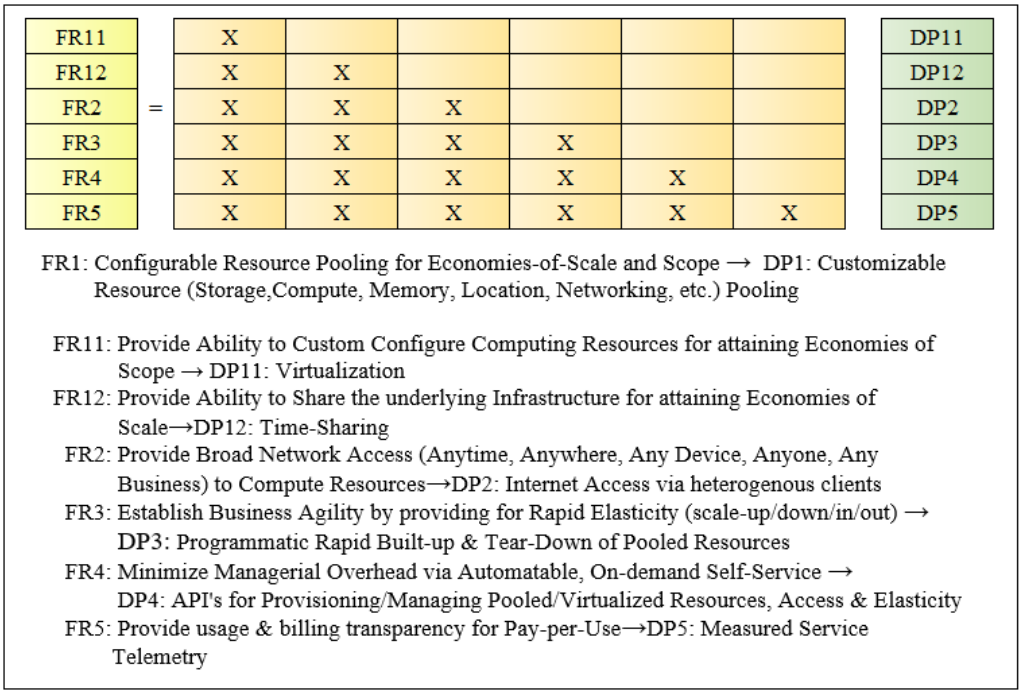


Fig. 1. Basic Cloud Computing Functional Requirement-Design Parameter (FR-DP) Design (based on NIST definition [5])

ii. Socio-Technical Stigmergy
iii. Complex Adaptive Systems (CAS)
iv. Heterarchically Hierarchical (|h|H) Knowledge
v. Power Law vs Gaussian Distribution
vi. Axiomatic Trace along Knowledge Hierarchy
vii. Cynefin
viii. Cloud OODA
ix. Axiomatic Maturity Diagram (AMD)
x. Non-FR's from a CAS Perspective
xi. Security of Cloud Computing
xii. Econo-Strategy in Cloud Design
xiii. Adaptive Architecture

Without the benefit of a holistic design framework, cloud architectures remain fragile. This is especially true in the context of cloud security. Unaddressed gaps in design become salient points of architectural weakness. Cloud offerings are increasingly vulnerable in this context, given the joint ownership model that the cloud operates within. The axiomatic design framework is rare in upholding the holism of design.

2 History of Cloud Computing

Cloud Computing was born in the shadow of the Cuban missile crisis (1962) that almost precipitated a nuclear holocaust [6]. During the crisis, the Pentagon discovered that its existing information infrastructure was practically unusable in a coordinated, large-scale, human-machine endeavor. Simultaneously, early prototypes of time-sharing systems were being developed in the early 1960s at MIT & BBN (Bolt, Beranek, and Newman) under the leadership of Prof. John McCarthy and Joseph C. R. Licklider (Lick). The missile-crisis gave the necessary impetus to launch the time-sharing Project-MAC (Multiple Access Computing) at MIT that spanned 1963-74. By 1969, Lick had expanded and democratized the time-sharing Project-MAC vision to what he colorfully called the "*Intergalactic Computer Network* [7]", which seeded the ARPANET--the historic precursor to the modern internet & gateway to the cloud.

To help stabilize the infrastructure that was constantly being upgraded under Moore's law, the early 1960s also saw market-initiated (primarily IBM, GE and Bell Labs) development of server virtualization [8].

Cloud Computing was born when these two seminal concepts (i.e., time-sharing & server-virtualization) fused in the 1990s. The term itself was coined by Prof. Chellappa in a 1997 talk [9]:

> *"Cloud computing can be defined as a set of frameworks that provides* ***on demand***, ***scalable***, ***customized***, ***quality services*** *in* ***Software***, ***platform*** *and also provides* ***sharable infrastructure*** *through* ***internet*** *that are* ***accessible and available everywhere****"*

In 2011, NIST standardized the official definition of Cloud Computing [5]:

> *Cloud computing is a model for enabling* ***ubiquitous***, ***convenient***, ***on-demand network access*** *to a* ***shared pool*** *of* ***configurable computing resources*** *(e.g., networks, servers, storage, applications, and services) that can be* ***rapidly provisioned and released*** *with* ***minimal management effort*** *or* ***service provider interaction****. ...Cloud systems automatically control and optimize resource use by leveraging a* ***metering capability*** *at some level of abstraction appropriate to the type of*

service (e.g., storage, processing, bandwidth, and active user accounts).

Based on the NIST standard, Cloud Computing may be formally captured in the lower-triangle, FR-DP configuration as shown in Fig.1 above. The design leads off with two seminal concepts that historically define cloud architecture: virtualization and time-sharing. The virtualization design-parameter (DP11) answers the Economies-of-Scope functional-requirement (FR11) for abstracting away from the underlying hardware resources (compute, storage, networking) in order to set up highly customizable resources. Virtualization allows the user to break away from tight coupling with any given hardware vendor (i.e., lock-in). It also allows the architect to mix-and-match technological offerings from various vendors that would have been impossible otherwise. Time-sharing (DP12) allows load-sharing driven Economies-of-Scale of software/hardware resources for substantial cost-reductions (FR12) across multiple users. The service is accessible (FR2) anywhere, anytime, for anyone, for any business, and from any of the client devices that can access the internet (DP2). Given the virtualization functionality across time-shared resources, it is now possible to programmatically provide rapid elasticity (DP3) on an as-needed basis. Such rapid built-up and tear-down of an arbitrary set of resources from the resource pool would have been unthinkable in previous on-prem architectures. From an FR-perspective, what this accomplishes is the agility to rapidly reconfigure and repurpose the strategic business thrust, and move on a dime (FR3). It is this agility that aligns closely with the Boydian strategy framework of asymmetric fast-transients (see section 7). FR4 demands that all of the above FR/DP's need to be managed and achieved on an automated (on the vendor-side) and self-service (on the customer-side) basis with minimal managerial overhead. Again, given the virtualization/time-sharing flexibilities, this requirement is also programmatically feasible (DP4). FR5 requires usage and billing transparency as provided by cloud telemetry (DP5) in order to honor the service contract as well as for providing usage feedback that the customer may use for adapting to changes in the demand curve.

With the above brief historical review of Cloud Computing along with the framing of the basic architecture using the Axiomatic Design, we may now review the rest of the thirteen architectural concepts.

3. Socio-Technical Stigmergy

Stigmergy [10] denotes call to work based on local signs or markings left by collaborating agents (α) at some time in the past and during the course of their work (either as a side-effect of the said work or as something in addition to the work). These markings aggregate to provide organizational directives (β-logic) available at various levels, both within the environment as well as within and between agents. Thus, even though there is no one controlling the set of agents in a top-down sense, there is nevertheless system-wide control being established in a bottom-up sense. Case in point is the ant-trail that emerges (see Fig. 2) from pheromone droppings by ant-agents. The trail then helps organize the ant swarm in its various activities.

The concept of stigmergy was discovered while searching for governing organizational motifs amongst eusocial insects such as ants and termites. Research indicates that these same organizational motifs may be observed in various human activities. Parunak [11] reports on a variety of such human-level socio-technical stigmergic processes, including forest trail-formation, highway traffic-flows, democratic elections, document editing, social-media groupings, viral-marketing, Google page-ranks, peer-to-peer computing, Amazon-style recommender-systems, etc. Stigmergic problem solving occurs wherever the problem context is beyond the ken of any one agent. In other words, one should expect stigmergic solutions to dominate in regimes that would be considered as complex. Architecting solutions within the Cloud Computing/Big-Data space clearly falls within this context. Indeed, the very essence of Big Data is in capturing and tracking stigmergic patterns of economically interesting behaviors across large populations sets. This could be in the realm of finance, health-care, threat-modeling & cyber-security, gaming, education, entertainment, etc. In architecting such systems, it is therefore important to track the stigmergic patterns that are forming across the vast Cloud/Big-Data ecosystem [12-15]. Prior to the cloud, socio-technical stigmergic processes and patterns were either trapped in the confines of the enterprise or merely left as shallow traces across the light-weight internet web-links. The modern cloud has the potential to allow data flow patterns to aggregate without necessarily compromising privacy.

Stigmergic signals from the cloud include DP5: Telemetry based transparency for usage-based billing (see Fig. 1). Organizations are perhaps unaware of the stigmergic significance of such accumulating data. Contractual agreements [16] with vendors need to be carefully negotiated, especially in the context of shape-shifting Complex Adaptive Systems (see section 5)).

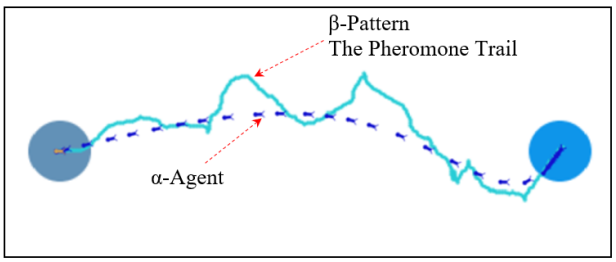


Fig. 2. Stigmergic trails by marching ants (NetLogo [17])

4. Complex Adaptive System (CAS)

To fully understand the strategic import that design plays in the cloud, one has to appreciate the underlying adaptive dynamics propelling the cloud. As Urguhart suggests in [18]:

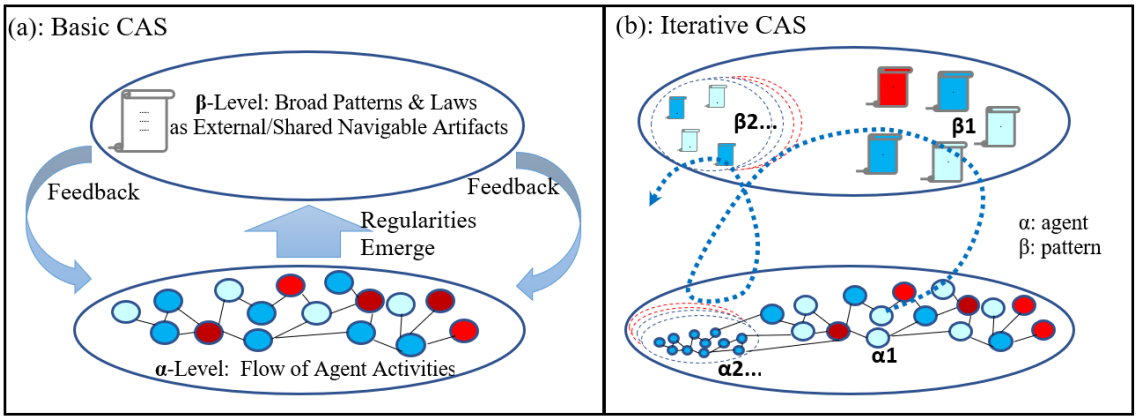


Fig. 3. Complex Adaptive System [21]. (a) Basic; (b) Iterative

Cloud as an adaptive system: The thing is, however, a certain class of complex systems, complex adaptive systems, have the additional trait that they can change their behavior in response to the success or failure of previous behaviors when a given event occurs—or when a certain series of events occurs. This ability to "learn" and adapt to the surrounding system environment creates amazing outcomes, including many of the most rich, enduring and powerful systems in our universe.

Shape-shifting adaptive-dynamics makes the cloud ecosystem a complex-adaptive system (CAS). As Prof. Holland describes it [19], CAS's "*are systems that have a large number of components, often called agents that interact and adapt or learn.*" Holland then proposed a two-tiered system as shown in Fig. 3a above.

The lower α-tier follows a fast-dynamic and is engaged in the flow of resources between diverse agents; while the upper β-tier follows a slow-dynamic that captures knowledge artifacts and aggregates from these which are then emitted system-wide as stigmergic signals that help the agents organize and scale.

Fig. 3(b) is an iterative variation of the basic CAS. At each follow-on feedback-loop/iteration, the CAS trace bifurcates the target population into higher levels of organizational complexity. As evidence of the cloud bifurcation process, the Azure Resource Manager [20] enables the organization and management of the cloud resource sprawl that used to exist in the Classic model.

5. Knowledge Hierarchy/Heterarchy

The following is a short review of the knowledge-hierarchy framework [21] which could aid in the mapping of various frameworks such as Axiomatic Design, Cynefin, OODA, Agile etc., into an integrated whole. Human knowledge is the engine that drives the knowledge economy. Given the abstract nature of human knowledge, it may be observed that domain knowledge is conical in shape; i.e., there are many more concretes than abstractions. The knowledge corpus captures the sum-total of truths/facts gleaned from nature and painstakingly accumulated across time. Induction (depicted in Fig. 4a as the upward flowing arch) involves creating higher-level generalizations. Note the similarity between induction and the upward flowing, regularity creating CAS emergence (Fig. 3a). In contrast, a deduction is the downward flowing arch involved in the application of the induced generalizations. The abductive cascade combines the inductive as well as the deductive flows into a long sequence of step-by-step problem-solving trace.

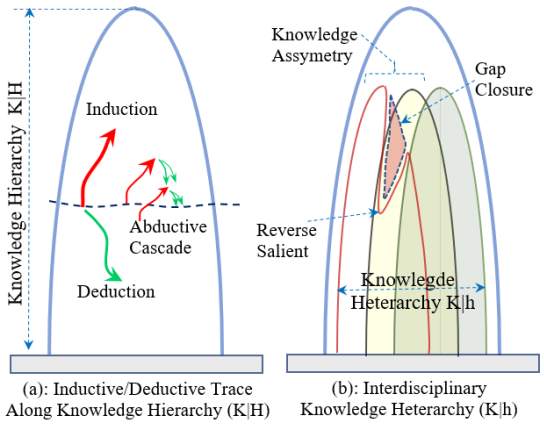


Fig. 4. Knowledge Hierarchy/Heterarchy [21]

When multiple domains are mapped side-by-side along with their shared conceptual linkages, the various hierarchies map onto a heterarchical span that share and cross-pollinate across the domain barriers (Fig. 4b). Hierarchies are denoted as |H, heterarchies as |h, knowledge hierarchy as K|H, and Knowledge as heterarchically-hierarchical as KA|h|H.

Reverse Salients are gaps that appear between knowledge hierarchies when mapped heterarchically, side-by-side. These create knowledge asymmetries between individuals as well as groups of individuals (including corporations and nations). Such units may hold opposing mental models based on the underlying knowledge asymmetries. For example, the management of Borders and Amazon were basing their corporate decisions on mental models that were at variance with each other. By the time gap-closure had occurred, Borders as a corporation was bankrupted.

Systems that display strong heterarchical engagements have very different growth patterns and architectural

opportunities/vulnerabilities. For example, bacterial colonies exchange genetic material not just vertically (in a parent-to-child sense), but also laterally [22] between cells that come into contact with each other. Vertical exchange is hierarchic; horizontal exchange is heterarchic. Evolutionary adaptation that uses the vertical exchange is slow as it has to work across organismal life-spans. In contrast, an adaptation that uses lateral exchange is rapid as it is able to quickly share successful mutations across large populations. But when both mechanisms work in tandem, organisms can rapidly navigate large search-spaces in order to solve species-wide existential threats. For example, this allows bacterial colonies to rapidly evolve and attain anti-biotic resistance [22].

The advent of Social Media, IoT (Internet-of-Things), Big-Data, AI/Machine-Learning, Mobile and Cloud Computing has accelerating such heterarchic linkages. Differences in heterarchic strengths and weaknesses could be complementary or noncomplementary. When pursued for competitive advantage, they result in asymmetric warfare (which is explored more fully in section 9 under Cloud OODA).

Heterarchic problems exist in the context of cloud data and cloud security. For example, the data-modeling complexity involved in a Big-Data/Data-Lake context has to do with ironing out the inevitable ontological inconsistencies across multiple domains, each competing for abstraction dominance as well as in establishing a wider inductive base spanning structured, semi-structured and unstructured data [23]. Likewise, in the context of cloud cybersecurity, the current dominance of the hierarchical defense-in-depth [24] approach is vulnerable to heterarchic attacks. Defense-in-depth assumes that data/applications are secure in the inner-most layer of its hierarchical & concentrically structured onion-rings. But such an approach is vulnerable, especially in the increasingly IoTized technology landscape where data & applications are placed at the edge with both hierarchical as well as heterarchical vulnerabilities. Alongside defense-in-depth, what is therefore also needed is an integrated defense-in-breadth focus [24].

6. Power Law vs Gaussian Distribution

Consider a variable which tracks a phenomenon that has multiple contributing factors, each of which obeys its own unique probability distribution. If these factors aggregate in an additive fashion, the resultant summing-distribution that characterizes the phenomenon would be a bell-curve. As per the Central Limit Theorem, it would result in the Gaussian normal distribution (Fig. 5a) if the contributing distributions were independent and identically distributed. The issue with the normal distribution shows up in the tail regions where the probabilities attenuate drastically. This gives rise to the problem of unreasonably thin tails. Seldom do natural phenomenon follow the Gaussian in the tail regions as it is attenuating too sharply by following the exponential of the negative square of the variable in question (Fig. 5a). Even so, many human-centric and natural phenomenon approximately follow the Gaussian distribution. Examples include temperature distribution in a city at a given day of any year, height/weight distribution of a population, delays in the arrival of public transportation, size and weight of fruits and vegetables, experimental observational errors, etc.

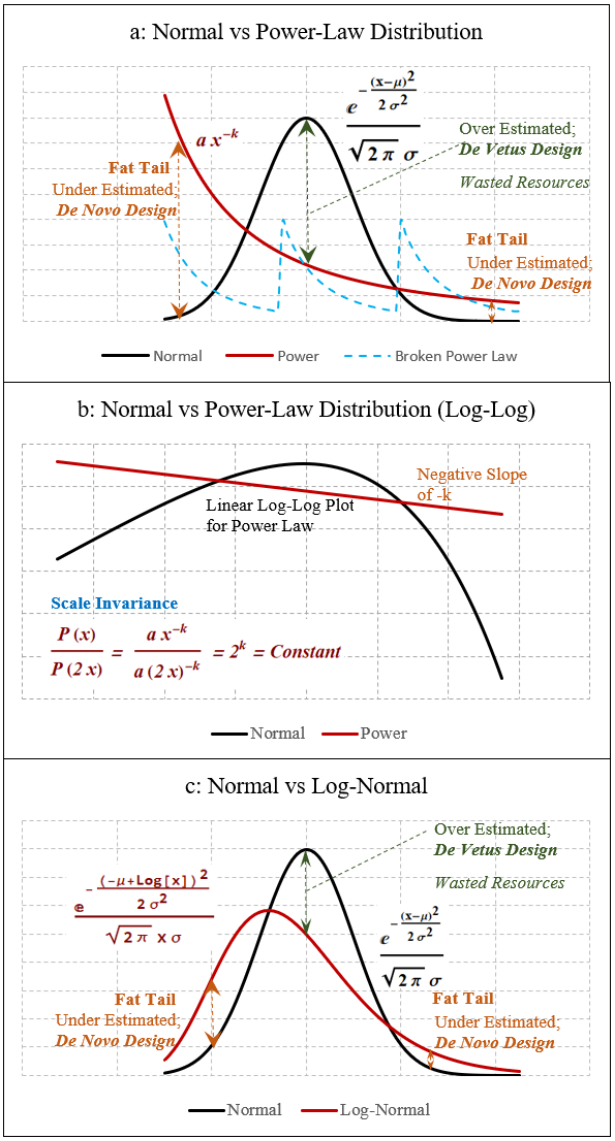


Fig. 5. Normal vs Log-Normal vs Power-Law Distribution

In contrast, if these factors aggregate in a multiplicative fashion, the resultant multiplicative-distribution that characterizes the phenomenon would be a power-law distribution which when plotted on a log-log plot, shows up as a linear plot (see Fig. 5b). Note that from an empirical perspective, there could be many variations of the power law [2], including the broken power-law which consists of piecewise combinations of multiple power-laws) as well as smoothing of the power-law with the exponential.

To illustrate the generative multiplicative process behind the power-law phenomenon, consider the following example of programmer productivity [25-26].

To begin with, suppose that just a few software programmers are proficient in the software-creation tools at their disposal. Even small differences in better tool-usage quickly aggregate to the advantage of the slightly exceptional individual. Each successful project completion builds confidence in the individual as well as in the eyes of the management that oversees the project. In time, with repeat deliveries and accrual of choice experiences, the end result is that these individuals are orders of magnitude more productive than the rest. Seeded by minor differences in the initial conditions (for example, here the slight difference in tool usage proficiency), the generative multiplicative process behind the power law has the potential to bifurcate target populations (in a CAS sense) into distinct groups and sub-groups. In time this could lead to deep-set social hierarchies between those who lead versus those who are led. Indeed, wherever human intellectual work is involved and made available to large populations in a free-market economy, the Rich-Get-Richer style power-law distributions are also likely to show up. This is one of the reasons why the power law is increasingly relevant when considering a knowledge economy.

Other examples [2] of the power-law include earthquake intensities, the population of cities, best-seller copies sold, the number of citations received, etc. In [27], Baraba´si and Albert highlight the generative process behind the power-law distribution of the vertex degree in a network of web-page links:

> *Because of the preferential attachment, a vertex that acquires more connections than another one will increase its connectivity at a higher rate; thus, an initial difference in the connectivity between two vertices will increase further as the network grows.*

From a symmetry perspective, the above argument could also be extended to include preferential *detachment* for nodes that fall out of comparative favor. In time, this creates significant bifurcations between the haves and the have-nots. Given the foundational level wherein the generative multiplicative process behind the power-law is operating from, no egalitarian legislative action could effectively overcome these biases without concomitantly also damaging the knowledge economy. Instead, the proper solution to the above politico-economic problem (in a knowledge economy) is to leverage the neglected heterarchy generation potential of the very same hierarchy generating, power-law driven, iterative-CAS. Over and above the hierarchy generation potential, an iterative CAS module (if and when assisted by favorable factors such as the cloud), also has the ability to generate *heterarchic*-hierarchies which (as explained below) is the fundamental solution to the above egalitarian conundrum. While knowledge hierarchies' fragment and bifurcate a given domain (and therefore the social groups that cultivate it) into smaller and finer grains, knowledge heterarchies create bridging artifacts that allow concepts and propositions to disperse across the overall knowledge fabric. This is similar to the lateral transfer of genes in a bacterial population.

Mindless specialization (into ever-deepening hierarchies) is the bane of modern life. Fortunately, countervailing forces are at play that promises to link the hierarchies via heterarchic bridge artifacts. For example, Axiomatic Design [28,29] is fundamentally a hierarchy-bridging heterarchic artifact that considers all creative designerly activities under a common breadth-seeking rubric. It is breadth-seeking because the very same two principles of design (i.e., Axiom I & II) apply regardless of the domain of interest. In other words, regardless of whether the design pertains to engineering, software, education, organizations, medicine, etc., the same two design principles apply [28].

By balancing the depth-seeking hierarchic drivers against the breadth-seeking heterarchic drivers, iterative-CAS has the potential to coordinate and disperse knowledge-agents across the totality of the knowledge economy, instead of crowding around just a few clusters. Thus, instead of egalitarially spreading the wealth (i.e., the product of human creativity) around, the heterarchy generating iterative-CAS is capable of spreading the wealth-generating engine of human creativity around. In colloquial terms, it is about "*teaching how to fish instead of giving fish [30]*."

Iterative-CAS has the potential to flatten deep hierarchies in favor of a network economy that bridges isolated hierarchies using a multitude of lateral linkages. Furthermore, these heterarchic linkages allow the search-space along all stages of the design process to be vastly improved. We may, therefore, be more confident that the search imperative embedded in Axiom II (i.e., minimize information content) is indeed delivering a true minimum. This is similar to how bacterial populations are able to rapidly navigate large search-spaces in order to solve species-wide existential threats by utilizing both horizontal as well as vertical gene transfer mechanisms.

In a knowledge-driven political-economy, the proper nurturing of the link between human-creativity embedded in the act of design and finding meaning and fulfillment in life is of paramount importance. Note that in such an economy, the neglected art and science of design ought to be center stage. Also, in such an economy, the flattening of deep-set knowledge hierarchies in favor of heterarchic-hierarchies allows for greater freedoms in human actualization. As an analogy, this is similar to the phenomenon of seed dispersal far removed from the parent, which allowed plants to colonize and spread itself across the globe. Dispersal of seeds opened up ecological pathways for the evolution of plants into rich, diverse forms that could successfully exploit the local micro-ecologies. However, the difference between seed dispersal and knowledge growth patterns is that while the available global surface area with sufficient sunlight for plant-growth is limited, such is not the case for the highly fractalized heterarchically-hierarchical knowledge architectures [21]. Indeed, such an abstract space is only limited by our imagination. In other words, there is sufficient "*surface area"* across the richly fractalized knowledge fabric for humans to flourish without having to

dominate and extinguish the creative entelechy in each other in the narrow hierarchies of disjointed knowledge fragments. Such ought to be the proper solution to the Rich-Get-Richer power law conundrum. And in this context, cloud computing is like the wind that is dispersing the seeds of human imagination into ever wider mash-ups of creative, uncharted territories.

While power-laws go hand-in-hand with CAS-generated hierarchies, they do not necessarily vanish with CAS-generated heterarchic-hierarchies. Both depth-driving hierarchies as well as breadth-seeking heterarchies are necessary for balanced growth, but need to exist in symbiosis. For example, the ancient redwood trees (symbolizing deep hierarchies) did not go extinct when grasslands (symbolizing wide heterarchies) started appearing 55 MYA. Instead, each continued to thrive separately within their respective niches while collaborating on the wider photo-synthetic gas exchange cycles. Also, within its respective niche, each exhibits various power-laws in its relevant allometric measurements [31,32]. In other words, it is not a single power-law across the overall span. Instead, it is broken into separate segments, each governing a separate niche. This is what was referred to earlier as the broken power law (Fig. 5a).

In cloud computing, Loboz finds significant backing for the power-law when he analyzed Microsoft's Azure Resource usage patterns [33]:

> *Analysis of daily resource usage by customer accounts on two Azure storage clusters had shown that the distribution of the resource usage on any given day is very heavy-tailed. We have found, for five different resource types that distributions are far from normal, exponential, or even log-normal – in fact they either are power-law or closer to power-law than any of the aforementioned distributions*.

When jointly plotted against each other, the power-law distribution highlights key regions of over/ underestimation of probabilities under the guidance of the bell-curve logic. As shown in Fig. 5a, the tails at the extremes are often underestimated, while the middle is overestimated. Underestimated tail regions are denoted as fat-tails. For example, suppose that the T-shirt industry was to be designed under the guidance of the normal distribution but in fact exhibited power-law distribution. If that were to be the case, it would be as if ready-made clothes were being designed mostly for the middle region of the normal curve (i.e., small, medium & large); but sizable populations continue to show up at the retail stores who are extremely large or extremely small. This, of course, does not happen in human body proportions which generally follows the Gaussian normal curve. But if that were to be the case, huge sections of the market that pertained to the massive fat tail at the lower end (i.e., extra-small) of the market would effectively be invisible to the designers. Likewise, some extremely large-bodied individuals would also not be serviced and therefore forced to obtain custom-tailored clothing. Also, in the middle regions (i.e, small, medium, large), there would be a huge surplus and wastage. Clearly, misjudging the market would be a major problem for the designer.

Note that fat tails may also form under non-power-law situations such as shown in Fig. 5c which overlays the normal against the log-normal. Like the power-law, log-normal is also multiplicative, but places restrictions such as stock-prices never being allowed to fall below zero. Examples of log-normal distributions include:

- File Size Distribution on the internet.
- The internet traffic rate
- How long users stay and peruse an online article

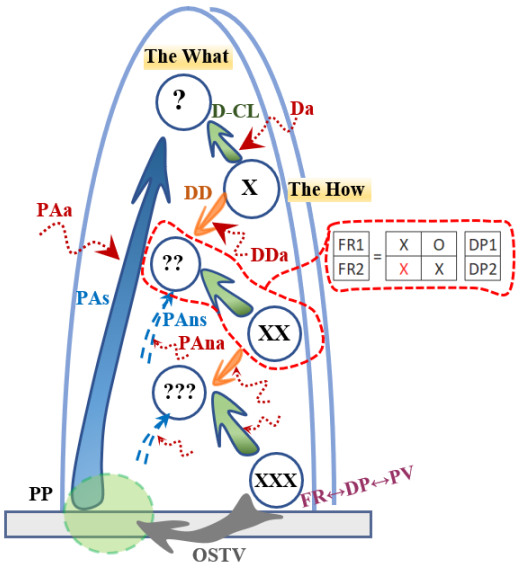
As shown in Fig. 5b, the defining property of the power-law distribution is that it is scale-free. In other words, in transitioning from x to 2x (or any multiple), the ratio of the distribution is constant (i.e., invariant) no matter what that x is. In other words, it exhibits the fractal property of self-similarity (i.e., the whole has the same shape as the part). Under purely hierarchic dominance, the power-law structure that results would be the trivial and featureless straight-line (Fig. 5b); but when mixed with heterarchic influences, the resultant broken power-law structure is far more interesting and realistic.

From an Axiomatic Design perspective, what is critically relevant about the power-law distribution is that the fat-tails are often underestimated and fall outside the mainstream. This means that the design requirements need to be freshly induced; and the design itself needs to be attempted in a solution-neutral, de-novo fashion. This is also probably the reason why the agile movement has resonated so well in our modern knowledge economy. If the requirements need to be freshly induced in order to cater for the fat-tails, any top-down water-fall type approach that caters to the middle regions of the bell-curve logic would incur major costs of wasted effort and misdirected resources. Power-law distribution is central to the modern knowledge economy. And fundamentally, given its structuring ability when dealing with solution-neutral/de-novo problem contexts, Axiomatic Design is uniquely situated in rising up to the modern challenge of designerly misguidance under the normal curve.

Note that the two fat tails as shown in Fig. 5a are neither symmetric nor perhaps of equal significance. Given the relative probabilities between the two, occurrences of the fat tail at the short end at left are far more likely than that of the long-end at right. In other words, there is far more statistics available for the short end as compared to the long end (for example, far more small earthquakes versus just a few really large ones). But from an impact perspective, events in the long-end are probably far more consequential than that at the short-end (for example, the energy released as well as damage from a very large earthquake). Also, tracing the causal linkages at the short end is far more hierarchical and Markovian [34] (i.e., lacking in memory) than that at the long end. In contrast, the tracing of causal linkages in phenomenon that exhibit memory requires far more heterarchical thinking (which currently is more demanding and therefore in short supply).

7. Axiomatic Trace

Given the knowledge hierarchy framework, the Axiomatic Design (AD) trace may be depicted as in Fig. 6 below. Given that human knowledge is hierarchical, the design-trace that leverages this knowledge is likewise hierarchical. As explained in section 6, fat-tails expose designerly blind-spots that need to be problem-abstracted afresh and designed in a solution-neutral, de-novo fashion. De-novo designs also occur in uncharted contexts (such as cloud-native architectures) that requires the designer to freshly induce the highest-level problem-context that will govern the overall design. In contrast, on-prem designs that compete with the cloud have legacy commitments that need to be carefully re-engineered in the context of the cloud as they have substantial amounts of the design-trace locked-up in long-shelf-life, cap-ex obligations. In all of these de-novo, cloud-native/hybrid contexts, the axiomatic approach could provide critical insights as to how best to proceed.



Abbreviations

PP: Problem Perception
PAs: Problem Abstraction-Synthesis
PAa: Problem Abstraction-Analysis
DCL: Design-Creative Leap
?: The What
X: The How
Da: Design Analysis
DD: Design Decomposition
DDa: Design Decomposition Analysis
PAns: Problem Abstraction (nested)-Synthesis
PAna: Problem Abstraction (nested) Analysis
FR: Functional Requirement
DP: Design Parameter
PV: Process Variable
OSTV: Overall System Testing & Validation

Fig. 6. Axiomatic Design Trace Along K|H [35]

As Prof. Suh indicates in [28], design "*involves four distinct aspects of engineering and scientific endeavor*" as listed below:

- Problem Definition
- Creative Leap
- Analytical Process
- Overall Testing & Validation

Problem Definition involves Problem Perception (PP), Problem Abstraction-Synthetic (PAs) as well as Problem Abstraction-Analytic (PAa). PP is how we perceive the problem (e.g., patient perceiving pain in the chest and showing up at the doctors). PAs is the diagnosis of the problem at the right level of abstraction in an essentialized sense (e.g., the medical doctor having done sufficient tests on the patient, decides that the patient has a life-threatening blockage in the coronary artery). When the patient takes the diagnosis and asks for a 2nd opinion about the diagnosis, that would be PAa: Problem Abstraction-Analytic. If the presiding doctors were Sigwart and Puel [36], the creative design leap (DCL) could be the world's very first heart stent. Prior to the advent of the coronary stent, the medical profession has had close to two centuries of experience in stenting of vessels in other minor organs. All of this knowledge would be relevant as prior art and therefore inform the de novo design of the coronary heart stent. Ideally, it should be explicitly captured as part of the growing K|H. Along with the axiomatic tools and corollaries from AD, the prior art (whether implicit or explicitly captured in the K|H) helps in the proper analysis of the design (i.e., in the Da: Design Analysis step). Despite the prior art, every de-novo design probably has unique elements (that falls outside the current prior-art) which requires rigorous testing.

Commenting on the hierarchical nature of design, Prof. Suh indicates in [28] that:

- *Everything we do in design has a hierarchical nature to it. That is, decisions must be made in order of importance by decomposing the problem into a hierarchy... When such a hierarchical nature of decision making is not utilized, the process of decision making becomes very complex.*
- *The designer must recognize and take advantage of the existence of the functional and physical hierarchies. A good designer can identify the most important FRs at each level of the functional tree by eliminating secondary factors from consideration. Less-able designers often try to consider all the FRs of every level simultaneously, rather than making use of the hierarchical nature of FRs and DP's.*

The above sentiment is the strongest indication that AD is closely aligned with K|H. The only distinction is that hierarchies are not just relevant in the top-down decompositional phase; it is also of equal (if not more) relevance in the original problem abstraction phase too (i.e., PAs & PAa). The familiar set of design matrices (as shown in the red-dot outlined offset in Fig. 6) also captures the hierarchical trace.

The step-by-step decomposition of the abstract design is aided by four auxiliary design processes:

- DD: Design Decomposition
- DDa: Design Decomposition Analysis
- PAns: Problem Abstraction (nested)-Synthesis
- PAna: Problem Abstraction (nested)-Analysis

These steps are templated along the abductive cascade as shown in Fig. 4a.

The 4th major step in the design process, namely OSTV: Overall System Testing & Validation makes sure that the original problem (in our above example, the pain in the chest) has been adequately addressed.

Tracing designs across the knowledge repository could be of value in at least six different ways:

- By tracing the design across a well-explicated heterarchic knowledge hierarchy, the design is also well documented. Documentation is apparently burdensome in modern agile practices on account of the effort involved. The design-trace approach could help overcome this burden by leveraging and reusing that which is common.
- When the design-trace is used for capturing evolving families of designs that are related by a common problem context, it creates a phylogeny which could be mined for stigmergic patterns which otherwise would be missed.
- The design-trace would add a valuable pedagogic tool for teaching design.
- In the hierarchical composition/decomposition of the design, the trace could help assure that the conceptual order is being maintained. In other words, it would be out of place to witness higher-level abstractions showing up at lower-level designs. And vice-versa, it would also violate the knowledge trace if lower-level abstractions show up in the higher rungs.
- As per the *ironic process theory* [37], attempts to suppress certain thoughts, unfortunately, makes it all the more likely to happen. Colloquially this is called the "*don't think about the white bear*" problem which results in the subject trapped in the very same thought process that is taboo. Likewise, the requirement to think out-of-the-box in a solution-neutral, de-novo sense is much harder when a solution already exists. Such cognitive traps may be avoided if the mind could free-range and view the overall conceptual landscape with the current de-vetus design being included rather than excluded.
- Given the nature of the intense specialization in modern knowledge-economies (i.e., the problem of the dearth of generalists), problems and solutions are posed within the limited domain expertise of the designer. By tracing the design across the heterarchic knowledge hierarchy, such self-limiting parochialisms may be avoided.

These are some of the myriad ways that the tracing of design across the heterarchic knowledge hierarchy could benefit Axiomatic Design.

8. Cynefin

While earlier technologies had recognizable life-cycle trajectories that could be analyzed along simple, clear, well-structured, top-down frameworks such as SWOT matrices [38], Porters five competitive forces [39], etc., the strategic and business-oriented framing of cloud computing has been addressed in mostly a piecemeal fashion (as for example, frameworks for cloud security [40], governance [41], migration [42], vendor selection [43], etc.). Snowden's Cynefin framework [44] is an integrated, inductive, bottom-up sensemaking framework that is complexity-aware and therefore of substantial relevance in the cloud context. It has, however, yet to be adapted for the cloud computing context [45]. Along with casting the Cynefin approach in the knowledge hierarchy framework, the following discussion highlights the cloud computing potential for Cynefin.

The Cynefin Framework (Fig. 7) highlights both the opportunities as well as the challenges faced by architects embracing the Complexity Challenge [44]:

> *In a complex context, however, right answers can't be ferreted out. It's like the difference between, say, a Ferrari and the Brazilian rainforest… Ferraris are complicated machines, but an expert mechanic can take one apart and reassemble it without changing a thing. The car is static, and the whole is the sum of its parts. The rainforest, on the other hand, is in constant flux—a species becomes extinct, weather patterns change, an agricultural project reroutes a water source—and the whole is far more than the sum of its parts. This is the realm of "unknown unknowns," and it is the domain to which much of contemporary business has shifted.*

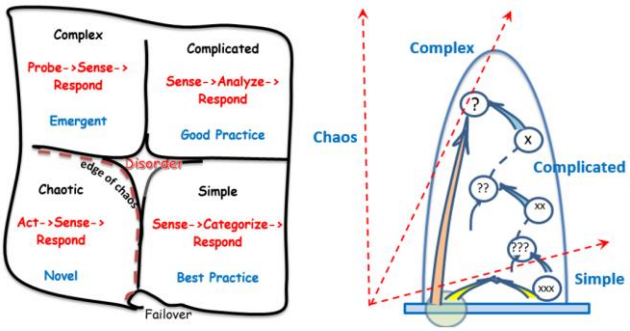


Fig. 7. Cynefin & Knowledge Hierarchy [46-47]

Cynefin is unique in emphasizing distinct and discernable managerial realms and heuristics (Fig. 7) where either reductionism (i.e., the whole is the sum of its parts) or holism (i.e., the whole is more than the sum of its parts) is the dominant operative. Holism and reductionism have ancient Greek heritage. As shown in Fig. 8 below, cloud architects may present three distinct temperaments when considering holistic vs. reductionistic tendencies. The ideal cloud architect (i.e., the architect-engineer) is both a generalist (in a big-picture sense) as well as a specialist (from a fast-moving technology perspective).

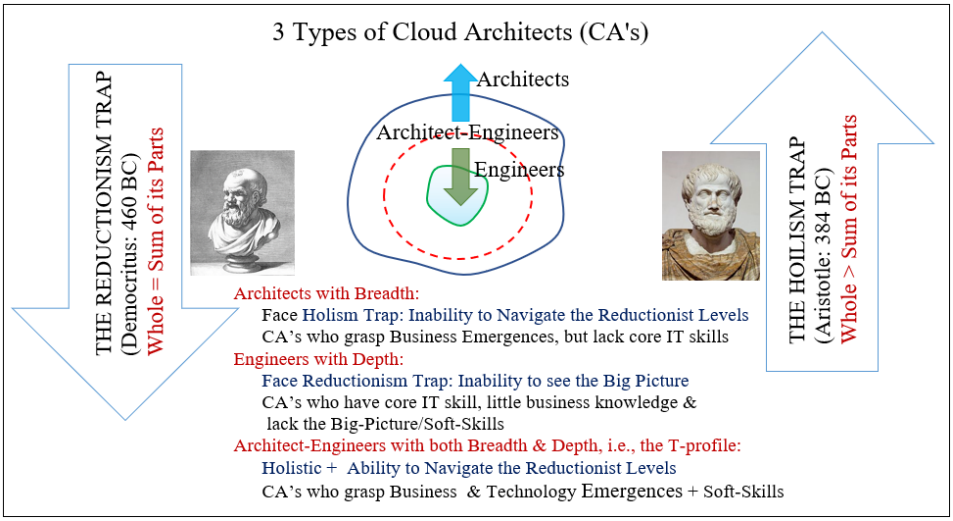


Fig. 8. Holism, Reductionism and Cloud Architects (Images from Wikipedia [48] & [49])

Such a skill-sets profile could be characterized as the T-profile (i.e., broad as the head of the T, as well as deep as the leg the T). The ideal cloud architect ought to be able to anticipate and envision technological shifts that might trigger business emergences, as well as business shifts that might trigger technological emergences (in a CAS-sense). Furthermore, the ideal architect ought to have the necessary people-skills/soft-skills to communicate, persuade, motivate and navigate across complex corporate terrains in order to help bring about the requisite corporate realignments that the envisioned emergences entail. As mentioned in section 3, given the fact that the Axiomatic Design Framework is capable of both hierarchical as well as hierarchy-bridging heterarchic designs, it is uniquely positioned in training the aforementioned architect-engineer. Also, if the design-trace is mapped against the knowledge hierarchy phylogeny, it could help anticipate potential emergences. Furthermore, given the fact that it is able to smoothly range across the abstraction spectrum, it is also able to assimilate both managerial as well as technical expertise into an integrated approach. Even the development of soft-skills as a well-organized collection of building-block skill-sets could be reduced to a problem of design [50]. Nevertheless, Axiomatic Design is not a panacea; it requires collaborative contributions from auxiliary frameworks (such as Cynefin) to help it in solving modern complex problems (such as cloud computing).

The vertical axis in Cynefin divides the ordered (on the right) versus the unordered (on the left). Of vital strategic essence is the proper locating of the problem in the proper regime. Here, the onus is to lead with bottom-up data in finding the right regime rather than applying any given framework in a top-down sense (i.e., let bottom-up induction have dominance).

By casting the Cynefin framework alongside the knowledge hierarchy framework (Fig. 7), it becomes clear that both the simple as well as the complicated are operating along well-structured knowledge hierarchies. In contrast, the necessary inductive base has yet to be established in both the complex as well as the chaotic regimes. The difference between the complex and the chaotic is that the former has at least partial conceptual order that overlaps with the conceptually know world. Furthermore, in the chaotic realm, higher values are under imminent threat (as in a medical Emergency Room situation) and require quick heuristic-based thinking and safety-enhancing actions. In the cloud context, chaos is when there is a major data breach with the host organization facing an existential crisis.

The knowledge-hierarchy framework shows the continuum between the various regimes; i.e., reductionism and holism are not set against each other. Instead, each requires the other in order for knowledge to progress. Given that many of modern business problems manifest first in the complex realm, one should, therefore, expect that after analysis, some parts of the problem would be treated in a reductionist sense, while others in a holistic sense. Cynefin, however, warns about the danger of treating a complex problem as if it were simple or complicated. The warning is that this could lead to failover of the project into disorder (shown centrally as well as with a bottom swoosh in Fig. 7) where the managerial governance is itself lost. Problem-solving in the Cynefin world is to be contained within the separate regimes. Such a heightened sense of alarm would be short-sighted. As indicated above, the right approach would have been to partition and transition the reducible parts of the problem over to the simple/complicated regimes while dealing with the non-reducible parts in its own rights.

Consider for example some of the professional practices in software engineering. The top-down, deduction-biased, process-heavy, waterfall framework was perhaps adequately suited for an earlier era of simpler software development; it, however, fails in any of the other regimes where induction dominates, and the

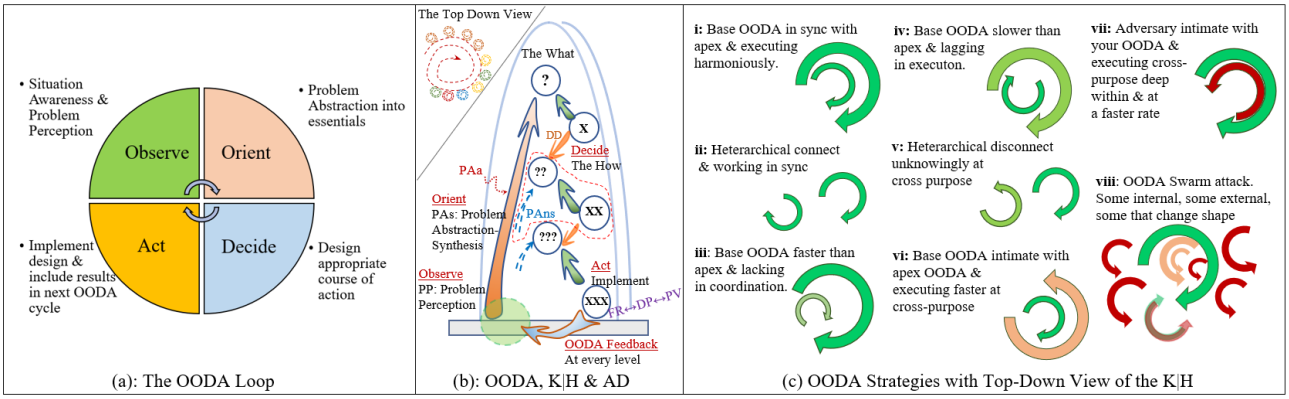


Fig. 9. (a) The OODA Loop (b) OODA, K|H and AD (c) OODA Strategies

rigidity of the process becomes a bureaucratic ball-and-chain against agility and innovation. In contrast, the agile framework is better suited for the complicated regime. Also (with adequate care), it could iteratively move the complex into a more manageable "complicated" regime. This is similar to the transformation of Time-Dependent Complexity in the Axiomatic Design/Complexity Theory (AD/CT) Framework [51] with the caveat that order and disorder is not merely temporal—it could also be geometric, chemical, informational, biological, etc.

The last line in the aforementioned quote (i.e., *...this is the realm of "unknown unknowns," and it is the domain to which much of contemporary business has shifted*) is relevant in the cloud context. The failure of design here is that while the majority of FR-DPs in a given problem context is situated in the well-established industry practices of the simple/complicated realms, a few top-level components of the problem are frequently situated in the complex realm. Failure also happens when engineering and management locate the totality of the problem in one or the other realms and therefore miss the combinatorial. These same blind-spots probably existed even prior to the advent of cloud computing. But what has changed with the cloud is the rapidity with which novel, top-level problems with the potential for major impact shows up. The fundamental problem of cloud computing, therefore, involves expeditiously coming up with de-novo designs for a few of the top-level, rapidly evolving FR-DP problem components even while much of the adjacent/lower-level FR-DP components remain static and therefore re-targetable (with minimal change) from the existing legacy/de-vetus play. The challenge, therefore, is in facing the rapidity with which the top layers need to be continuously shaped, reshaped and reformulated. It is in these de-novo, de-vetus and mixed cases that the sense-making framework of Cynefin, along with the structuring that the Axiomatic Design framework provides, helps. In the presence of |h/|H, a judicious mixture of the various components (i.e., complex, complicated & simple) would be more realistic.

9. Cloud OODA

We now turn to Boyd's OODA framework [52] which is well placed in coming to terms with two of the most fundamental concepts in modern strategy, namely asymmetric warfare and fast-transients. Asymmetric warfare is related to heterarchies, while fast-transients is related to hierarchies. Military Strategist/Systems Architect (of the LWF: Lightweight Fighter program, which gave rise to the legendary F16 fighter plane) Colonel John Boyd highlighted the importance of fast-transients in his OODA (Observe-Orient-Decide-Act) loop framework (see Fig. 9 as well as [53]):

> *Idea of fast transients suggests that, in order to win, we should operate at a faster tempo or rhythm than our adversaries—or, better yet, get inside our adversary's Observation-Orientation-Decision-Action time cycle or loop.*

The four-stage OODA loop is as depicted in Fig. 9a. In Fig. 9b, the four stages of the OODA loop (as well as that for OODA feedback) are traced across the knowledge hierarchy along with the trace for Axiomatic Design (i.e., key elements from Fig. 6) we had discussed in section 7.

In [54], Colonel Richards reviews the strategic value of the OODA Framework:

> *What Boyd discovered was that the side with the quicker OODA loops began to exert a strange and terrifying effect on its opponent. Quicker OODA execution caused the slower side to begin falling farther and farther behind events, to begin to lose touch with the situation. Acting like the "asymmetric fast transients" experienced by fighter pilots, these mismatches with reality caused the more agile side to start becoming ambiguous in the mind of the less agile.*

The key phrase, "*asymmetric fast transient*" needs to be carefully dissected and elaborated upon.

In both OODA as well as Axiomatic Design, the critical step is that of problem abstraction, which in OODA is denoted as the orientation stage. As Boyd writes [53]:

> *The second O, orientation—as the repository of our genetic heritage, cultural tradition, and previous experiences—is the most important part of the O-O-D-A loop since it shapes the way we observe, the way we decide, the way we act.*

Likewise, as Prof. Suh indicates in [28]:

> *It may be useful to state once more the importance of proper problem definition: the perceived needs must be reduced to an imaginative set of FRs as the first and most critical stage of the design process.*

Both approaches highlight the seminal value of problem abstraction. When placed in the K|H context, it becomes clear why the problem abstraction phase (i.e., orientation) has such strategic import. In comparison to every other step, it has the longest arc (i.e., PAs/PAa in Figs. 6 and 9b) along the K|H. Furthermore, being upward oriented, it is fundamentally inductive, which makes it more error-prone. The manner in which Boyd came up with the E-M (Energy-Maneuverability) theory [52] that informs fighter-aircraft design illustrates the inductive challenge. The E-M theory resulted from synthesizing various contributing insights, including firsthand practical experience battling Mig-15 fighters in the Korean War, studies in strategic warfare, thermodynamics/ aerodynamics as well as voluminous computer simulations designed to help create flight performance envelopes. From a design theory perspective, what was being induced was not the FR specifications for any specific fighter-plane; instead it was establishing the overall theoretical design envelope for all possible fighter planes. With the benefit of this framework, a fighter aircraft designer could reasonably articulate a feasible set of FR's based on the perceived needs of the customer (i.e., the defense department).

But the E-M Theory helps structure only part of the OODA loop (i.e., PAa/PAs). Furthermore, it could not explain the 10:1 kill-ratio between the F-86 (predecessor to F-16) and the technically superior Mig-15 (higher ceiling, tighter turn radius, higher maximum speed). In fact, the Mig-15 was better positioned on the E-M profile. As Colonel Richards observed in [54], the controversy was that the "*MiG's theoretically higher EM performance rarely led to wins in actual or even in practice air-to-air combat.*" Puzzling over this ambiguity, Boyd noticed the following countervailing facts in favor of the F-86 [54]:

- The F-86's bubble canopy provided a simple, direct 360-degree field of vision that helped the pilot become better situationally aware (in visually detecting the enemy aircraft) as compared to the constraining view (i.e., the rear-view was blocked) from the Mig-15 canopy. Engaging in a fast-transient dog-fight requires better tools for situation-awareness and problem-perception (PP in Figs. 6 & 9b). This pertains to the observe-phase of the OODA loop. Likewise, in the context of the cloud (with divided responsibility between the client and the cloud-vendor), it is worthwhile creating automated monitoring algorithms based on the telemetric signals (i.e., DP5 in Fig. 1) along with simple dashboards to help the DevOps team be continuously situation-aware. Clutter, complexity-worship, and confusion need to be removed in favor of simplicity. This is in contrast to the Cynefin framework which cautions against moving from the complex realm into the simple for fear of falling into disorder. In other words, while Cynefin is valuable in becoming situationally aware as to where one begins with, it may not be the right choice in transitioning the problem (either whole or in part) from the complex into the more manageable complicated and simple regimes. Note that in pursuing the simple versus the complex/complicated, OODA agrees with AD's Axiom II which recommends the minimization of information content.
- The F-86 had fully hydraulic controls which allowed the pilot to command the aircraft with a single finger. In contrast, the Mig-15 pilot had to strenuously exert physical labor in controlling the aircraft that relied on mechanical linkages. The ability of the Mig-15 pilot to act in a consistent, coordinated fashion degraded under physical exhaustion. Note that this pertains to the act-phase of the OODA loop. Poor actions in one loop feed every phase in the follow-on OODA loop, thereby creating vicious cycles. Here again, we see the importance of simplicity, but now in the realm of human action. The tactical end result of the hydraulic controls was that in comparison to the Mig-15 pilot, the F86 pilot could engage in faster transitions from one OODA maneuver to another.

Along with comparable EM credentials, the addition of a bubble canopy as well as hydraulic controls made the F-86 fighter plane strategically superior to the Mig-15 in an "*asymmetric fast-transient sense*". The asymmetry here is from the OODA-related strategic value of the bubble canopy as well as the hydraulic controls. And it is this asymmetry that is feeding the step-by-step degradation of the adversary, the process whereby the slower side begins *"falling farther and farther behind events"* and *"to lose touch with the situation [54]."*

It is important to realize that cloud computing is strategic and needs to be considered in warfare terms. Critical assets deployed on the cloud are no longer business-as-usual; it, in fact, is being positioned on a winner-take-all basis. In this context (and as was illustrated in the fighter-plane example), every aspect of the OODA loop needs to be examined for its asymmetry potential. These insights, when abstracted and generalized using the KA|h|H framework along with AD, has the potential to scale.

It is true that as the top-level design is decomposed, there are many more nested abstractions (i.e., the steps denoted as PAns in Figs. 6 and 9b). Each of these is a baby-step (compared to PAs) and closely triangulated with the aid of the downward arching deductive decompositions, (i.e., the steps denoted as DD in Figs. 6 and 9b). These nested PAns's are the orientation phases of the nested OODA loops which when viewed from the top, creates a fractal pattern of spirals (see top-left in Fig. 9b).

Consider once again the problem of problem abstraction (i.e., PAa/PAs). In a negative sense, if the problem is poorly stated; or worse if the wrong problem is being addressed, all downstream effort is wasted. In the medical context, this is called misdiagnosis. To put a human face on the cost of misdiagnosis, consider the following healthcare summary from a John Hopkins [55] report:

- Significant cases of permanent injury or death from misdiagnosis estimated to be 80K-160K/year.
- Diagnostic-error related medical claims dominate in the total count (28.6%) and amount (35.2%).
- For new diagnosis, error may range up to 15%.

As we saw in section 3 (where we discussed the problem of fat-tails in the context of the power-law), if we perceive the problem to be normally distributed when in fact it is operating under the power-law or log-normal distributions, it would force the designers to effectively "*bark up the wrong tree*." Referring back to the Borders bookstore case discussed in section 1, the management failed to perceive the threat of the internet as well as the demise of the brick-and-mortar, resulting in the bankruptcy of the firm in a short 10-year period.

In contrast, if designers could rapidly diagnose and orient (i.e., problem abstract in essentials) around real and pressing issues, it would make all the difference in orienting the design and execution teams in the right direction and solving the real problem.

Note that there are subtleties and nuances in the concept of asymmetric fast-transients associated with the OODA-loop. For example, in the above illustration of how hydraulic controls contributed to fast transitions from one OODA maneuver to another, speed is of the essence. But note that such an OODA-loop is situated at the tactical level. In cloud computing, this would be similar to the roles and responsibilities of the DevOps team. In Fig. 9b, the tactical OODA has been demarcated within a red-dotted boundary in the middle. In contrast, in the case of the overall OODA loop which included the E-M theory, the issue is more strategic as it is not so much about physical speed; it is about swiftly tapping into accurate mental models [52]. In other words, the issue is not so much about how quickly the fighter pilot can tactically transition from one maneuver to another in order to get around and get behind the enemy aircraft for establishing air dominance in three-dimensional physical space; the issue is more about how quickly and efficiently the aircraft system architect could design, test and transition from one configuration to another in the abstract conceptual space, as and when the requirements change.

Given the broad centrality and reach of cloud computing, the architectural design of the cloud is fundamentally strategic in nature. But given the speed at which cloud systems may be assembled and torn-down, everyone is operating under tactical time-pressures. Change is relentless in the cloud, and time is of the essence. In other words, architects do not have the luxury of time in formulating their architectural designs. They have to deliver strategic designs under tactical time constraints.

In Cloudonomics [54], Weinman highlighted a similar theme in his 7/10 laws of Cloudonomics:

> *A real-time enterprise derives competitive advantage from responding to changing business conditions and opportunities faster than the competition.*

It is in the fast-transient challenge that the cloud poses an "*existential threat*" to the legacy (e.g., banking) as well as an "*irresistible opportunity*" to the upstarts (e.g., FinTech).

As was the case with Boyd (who had hands-on experience flying sorties in the Korean war), the cloud architect needs to have hands-on experience in all facets of the cloud. And as indicated in Fig. 8, the cloud architect also needs to be able to span the strategic business/technological context in the widest possible abstract terms. In the context of cloud architecture, this means that in contrast to the tactically oriented DevOps role, the above strategically significant architectural role ought to be rightly designated as BizArch. Such combinations occur because the looping mechanism in OODA is a guided search that is looping across organizational levels and responsibilities in trying to solve the problem posed by the top-level FR-DP. Here OODA is genuinely heterarchical in creating and encouraging information flows across organizational/disciplinary spaces. In this process, OODA brings about creative mash-ups such as the aforementioned DevOps and BizArch. For example, ArchOps is a role being popularized by the Amazon Web Services [57,58]. In a similar vein, other such mashups may include GovArch and BizVend (with Gov for governance and Vend for vendor relations), etc.

While OODA is about decision making in rapidly evolving strategic and tactical terrains, Boyd also highlighted the importance of exploiting strategic asymmetries between allies and combatants. Between two or more allies and/or adversaries, asymmetries may exist along various dimensions, including asymmetries in wealth, culture, manpower, mental models, technological prowess, physical skill-sets, team cohesion, group dynamics, etc. For example, the bubble canopy, as well as hydraulic controls pertain to technological asymmetries. But amongst all the above asymmetries, mental model asymmetries are unique in that they fall out of heterarchical knowledge asymmetries. And as indicated in section 5, when corporations (such as Borders versus Amazon) lock horns across these asymmetries, they have the power to systematically and inexorably lay waste (in classic OODA-style) the knowledge-gapped corporation or nation.

Fig. 9c captures a few of the OODA patterns whereby asymmetric fast-transients could make or break a corporation or a nation. The context is that of a far-flung, multi-national corporation that is hierarchically administered. The view is the top-down view as was described in the context of a similar view shown in the top-

left of Fig. 9b. The various OODA strategies may be characterized as listed below:

i. Base OODA in sync with apex and executing harmoniously: This is the benchmark case where the apex and basal layers of the organization are working in close coordination (both in time-synchrony as well as in policy). In the cloud context, it means that the organization is well aware of the heightened cloud-cadence and is fundamentally organized, top-to-bottom with this in view.
ii. Heterarchical connect, and working in sync: This is the more demanding benchmark that requires heterarchically-hierarchical units of a far-flung multi-national corporation being able to work in close coordination. In the cloud context, it means that the corporation is deft and experienced in navigating the international regulatory strictures regarding data location, data privacy & security.
iii. Base OODA faster than apex & lacking in coordination: This is where the apex and basal layers are asynchronous, with the basal layers evolving at a much faster rate. This situation is not uncommon, given that many corporations treat technology merely as an enabler, and not sufficiently strategic. In extreme cases, it could lead to counterproductive corporate pathologies such as insubordination and toxic workplace cultures. Such a situation might arise in the cloud context if the top-management bought into the cloud-as-hype without properly understanding its strategic implications. In other words, there was no serious rethinking of the current architectural strategies in view of the cloud.
iv. Base OODA slower than apex & lagging in execution: This could happen when the founders of the firm who are technically and managerially astute are leading the firm, but under growth pressures they went on a hiring spree that failed to do due diligence and quality check.
v. Heterarchical disconnect unknowingly at cross purpose: This is the classic case of "*the left hand not knowing what the right hand does* [59]". Given that there are no linking mechanisms between far-flung multi-national corporate units, the disconnect continues unabated for long durations. In the cloud context, such multi-national corporate dysfunctions could be disastrous given the fact that miscommunications and cultural insensitivities could escalate rapidly out of control (both within the firm as well as in the larger marketplace sense).
vi. Base OODA intimate with apex OODA & executing faster and at cross-purpose: This is clearly corporate sabotage. In a legacy corporation, there usually exists sufficient checks and balances to make sure that such intentional and highly coordinated actions and their actors do not find refuge. But in the cloud context, given the speed at which policies and personnel can change, it is not unlikely that highly coordinated sabotage teams could take residence, and no one is the wiser.
vii. Adversary intimate with your OODA & executing cross-purpose deep within & at a faster rate: This is the case of an external agent that is somehow privy to the internal corporate strategies & technological initiatives. Being intimate with the corporate agenda and capabilities, such an agent could competitively outsmart the corporation. This is the case of corporate espionage. In the cloud context, a single breach (at the firm, partner or vendor boundaries) could drop unwanted listening assets within the vast sprawl of the firm's cloud belongings. Stigmergic listening (see section 3) would fall in this category.
viii. OODA Swarm attack--some internal, some external, some that change shape: This is akin to the DDoS Distributed Denial of Service attack. The classic DDoS swarm is synchronous and distributed but coordinated (in an algorithmic sense). It, however, lacks a clear center which could be targeted. Also, what it may lack in sophistication, it makes it up in the sheer number of resource exhausting attacks that are launched. A CAS swarm is similar, but it is shape-shifting and does not have to be synchronous. It could, therefore, play out in time, giving it more ambiguity and cover. An OODA-CAS swarm could be asynchronous, shape-shifting and executing along asymmetric fast-transients. If such sophistication exists on the cloud, it would probably be at the behest of a state sponsor.

Given the close alignment between OODA and AD, the question is what added value does OODA provide AD? Likewise, what added value does AD provide OODA? OODA was formulated for the purpose of establishing dominance in asymmetric warfare; i.e., how to exploit subtle differences (in mental models, technological prowess's, physical skill-sets and group dynamics between adverse as well as aligned participants) in order to obtain strategic (and often changing) objectives in fluid, fast-changing environments. In contrast, AD was formulated for the explicit purpose of establishing how design may be "*made into a science* [28]." It is true that both OODA and AD track closely when mapped along K|H; but they are not dealing with the same issues. AD is more generic than OODA and could be used to enhance the design aspects of OODA. In other words, OODA could benefit from the logical tripartite mappings between FR's, DP's and PVs. In a similar vein, AD could benefit from OODA in recognizing the strategic significance and asymmetric competitive value of certain key elements of a proposed design.

10. Iterative Axiomatic Maturity Diagram (AMD) Ensembles

If failure could be de-stigmatized, it has potent stigmergic value in systematically learning and becoming familiar with the design landscape. Unfortunately, the various "Fail-X" phrases in use today have created needless confusion.

Failure as a worthwhile end goal does not make sense; it only has value in an interim sense when it is being harnessed for learning the topography of a complex design surface or improving a given design that is flawed. It is never the end-goal. A company that prides itself on delivering nothing but failures will cease to exist before long.

Following are two Fail-X listings, the former which could lead to stigmergic learning, and the later which could very well thwart it. From a strategic OODA perspective, we would want our own teams (and those of our allies) to embrace the former while encouraging our adversaries to embrace the later:

I. Fail-X's that encourage stigmergic learning:
   i. Safe-Fail [60]: Through deliberately engineered failures, designers learn the complex terrain being navigated.
   ii. Fail-Fast: To find if a system is ill-designed, best to quickly test it early on, rather than prolonging the discovery of the flaw.
   iii. Fail-Often: Setting up a sequence of small bite-size goal-posts, which creates opportunities for many successes and failures.
   iv. Fail-Early: Provide greater latitude for failure, but only during the early phases of a project which could create the right attitude of seriousness towards successes and failures.
   v. Fail-Forward: In failure, take advantage of the lessons learned for the next iteration.
   vi. Fail-Small: Set up small, bite-size goal-posts, which creates opportunities for small successes & failures.
   vii. Fail-Well: Compartmentalize & Contain the Failure from Spreading. This agrees well with the uncoupled/decoupled design in AD.
   viii. Fail-Safe: In production, if and when you fail, fail safely by not endangering life & property.

II. Fail-X's that discourage stigmergic learning:
   i. Fail-Backward: Lack of team resilience/ability to recover from failure; hence no learning.
   ii. Fail-Big: Fail colossally at something big. This is high risk for high rewards. It is not driven by incremental, iterative stigmergic learning.
   iii. Fail-Badly: When the system fails, it is catastrophic; therefore, no learning.
   iv. Fail-Silent: When failure happens, it is suppressed from public view with no indication of failure; therefore, no learning.
   v. Fail-Deadly: Mutually Assured Destruction (MAD) such as is the case for nuclear deterrence. Cloud development has yet to reach the stage of cloud-wars where MAD may be relevant.

These are some of the colloquial ways to characterize candidate designs. In general, designs (as well as design processes) may be characterized and critiqued in at least six different ways:

   i. Viability of the select design in regard to the CRs/FRs. This includes the independence axiom.
   ii. Performance of the select design in functional comparison to a family of other valid designs. This includes the information axiom as a selection criterion.
   iii. Performance of the select design in cost-comparison to a family of other valid designs.
   iv. Performance of the select design in view of the change-dynamic that is evident in the phylogeny of the problem domain. In other words, what is the rate of change in the FRs? And how does the candidate design cater to such a rate of change in the FRs?
   v. Performance of the design process that created the design with respect to time-to-market? How quickly can such designs be designed and implemented?
   vi. Performance of the design as part of a family of other designs that exist in the same enclave (for example, being hosted by the same cloud vendor).
   vii. Performance of the design as well as the design process in catering to emergent FRs that may not be known a priori until the design/design-process has matured sufficiently.

While the first two criteria from the above list do have axiomatic representation, the last five items do not have representation (except perhaps as rigid, a priori constraints). The Agile methodology has made valuable contributions in addressing the last of the above cases, namely how to go about designing in the Complex realm (see Fig. 7) where FRs are emergent and unknown a priori, where statistics is rare, and where fat-tails & power-laws are common. Axiomatic Design could learn from the Agile gambit and address this lacuna in a principled fashion without compromising its holistic strengths. Note that in a similar vein, items iii-vi above also remain unaddressed in AD.

As discussed earlier in Section 9, the Problem Abstraction Phase (i.e., PAs in Fig. 9b) arches upward on the inductive design trace. But induction takes its own time; it does not have the same rapid cadence of deductive logic. In other words, it takes time to marshal the necessary holistic view that the axiomatic approach prefers.

Also, the inductive component is more error-prone. Misdiagnosis of the problem can be costly. Agile adopts various Fail-X approaches in order to mitigate this risk. For example, the Safe-Fail approach [60] accommodates FR's being emergent in a CAS sense. In other words, the FR's do not exist a priori. Referring back to the manner in which the β-layer forms in a CAS setup (Fig. 3a), Agile seems to suggest that FR's are stigmergically emergent (using agile-style Post-It notes, etc.). They cannot be discerned a priori except through repeat trials and errors. They emerge along the design pathway that is engaged in solving a larger problem. In such a situation, a top-down linear approach such as the waterfall model that does not iterate back to the root FR's will misdiagnose the problem and therefore fail to address the emergent issues.

As would be the case in other approaches, true rapidity/agility in Agile occurs in problem contexts that have been well plowed. And in problem contexts that are more bottom-up, inductive and tentative in nature (such as Cynefin's Complex regime), Agile adopts an iterative

approach of sprints and retrospectives which would necessarily take longer. If that is the case, where exactly is the agility in Agile? In order to understand the agility aspect of Agile, let us compare the waterfall approach to Agile. If in waterfall-type approaches, spurious FRs are being addressed while relevant FRs (which happen to be emergent) are left unaddressed, it is obvious that such designs are never timely. As an analogy, if the train arrives at the wrong destination, it is indefinitely late for arrival at the right destination. The agility of Agile is in solving the right problems, and not because it is inherently agile. In other words, it is fundamentally on the basis of the emergent FR problem (which is real and salient, especially in a knowledge economy) that Agile has staked its claim on the totality of design.

No	Agile (X over Y)	Axiomatic Design (X because of Y)
1	Individuals & Interactions over Processes and Tools	Individuals are able to creatively interact and engage vigorously because of the Processes and Tools that AD provides.
2	Working software over comprehensive documentation	Robust Products because of Principled, Comprehensive/Holisitic design which includes Succinct (FR=DMxDP) Documentation
3	Customer collaboration over contract negotiation	Customer Collaboration via Contractual Obligations as per CR: Customer Requirements which could be flexibly structured to support CR/FR emergences.
4	Responding to change over following a plan	Resilient to change because it is following a Plan (i.e., Design) which includes Design-for-Change.

Fig. 10. X over Y (Agile) vs. X because of Y (AD)

However, such a broad claim on the totality of design needs to be challenged. For example, the differences between Agile versus a more formal/structured approach has been couched in the Agile manifesto as "X over Y [61]". Such a conflicted approach is unnecessary. For example, in the AD context (as shown in Fig. 10 above), it is more than likely that it is a case of "X because of Y". The industry is increasingly becoming aware of many of the Agilist blind-spots [62-64].

Agile fails to master the problem of design in at least two significant ways:

- Design is holistic. Piece-meal designs seldom scale, especially in the Complex regime. Given the significance of the emergent FR problem (especially in a knowledge-economy), Agile downplay the very concept of system-wide/holistic design. This can be a problem when considering cybersecurity which tends to expose unaddressed gaps in non-systemic, ad-hoc designs. However, it would not be too difficult to bring in holism (especially during the final-stage sprints and retrospectives).
- Without formal documentation, the stigmergic pattern-making process rarely takes root. In the current technological context where anything and everything is being dutifully noted and recorded (thanks to myriad IoT's and other instruments of constant vigil), it is unfortunate that seamless & effortless documentation is not de-jure in the design realm. The Agile manifesto on documentation (i.e., "*working software over comprehensive documentation*") is therefore misguided. In the modern age, design documentation should be automated and effortless. The bias that Agile has against documentation is quite anachronistic and self-defeating especially given the role that documentation plays in the formation of stigmergic patterns. Once again, it would not be too difficult to correct this problem.

From an AD perspective, there are valuable insights to be learned from Agile. As mentioned earlier, the key distinction between Agile and AD is in the context of emergent FRs. This is an area where AD could learn from Agile. While keeping the holistic view, the Axiomatic approach can strategically borrow Agile's iterative stance (which, incidentally agrees with the OODA loop). This is precisely what has been proposed in [65] wherein Puik & Ceglarek have advocated using the Axiomatic Maturity Diagram (AMD) in an ingenious way for bringing synergy between the Axiomatic and Agile approaches. The following discussion extends the AMD approach by explicitly adding the stigmergic tightening of the AMD patterns along the time dimension (see Fig. 11 below).

As suggested in [65], the key is in understanding the competing thrusts of the main three drivers of the design process (Fig.11a):

- Do the Right Thing: Find the right set of FR's, DP's, PV's) that has total, holistic capture of the problem at hand and a design for it that satisfies the independence axiom (Fig. 11/X-Axis).
- Do Things Optimally: Minimize the holistic, system-wide information content among candidate designs in order to find the right solution (Fig. 11/Z-Axis)
- Do Things Fast: Navigate the design space in order to reach the target solution at a rapid pace; i.e., minimize θ, the temporal splay along Fig. 11/Y-Axis)

Emphasizing any one of these drivers stand-alone or even two-by-two will only succeed up to a certain point. As suggested in [65], the Axiomatic Maturity Diagram (AMD) could be modified ever so slightly to incorporate Agile's iterative insight (Fig. 11.b).

Normally, the AMD is a 2D plot that captures the path-dominance between Independence vs. Information axioms. For de-novo designs, usually the Independence Axiom dominates the initial stages; it is only after the design-trace has reached a certain level of maturity that the information axion is triggered. For de-vetus cases, the information axiom may gainfully be put to use early on given the level of experience and history that is readily

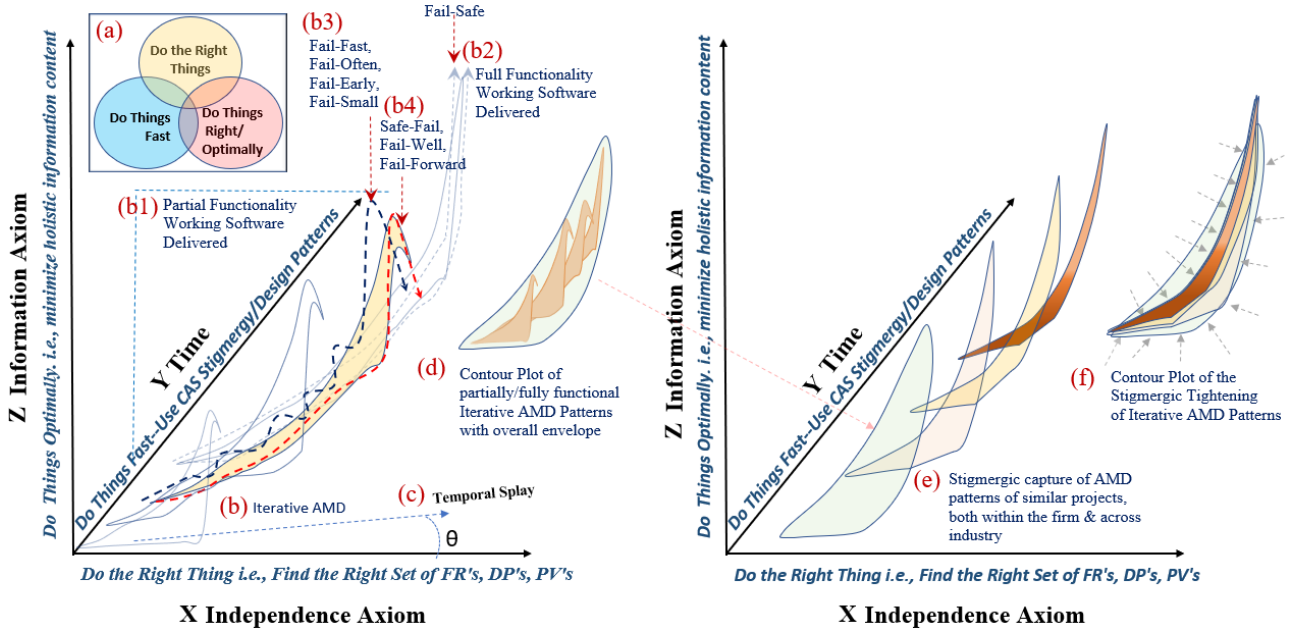


Fig. 11. Iterative Axiomatic Maturity Diagram (AMD) Ensembles (adapted from [65])

available. For the problematic case of emergent FRs (which by its nature is de-novo), an iterative approach from the very beginning (i.e., as in Probe→Sense→Respond from the Complex realm of Cynefin, Fig. 7) is probably the right way to proceed. Providing such early and repeated reality checks is one of the hallmarks of the Agile approach. As suggested in [65], the dotted curves in Figs. 11b3-11b4 capture such agilest iterations that have been abstracted as the underlying wedge. The distinction between 11b3 and 11b4 is that the former (i.e., 11b3) has a slight bias towards the information-axiom which should show up in colloquialisms such as Fail-Fast, Fail-Often, Fail-Early, and Fail-Small. In contrast, the latter (i.e., 11b4) is more conservative in its approach and might include colloquialisms such as Safe-Fail, Fail-Well, and Fail-Forward. These loose colloquialisms are merely suggestive; they are not categorical distinctions. The tan-colored wedge captures a variety of iterative/combinatorial possibilities that span between the boundaries of 11b3 and 11b4.

In each sprint (as Fig. 11b1 indicates), only a partial list of the required functionality is being delivered. There are four sprints that have been outlined, of which only the 3rd is colored tan and bounded by Figs. 11b3-11b4. Fig. 11b2 indicates the end of all sprints, with the full working functionality being delivered. Each of the sprints could have fresh additions (of emergent FR's). FR's may also be deleted from a previous partially working solution.

The slight temporal splay (θ) of the very first wedge captures the time taken in the underlying iteration (Fig. 11c). The agilest argument is that without an iterative approach, the splay would be much wider, the FRs often hastily & improperly induced, and the resultant design rigid, fragile and even abandoned. These agilest arguments are completely valid. But there is nothing stopping the Axiomatic approach from adopting an iterative stance as shown. And the comparative advantage of the axiomatic approach is that it never loses sight of the fact that the problem of design is holistic. With that ideal in mind, it is therefore motivated to reach for the holistic view.

The contour plot of the four sprints is as depicted in Fig. 11d. It captures all four partial/fully functional iterative AMD patterns along with an overall containing envelope (in light green) that captures it in abstract. Just the abstract outline is repeated on the right AMD figure in order to help capture the larger stigmergic patterns if proper documentation were to be enforced.

Fig. 11e is the stigmergic capture of AMD patterns of similar projects (i.e., the phylogeny) which may exist both within the firm as well as across the industry. Fig. 11f captures the contour plot of the stigmergic tightening of the various iterative AMD patterns as displayed in Fig. 11e. Such stigmergic tightening is feasible, given the fact that the axiomatic approach is pithy & documentation-friendly in its succinct design-matrix capture. Thus, across multiple AMD wedge-iterations (Fig. 11e-f), the axiomatic approach is capable of not just capturing the holistic demand; it is also capable of leveraging the stigmergic tightening of the AMD patterns.

In time, the AMD trace is well established and settles into a thin slice (displayed as a dark-orange slice at the center). Once the iterative uncertainty has been removed and the stigmergic wedge well established, a waterfall approach would work just as well. In other words, the complex has now been tamed (at least in part) into the complicated/simple regimes; i.e., Good Practice and Best

Practice available in the industry (see Cynefin, Fig. 7). It is indeed a waste of scarce resources to indiscriminately treat every problem as if it were always attached to the complex regime in each of its decompositional details.

11. Non-FR's from a CAS Perspective

Non-Functional Requirements (Non-FR's) dominate the design of cloud architectures. These include system-wide -*ilities* such as:

- Scalability
- Adaptability
- Reliability
- Security
- Maintainability
- Availability
- Customizability
- Testability, etc.

But as a term of common usage, the non-FR's are indeed a misnomer; for there is nothing non-functional about the concern at hand. Thus, the ability to rapidly scale up or down a certain web site based on the seasonal load at hand is most definitely a functional requirement—except that instead of it being at a final user level, it is now at a system-wide/population-wide level. It is, therefore, a failure in the design community to understand the functional domain when it asserts that the above list of requirements is somehow non-functional [66, 67].

The deeper question that, however, needs to be probed is where do these non-FRs come from? To solve the puzzle of the origin of functionally relevant non-FR's, one has to study the requirement formation as an iterative CAS process (Fig. 3b). If the regular FR's are to be found in the β1 ensemble, the non-FR's are to be found in the β2 ensemble. Thus, the so-called non-FR's are indeed FR's and therefore subject to the standard design approaches. Indeed, it is easy to project a future point where there will be a β3 ensemble one day. It is, therefore, best to acknowledge levels of design in preference to ad-hoc terminology (such as non-FR) when dealing with systems that are fundamentally CAS in nature. In other words, the various -*ilities* requirements are likely to adapt and evolve away from the current strictures. Or to put it differently, of all the ilities mentioned above, the adaptability requirement is dominant and overarching over asset/agent/artifact space as well as time. The cloud is fundamentally operating and evolving at a much faster cadence than the systems that are residing in the traditional on-prem ecosystem. Architecting such rapidly evolving systems requires the architect to understand the CAS α↔β pattern forming mechanism and consider the problem of design from the highest β-level reached thus far, as well as the projected CAS trajectory. Restating the above in Boydian OODA terms, asymmetric fast-transient's in the cloud are operating with latency in milliseconds instead of weeks or months. Missteps can be fatal. This is especially true when considering cloud cyber-security.

12. Security of Cloud Computing

Once the corporate assets have been migrated (in part or whole) over to the cloud, the legacy threat surface is significantly altered. Depending on the cloud footprint [68] (i.e., IaaS, PaaS, SaaS), the onus of securing the assets is now a joint responsibility. What was previously an in-house responsibility is now a shared undertaking that juxtaposes the evolving footprint and complexity of the vendors cloud infrastructure and operations (with its global reach and geophysical asset spread) against the cloud-maturity of the in-house architects, developers, users, and operators.

Some recent quotes regarding cloud-related security breaches in the news include:

- *A group of angry customers filed a lawsuit against Capital One...following the hack that affected more than 106 million people...the group also named Amazon Web Services, Capital One's cloud provider, alleging the tech giant is also culpable for the breach*. (August 9, 2019, [69])
- *Unprotected Database Puts 65% of American Households at Risk…data included on the 24 GB database [hosted on Microsoft Azure] is people's full names, full street addresses, marital status, date of birth, income bracket, home ownership status and more.* (April 29, 2019. [70])
- *Verizon Partner Exposed Millions of Customer Accounts...a misconfigured cloud-based file repository exposed the names, addresses, account details, and account personal identification numbers (PINs) of as many as 14 million US customers of telecommunications carrier Verizon.* (December 12, 2018. [71])
- *In December [2018], Google revealed the details of...data breach that happened...leaving the data of close to 52.5 million Google+ users vulnerable to hackers...Google+ is shutting down in 2019.* [72]
- *Between July and September 2018, hackers leveraged the "view as" feature on Facebook to steal tokens for access profiles. This breach compromised the personal details of close to 29 million users across the globe. It divulged personal information including names, phone numbers, email addresses, and other personal details Facebook collected over time. The breach was disclosed to the general users on September 28.* [72]

According to a recent multi-factor industry survey of the leading concerns of 400,000 cloud cybersecurity professionals [73]:

- Biggest threats to cloud security – Misconfiguration of cloud platforms (62 %).
- Legacy on-prem security tools are ill-designed for the virtual, dynamic and distributed cloud. Traditional security solutions either don't work at all in cloud environments or have only limited functionality (84%).

- Operationally, the leading security control challenges (SOCs) include (a) poor visibility into infrastructure security (43%); (b) compliance (38%); (c) Setting consistent security policies across cloud and on-premises environments (35%); (d) security not keeping up with the pace of change in applications (35%).
- Compared to an on-prem deployment, 49% of the respondents believe that the public cloud poses a greater security risk, 30% responded about the same, and only 17% responded in favor of the cloud.

Many cloud-related cyber insecurities are related to poor design. These design issues could show up as misconfiguration, poor consistency, lack-of-transparency, inadequate security tools, inability to keep pace with application changes, etc. With the rapid evolution of the threat surface, it is no wonder that even well-configured systems sprout leaks.

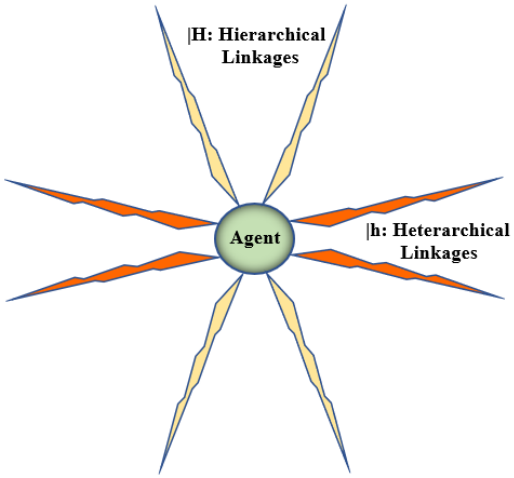


Fig. 12. Defense-in-Depth/Width (Hierarchy vs. Heterarchy)

But the greatest cybersecurity vulnerability in cloud computing has to do with a flawed Defense-in-Depth (DiD) mindset that currently dominates the industry (see section 5 for a prior discussion on this topic in connection to knowledge hierarchy/heterarchy). There are two major problems with the DiD posture [24]:

- **Technological Exposure**: Rapid evolution of technology exposes interface mismatches across the various onion layers. Ubiquitous and remotely located edge nodes & IoT devices can be compromised. Each system is designed for a certain system range. Swarm attacks can be designed to overwhelm the elasticity of these system ranges. While traditional attacks used to be sequentially directed against any single layer, modern swarm attacks target multiple layers simultaneously. Traditional attacks were directed at the network layer that is easier to detect, while modern attacks target the application layer that is harder to detect.
- **Socio-Technical Exposure**: Human cognitive biases and vulnerabilities are constantly being probed via social engineering techniques and strategies such as weak passwords, robocalls, and phishing attacks. The performance overhead from any of the security defenses can reach a point of intolerance for the human agents to become overwhelmed and then switch it off.

In contrast to Defense-in-Depth, the emerging Defense-in-Breadth (DiB) approach also considers the wider angle of vulnerabilities. NIST has helped in defining and contrasting the two [74]:

- **Defense-in-Depth**: *Information security strategy integrating people, technology, and operations capabilities to establish variable barriers across multiple layers and missions of the organization.*
- **Defense-in-Breadth**: *A planned, systematic set of* ***multidisciplinary*** *activities that seek to identify, manage, and reduce risk of exploitable vulnerabilities at every stage of the system, network, or subcomponent life cycle (system, network, or product design and development; manufacturing; packaging; assembly; system integration; distribution; operations; maintenance; and retirement).*

It is the multidisciplinary emphasis in DiB that provides the hint that the underlying system and the concerns thereof are heterarchical in nature (see Fig. 12). In contrast to traditional hierarchical systems, the combinatorial space that the architect has to master when dealing with heterarchical systems is vastly more complex and expanded. As shown in section 10 above (i.e., iterative AMD), such systems may best be designed in a systematic and principled way by leveraging the stigmergic patterns that accrue over time.

13. Econo-CAS-Strategy in the Cloud

None of the top four companies by Market Cap from the year 2000 (i.e., General Electric, ExxonMobil, Pfizer, and Citigroup) were able to retain their leadership positions. Instead, it is now Microsoft, Amazon, Apple, and Alphabet (Google). Among these four, three have strong vendor presence in the cloud; only Apple is weak and is dependent on Amazon. As reported in [75]:

> *Apple is deeply reliant on AWS to operate core parts of its business, even though doing so means working with a soon-to-be-rival in online video and a current competitor in areas like artificial intelligence, streaming music, and smart home products.*

Apple is facing a similar problem as was the case with Borders. Many other major firms (including Netflix) face the same conundrum that Borders faced when dealing with a power-law driven knowledge-economy.

As was mentioned earlier, cloud computing is strategic and needs to be considered in warfare terms. Critical assets deployed on the cloud are no longer business-as-usual; it, in fact, is being positioned on an asymmetric, winner-take-all, fast-transient OODA basis. In such a

context, every aspect of the OODA loop needs to be examined for its asymmetry potential.

In order to scale a higher peak, one often has to climb down from the current summit. Traditionally, it is economics that has provided navigational guidance in the summit-to-summit route-finding endeavors. But what if

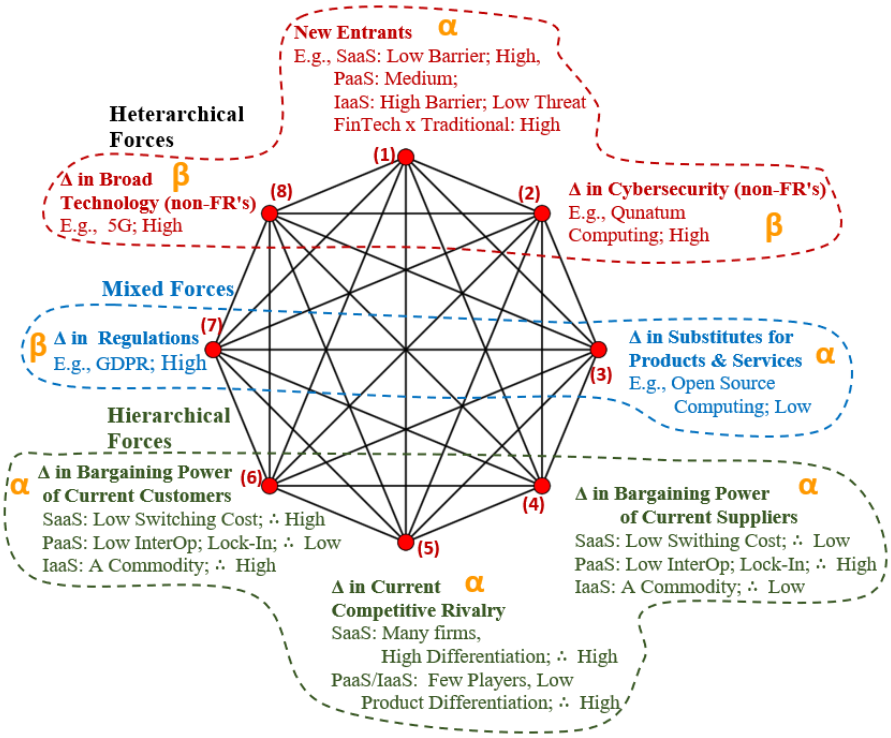


Fig. 13. CAS-Based Extension of Porters 5 α-Level Competitive Cloud Forces [39] to Include 3 β-Level Forces

the economic heuristics we have here-to-fore depended on, themselves change? What if the governing economic rules are being re-written to accommodate fat-tails even as the corporate Econo-strategist tries to navigate across the shifting landscape? It is in this sense that cloud computing is "*both an existential threat and an irresistible opportunity* [76]."

It is indeed an existential threat for those wedded to the status quo; but it is also an opportunity for those willing "*to climb down*" from their current summits and look at the de-novo Econo-design landscape that is opening up. These include social network economies (Facebook, LinkedIn, Twitter etc.), big-data plays (GreenPlum, Cloudera/Hortonworks, Palantir, etc.), streaming economies (Netflix, Hulu, Amazon Prime, YouTube TV, etc.), gaming plays (PlayStation Now, Shadow, GeForce NOW, etc.) and others.

As one steps back from the micro-view in order to then take in the big-picture/macro-view, it is becoming increasingly clear that the architectural design of the cloud computing play is anything but simple and straight-forward. For example, in the context of cybersecurity, the attack surface is the map of all ports of entry/exit whereby an attacker may launch an attack, and/or spirit off corporate assets. And as the business grows, the dangers of cyber-insecurity increase as the attack-surface proliferates and mutates across pathways and resources that the corporation does not fully command.

Likewise, the economic attack surface for cloud-based corporate ventures is orders of magnitude more complex than the traditional on-prem ventures. Once the corporation establishes key assets in the cloud, it is operating in a shared environment where its business activities leave open and visible stigmergic traces. These include what is openly known about the strengths/weaknesses of the cloud vendor.

Once the business-model proves viable in the cloud, the competitive attack surface can bring in a swarm of traditional/non-traditional challengers unconstrained by erstwhile barriers-to-entry that have either been leveled or rendered irrelevant. The cloud fundamentally lowers many of the traditional barriers to entry. For example, it is true that the Chinese firm Ant-Financial was rebuffed in establishing a FinTech foothold in the U.S via the purchase of MoneyGram [77]. But there are no such barriers for the rapid migration/replication by native agents of a successful business model such as the Ant-Financial. And it is in such a rapidly evolving cloud ecosystem, that a well thought out, adaptive architectural design could take advantage of the new economies-of-scale and elasticity that the cloud makes available. Creating a viable cloud enterprise increasingly involves architecting of a complex adaptive system [18]. The following discussion is a CAS-based extension of Porters original five α-level competitive cloud forces to include three β-level forces for a total of eight (see Fig 13 above).

The fundamental difference to notice is that all eight competitive forces are at the periphery and jostling with each other for dominance; whereas Porter's framework had centralized on Current Rivalry as the core node. Also, the forces are characterized using the CAS α-β notation to help identify the level at which agents are forming their strategic intent. Porter's formulation had the original five forces occurring at the inter-agent α-level. The three new entries (i.e, Cybersecurity, Technological Shifts & Global Regulations) exist at the β→α level. These additions are just a sampling of the missing entries in the original formulation; there could, of course, be many more than just these three additions.

The set of eight competitive forces is split into three groups: Hierarchical, Heterarchical and Mixed. Hierarchical forces are incremental, slow-moving, and works within the confines of the current competitive landscape. For example, if there is a PaaS Lock-In (as is at node 6 in Fig. 13 above), it is not easy to shift out of this. Heterarchical forces could also be slow-moving. But a few are strategic, rapid and works orthogonal to the current competitive landscape (e.g., 5G at node 8, Fig. 13). Mixed items have both heterarchical as well as hierarchical elements within them.

To illustrate the framework, consider the IaaS cloud offering. Here, there is a low threat from new entrants (see node 1, Fig. 13) given the sizable upfront Capex outlays that the hosting of a full-stack cloud infrastructure requires. Nevertheless, given the state-level strategic significance of the cloud for any given nation, the above deep-pocket Capex stricture, therefore, does not preclude state-level agents from entering the competitive landscape. Yet given the speed at which the underlying technology shifts (for example, the impending shift from 4G to 5G), any state-sponsor faces steep odds in keeping up with the fast-moving heterarchic front. State sponsors tend to be hierarchical in nature; they have yet to master the unwieldy socio-technical heterarchic-hierarchies of today. But when Cybersecurity issues are raised (see node-2, Fig. 13), the state may have a strategic incentive to take on the infrastructural challenge head-on. All three competitive forces as shown in brown (New Entrants, Technological Shifts, and Cybersecurity) fall in the heterarchical space as each of these driving forces has the power to shift the competitive landscape from a point-of-view that is orthogonal to the α↔α focus (in Porter's framework), and that too, potentially overnight.

Now consider the hierarchical tranche. From a consumer's perspective, the shift from Capex to Opex has fundamentally lowered the bargaining power of the IaaS supplier as there is very little lock-in (see node 6, Fig 13). If one of the competitors offers a similar or enhanced set of infrastructural offerings at a sufficiently competitive price (competitive enough to overcome the switching costs), the consumer has every incentive to shift. The real bargaining power of the supplier exists not at the level of any given α-agent-level supplier-consumer contract; instead, it is in successfully leveraging the β-level *-ilities* which the industry mistakenly identifies as the non-FR's (see section 11).

The mixed-set includes the realm of substitutes that rise up either hierarchically within the current competitive landscape, or heterarchically orthogonal to it. The mixed-set also includes the shifting regulatory fractal landscape that has a global reach. It is "mixed" in the sense that much of the regulatory landscape is hierarchically constrained by precedent; but occasionally, the regulatory bodies do reach across and claim jurisdiction (especially in uncharted areas that new technological mash-ups have recently opened up). From a regulatory perspective, any of the major state-level agents have the power to overnight shift the competitive landscape with the understanding that when dealing with a CAS, the overall system will react back. In other words, in a CAS system, any unilateral action would face resistance at various levels. This is the reason why all the eight forces (and there may be many more β-level forces) are depicted at the periphery and engaging each other in a complete network. Porter's framework only considered the α-level agents and the direct interplay between them; it is silent about the β-level play wherein much of the competitive landscape has shifted.

14. Adaptive Architecture

Organizational Psychologist, Karl Weick originated the modern concept of loose vs. tight coupling in the mid-1970s. Coupling plays a central role in Axiomatic Design. The following discussion frames both the Weickian as well as the axiomatic approach on coupling in the CAS framework.

As Prof. Suh noted in [28, 29]:

When the design matrix [A] is diagonal, each of the FRs can be satisfied independently by means of one DP. Such a design is called an uncoupled design. When the matrix is triangular, the independence of FRs can be guaranteed if and only if the DPs are determined in a proper sequence. Such a design is called decoupled design. Any other form of the design matrix is called a full matrix and results in a coupled design.

Likewise, as Prof. Weick noted in [78]:

...loose coupling is evident when elements affect each other "suddenly (rather than continuously), occasionally (rather than constantly), negligibly (rather than significantly), indirectly (rather than directly), and eventually (rather than immediately).

The dynamics embedded in the Weickian concept of loose coupling may be best understood using CAS-framework that emphasizes levels, temporal spans as well as impact:

- **Levels**: Rejects direct intervention (α→α) in favor of the indirect (α→β→α) (i.e., *"indirectly (rather than directly)"*)
- **Time Delay**: Rejects the chronic and continuous in favor of episodic in the short-term or long-term (i.e., *"suddenly (rather than continuously), occasionally*

(rather than constantly), …eventually (rather than immediately)")

- **Impact**: Favors small versus the large impact (i.e., *"negligibly (rather than significantly)"*)

In [79], Prof. Weick further notes that:

By loose coupling, the author intends to convey the image that coupled events are responsive, but that each event also preserves its own identity and some evidence of its physical or logical separateness.

Based on the above characteristics, Prof. Weick defines the following types [78]:

If there is neither responsiveness nor distinctiveness, the system is not really a system, and it can be defined as a noncoupled system. If there is responsiveness without distinctiveness, the system is tightly coupled. If there is distinctiveness without responsiveness, the system is decoupled. If there is both distinctiveness and responsiveness, the system is loosely coupled.

A system of CAS agents is responsive when sufficient β-level orchestrating patterns have accumulated (within a given iteration) in order to help α-level agents respond in a coordinated fashion to the events at hand. Furthermore, the system is distinctive (within a given iteration) when the α-level agents have retained sufficient degrees of freedom (DOF's) to move orthogonally to the restrictions placed by the previous β→α coordination.

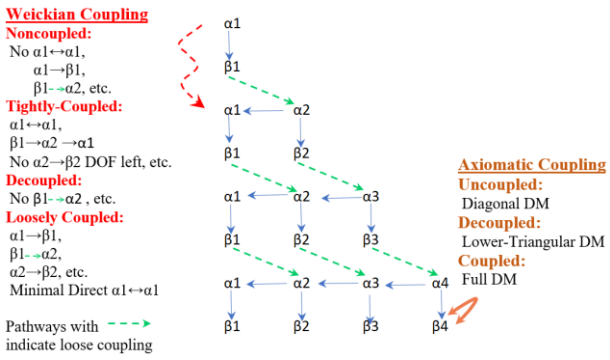


Fig. 14. Axiomatic vs. Weickian Coupling

Based on the above CAS reframing, the four Weickian categories (see Fig. 14 above) may be restated as follows:

- **Noncoupled**: No α↔α, α→β, No β→α
- **Tightly Coupled**: α↔α & β→α exists, but no orthogonal α→β exists in the next iteration (i.e., no degrees of freedom left)
- **Decoupled**: α→β exists; but the β-level patterns are either non-existent or not mature enough to provide β→α coordination and control.
- **Loosely Coupled**: Minimum α↔α; Both β→α coordination & control as well as next orthogonal/iterative α→β exists (i.e., remainder degrees-of-freedom exist even after β→α coordination & control).

Loosely coupled systems are an organizational ideal for social, technical as well as socio-technical systems. They are neither brittle (i.e., they are resilient in the face of change-dynamics) nor anarchic (i.e., not lacking in coordination and concerted action). The resilience is because (when faced with novel situations) the agents retain sufficient degrees-of-freedom to coordinate and self-organize new structures outside the current strictures. But for the most part, the agents are operating within the current strictures. Or as Weick asserts in [80]: "*the real trick in highly reliable systems is somehow to achieve simultaneous centralization and decentralization.*"

Here, the centralization mandate is achieved via β→α coordination [81]. The α-level decentralization captures the remaining orthogonal degrees of available freedom given the current state of centralization. As the overall system gears across multiple α→β iterations (see Fig. 3b), and the agents get more organized, the loose-coupling frontier shifts to the highest iterate. Everything lower down is stable, highly orchestrated, coordinated, waterfallish, and tightly coupled. Since there are no degrees of freedom available at the lower levels, consequently the problem of design itself does not exist. An example of this is assembly programming; i.e., for most higher-level programming language cases, there are no degrees of freedom left at this base layer. All the remaining degrees of freedom exist in one of the higher-level programming languages. Thus, the problem of design exists only in the outermost iterate where degrees of freedom exist for the respective agents. All the lower levels exist as constraints and context for the problem of design.

The fundamental problem of design only exists where there are degrees-of-freedom available, which typically exists in the outermost layers of the CAS system. The problem of coupling in Axiomatic Design, therefore, exists at the outermost iterate layer (see bottom-right in Fig. 14). Given the symbiotic reach of both of these systems, the AD framework (as well as that for Agile) could, therefore, work well with the Weickian loose-coupling framework. Across each iteration (as the system gears up), the axiomatic approach of preferring uncoupled/decoupled in preference to the coupled is, therefore, sound advice.

The question now is how does the above discussion relate to Cloud architectures? As we indicated in section 9, the cloud is anything but static. Instead, it is a socio-technical CAS system that is rapidly morphing, adapting and evolving. And it is in this context that the Weickian approach could work symbiotically alongside the Axiomatic approaches; i.e., via Weickian loose-coupling at the basal layers alongside AD uncoupling/decoupling at the growing meristem of the CAS edifice. For example, as in [12], by using an intermediary broker mechanism, direct α↔α interactions are streamlined and minimized. Instead, new broker agents are used (under the guidance of β→α patterns) for indirect information exchange between agents. AD could assist in the establishment of such a CAS scaffold. This would provide for framing the loose-coupling between α↔α. Once the α-β architecture of the loosely-coupled bipartite-CAS framework is established, AD could further be used in designing each of

the layers as per the AD coupling/de-coupling logic. In a follow-up report we intend to illustrate CAS-based loosely-coupled designs along with various other cloud-related design patterns.

15. Conclusions

A substantial number of theoretical/practical issues related to an adaptive architectural design surrounding cloud computing was covered in this report. A variety of beneficial frameworks were formally integrated for the first time in providing an integrated, overarching approach to help tackle the problem at large. These included Axiomatics, Knowledge Hierarchy/Heterarchy, Cynefin, OODA, Stigmergy, CAS, AMD, Non-FR's, and Adaptive Loose-Coupling.

Some of the key findings include:

- The cloud is a CAS system. The architectural design of the cloud requires coming to terms with the underlying CAS dynamics & patterns.
- The framework of knowledge hierarchy/heterarchy provides a simple but robust approach to help many orthogonal, but mutually supportive frameworks such as Axiomatics, OODA & Cynefin.
- Time-axis extension of the iterative AMD approach as reported in [65] provides a pragmatic way to bring agility, axiomatics and pattern logic together in one place to help bring about principled design in an agile fashion.
- Using CAS, the non-FR's have been folded into the standard FR-DP mappings.
- Security of the cloud has been critiqued, both from a design perspective as well as from a Defense-in-Depth/Defense-in-Breadth perspective.
- Using CAS, the fundamental concept of coupling has been broadened to include loose-coupling which powers much of modern technology, including the cloud. We hope to illustrate this approach with a follow-up report.

Unfortunately, we could not cover the following issues that remain to be addressed:

- Cloud computing has matured enough to put forth well-established design patterns. We did not cover the vast cloud pattern landscape.
- We did not address the realm of microservices that atomize the responsibilities to such a fine degree that the problem of coupling becomes moot.
- Addressing each of the non-FRs is itself a major undertaking. This was not addressed.
- We did not address any of the myriad vendors and their offerings.
- Retrospective and Prospective cloud computing trends were not discussed

Even so, we do hope that the above report is a good start for placing design in the center stage of modern cloud computing.

References

1. Wikipedia. Borders Group. https://en.wikipedia.org/wiki/Borders_Group (accessed on July 14, 2019).
2. Wikipedia. Power law. https://en.wikipedia.org/wiki/Power_law (accessed on July 14, 2019).
3. E. Siegel. Predictive Analytics: The Power to Predict Who Will Click, Buy, Lie, or Die. Wiley. 2013.
4. J. Surowiecki. The Wisdom of Crowds. Anchor. 2005.
5. P. M. Mell, T. Grance. The NIST Definition of Cloud Computing. Special Publication (NIST SP) - 800-145. September 28, 2011 https://www.nist.gov/publications/nist-definition-cloud-computing
6. T. V. Vleck. Project MAC. Available at https://multicians.org/project-mac.html (accessed on July 14, 2019).
7. Wikipedia. Intergalactic Computer Network. https://en.wikipedia.org/wiki/Intergalactic_Computer_Network (accessed on July 14, 2019).
8. Wikipedia. Timeline of virtualization development. https://en.wikipedia.org/wiki/Timeline_of_virtualization_development (accessed on July 14, 2019).
9. R. Chellapp, Intermediaries in Cloud-Computing: A New Computing Paradigm, Presented at INFORMS Meeting, Dallas, 1997
10. P. P. Grassé. Insectes Sociaux VI. 79. 1959
11. H. V. D. Parunak. A survey of environments and mechanisms for human-human stigmergy. 2006. Environments for Multi-Agent Systems II: 163-186.
12. C. Fehling, F. Leymann, R. Retter, W. Schupeck, P. Arbitter. Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications. Springer; 2014
13. T. Erl, R. Cope, A. Naserpour. Cloud Computing Design Patterns. Prentice Hall; 1 edition (March 19, 2017)
14. S. M. F. Akhtar. Big Data Architect's Handbook: A guide to building proficiency in tools and systems used by leading big data experts. Packt Publishing, 2018
15. T. Erl, W. Khattak, P. Buhler. Big Data Fundamentals: Concepts, Drivers & Techniques. Prentice Hall, 2016
16. D. W. Tollen. The Tech Contracts Handbook: Cloud Computing Agreements, Software Licenses, and Other IT Contracts for Lawyers and Businesspeople. American Bar Association. 2016.

17. U. Wilensky. Ants. NetLogo Models Library. http://ccl.northwestern.edu/netlogo/models/Ants 1997
18. J. Urquhart. Cloud is complex—deal with it. Jan 8, 2012. https://gigaom.com/2012/01/08/cloud-is-complex-deal-with-it/ (accessed on July 14, 2019).
19. J. H. Holland. Studying complex adaptive systems. Journal of Systems Science and Complexity 2006, 19(1), 1-8.
20. S. A. Karthikeyan. Azure Automation Using the ARM Model: An In-Depth Guide to Automation with Azure Resource Manager. Apress; 1st ed. November 12, 2017
21. J. Thomas, A. Zaytseva. Mapping Complexity/Human Knowledge as a Complex Adaptive System. 2016 Wiley Periodicals, Inc., Vol. 21 No. S2. DOI 10.1002/cplx.21799. 24 June 2016. https://onlinelibrary.wiley.com/doi/abs/10.1002/cplx.21799
22. Wikipedia. Horizontal gene transfer. https://en.wikipedia.org/wiki/Horizontal_gene_transfer (accessed on July 14, 2019)
23. J. Lee, T. Wei, S. K. Mukhiya. Hands-On Big Data Modeling: Effective database design techniques for data architects and business intelligence professionals. Packt Publishing. 2018.
24. D. Igbe. Defense in Breadth or Defense in Depth? https://www.cloudtechnologyexperts.com/defense-in-breadth-or-defense-in-depth/ June 12, 2017. (accessed on July 14, 2019)
25. P. Louridas, D. Spinellis, V. Vlachos. Power laws in software. ACM Transactions on Software Engineering and Methodology, 18(1):1–26, September 2008. Article 2. (doi:10.1145/1391984.1391986)
26. H. Ward. How to Hire. https://medium.com/eshares-blog/how-to-hire-34f4ded5f176#.b04yy5bka January 1, 2016. (accessed on July 14, 2019)
27. A. L. Barabási, R. Albert. Emergence of Scaling in Random Networks. Science 286, 509 (1999). DOI: 10.1126/science.286.5439.509
28. N. P. Suh. The Principles of Design. 1st ed. New York: Oxford University Press; 1990
29. N. P. Suh. Axiomatic Design--Advances and Applications. Oxford University Press. 2001.
30. Wiktionary. Give a man a fish and you feed him for a day. https://en.wiktionary.org/wiki/give_a_man_a_fish_and_you_feed_him_for_a_day;_teach_a_man_to_fish_and_you_feed_him_for_a_lifetime (accessed on July 14, 2019)
31. T. Anfodillo, M. Carrer, F. Simini, I. Popa, J. R. Banavar, A. Maritan. An allometry-based approach for understanding forest structure, predicting tree-size distribution and assessing the degree of disturbance. Royal Society Publishing. 22 January 2013. https://doi.org/10.1098/rspb.2012.2375
32. K. J. Niklas. Plant Allometry: The Scaling of Form and Process. University of Chicago Press. 1994
33. C. Z. Loboz. Cloud Resource Usage—Heavy Tailed Distributions Invalidating Traditional Capacity Planning Models. Journal of Grid Computing. December 2010. DOI: 10.1145/1996109.1996112
34. Wikipedia. Markov property. https://en.wikipedia.org/wiki/Markov_property (accessed on July 14, 2019)
35. J. Thomas. Archstand Theory of Design for Innovation. PhD Thesis, 1995. Available at: http://dspace.mit.edu/handle/1721.1/11722 MIT (accessed on July 14, 2019)
36. Wikipedia. Stent. https://en.wikipedia.org/wiki/Stent (accessed on July 14, 2019)
37. Wikipedia. Ironic process theory. https://en.wikipedia.org/wiki/Ironic_process_theory (accessed on July 14, 2019)
38. Wikipedia. SWOT analysis. https://en.wikipedia.org/wiki/SWOT_analysis (accessed on July 14, 2019)
39. Wikipedia. Porter's five forces analysis. https://en.wikipedia.org/wiki/Porter%27s_five_forces_analysis (accessed on July 14, 2019)
40. W. Jansen, T. Grance. Guidelines on Security and Privacy in Public Cloud Computing. National Institute of Standards and Technology. Special Publication 800-144. December 2011. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-144.pdf (accessed on July 14, 2019)
41. The Open Group. Cloud Computing Governance Framework. http://www.opengroup.org/cloud/gov_snapshot/p3.htm (accessed on July 14, 2019)
42. E. Passmore. Migrating Large-Scale Services to the Cloud: A master checklist of everything you need to know to move to the Cloud. Apress. 2016.
43. Cloud Industry Forum. 8 Criteria to ensure you select the right cloud service provider. https://www.cloudindustryforum.org/content/8-criteria-ensure-you-select-right-cloud-service-provider (accessed on July 14, 2019)
44. D. J. Snowden, M. E. Boone. A Leader's Framework for Decision Making. Harvard Business Review. November 2007. https://hbr.org/2007/11/a-leaders-framework-

for-decision-making (accessed on July 14, 2019)

45. G. Wong. Jumping the S-curve. September 29, 2011. https://cognitive-edge.com/blog/jumping-the-s-curve/ (accessed on July 14, 2019)
46. J. Thomas, P. Mantri. Axiomatic Design/Design Patterns Mashup: Part 1. In: 9th International Conference on Axiomatic Design (ICAD), Procedia CIRP, 2015, 34, 269-275. Available at: http://www.sciencedirect.com/science/article/pii/S2212827115008501
47. J. Thomas, P. Mantri. Axiomatic Design/Design Patterns Mashup: Part 2. In: 9th International Conference on Axiomatic Design (ICAD), Procedia CIRP, 2015, 34, 276-283. Available at: https://www.sciencedirect.com/science/article/pii/S2212827115008513
48. Wikipedia. Democritus. https://en.wikipedia.org/wiki/Democritus (accessed on July 14, 2019)
49. Wikipedia. Aristotle. https://en.wikipedia.org/wiki/Aristotle (accessed on July 14, 2019)
50. J. Sonmez. Soft Skills: The software developer's life manual. Manning Publications. 2014.
51. N. P. Suh. Complexity: Theory and Applications. 1st ed. New York: Oxford University Press; 2005
52. R. Coram. Boyd: The Fighter Pilot Who Changed the Art of War. Little, Brown and Company. 2002
53. Wikipedia. OODA loop. https://en.wikipedia.org/wiki/OODA_loop (accessed on July 14, 2019)
54. C. Richards. Certain to Win: The Strategy of John Boyd, Applied to Business. Xlibris. 2004
55. D. Newman-Toker. Diagnostic Errors More Common, Costly And Harmful Than Treatment Mistakes. April 23, 2013. https://www.hopkinsmedicine.org/news/media/releases/diagnostic_errors_more_common_costly_and_harmful_than_treatment_mistakes (accessed on July 14, 2019)
56. J. Weinman. The 10 Laws of Cloudonomics. Sep 7, 2008. https://gigaom.com/2008/09/07/the-10-laws-of-cloudonomics/ (accessed on July 14, 2019)
57. DevOps@Logicworks. ArchOps vs. DevOps: Foundation and Automation. https://www.logicworks.com/blog/2015/03/aws-devops-archops-automation/ (accessed on July 14, 2019)
58. G. Hohpe. The Architect Elevator — Visiting the upper floors. https://martinfowler.com/articles/architect-elevator.html#ArchopsBuildAVerticalArchitectureTeam (accessed on July 14, 2019)
59. Wikipedia. Matthew 6:3. https://en.wikipedia.org/wiki/Matthew_6:3 (accessed on July 14, 2019)
60. D. Snowden. Safe-fail or Fail-safe. https://cognitive-edge.com/blog/safe-fail-or-fail-safe/ September 2, 2006. (accessed on July 14, 2019)
61. J. Kern. Agile in the Context of a Holistic Approach. https://www.infoq.com/articles/agile-holistic-approach/ October 01, 2018 (accessed on July 14, 2019)
62. B. Meyer. Agile!: The Good, the Hype and the Ugly. Springer. 2014
63. G. S. Winters. Why Agile is Failing at Large Companies. Ty yn Goch Forrest Publications. 2016
64. T. J. Brizard. Broken Agile. Apress. 2015.
65. E. Puik, D. Ceglarek. Application of Axiomatic Design for Agile Product Development. MATEC Web of Conferences 223, 01004 (2018). ICAD 2018 https://doi.org/10.1051/matecconf/201822301004
66. M. K. Thompson. Where is the 'Why' in Axiomatic Design? ICAD2014. The Eighth International Conference on Axiomatic Design. Campus de Caparica – September 24-26, 2014
67. K. M. Adams. Non-functional Requirements in Systems Analysis and Design. Springer. 2016.
68. P. Sroczkowski. Cloud: IaaS vs PaaS vs SaaS vs DaaS vs FaaS vs DBaaS. https://brainhub.eu/blog/cloud-architecture-saas-faas-xaas/. (accessed on July 14, 2019).
69. N. Levy. Amazon and Capital One face legal backlash after massive hack affects 106M customers. August 9, 2019. https://www.geekwire.com/2019/amazon-capital-one-face-lawsuits-massive-hack-affects-106m-customers/
70. C. Morris. A Goldmine for Identity Thieves: Unprotected Database Puts 65% of American Households at Risk. April 29, 2019. http://fortune.com/2019/04/29/security-gap-personal-information-breach/
71. D. O'Sullivan. Cloud Leak: How A Verizon Partner Exposed Millions of Customer Accounts. December 12, 2018. https://www.upguard.com/breaches/verizon-cloud-leak
72. B. Roussey. 10 biggest 2018 data breaches — and what they mean for 2019. March 22, 2019. http://techgenix.com/2018-data-breaches/
73. H. Schulze. Cloud Security Report-2018. https://pages.cloudpassage.com/rs/857-FXQ-

213/images/2018-Cloud-Security-Report%20%281%29.pdf

74. R. Ross, M. McEvilley, J. C. Oren. Systems Security Engineering Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems. NIST Special Publication 800-160. November 2016. https://doi.org/10.6028/NIST.SP.800-160
75. N. Statt. Apple's cloud business is hugely dependent on Amazon. Apr 22, 2019. https://www.theverge.com/2019/4/22/18511148/apple-icloud-cloud-services-amazon-aws-30-million-per-month (accessed on July 14, 2019)
76. J. Weinman. Cloudonomics: The Business Value of Cloud Computing. Wiley. 2012.
77. G. Roumeliotis. U.S. blocks MoneyGram sale to China's Ant Financial on national security concerns. https://www.reuters.com/article/us-moneygram-intl-m-a-ant-financial/u-s-blocks-moneygram-sale-to-chinas-ant-financial-on-national-security-concerns-idUSKBN1ER1R7 January 2, 2018. (accessed on July 14, 2019)
78. K. E. Weick, J. D. Orton. Loosely Coupled Systems: A Reconceptualization. Loosely Coupled Systems: A Reconceptualization. Academy of Management Renew, 1990, Vol. 15, No.2, 203-223
79. K. E. Weick. Educational Organizations as Loosely Coupled Systems. Administrative Science Quarterly. Vol. 21, No. 1 (Mar., 1976), pp. 1-19. Published by: Sage Publications, Inc.
80. K. E. Weick. Making Sense of the Organization, Oxford, England: Blackwell Publishers. (2001).
81. J. Thomas, P. Mantri. Complex Adaptive Blockchain Governance. MATEC Web of Conferences 223, 01010 (2018) https://doi.org/10.1051/matecconf/201822301010