Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 34 (2015) 269 – 275

www.elsevier.com/locate/procedia

9th International Conference on Axiomatic Design – ICAD 2015

Axiomatic Design/Design Patterns Mashup: Part 1 (Theory)

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

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

* Corresponding author. *E-mail address:* johntom@cogtools.com

Abstract

Traditionally, the Axiomatic Design (AD) Framework aids the top-down design in at least three fundamental ways. Firstly, it helps capture the requisite, high-level systemic-property of a given design. Secondly, given an ensemble of candidate designs, it helps evaluate the designs. Finally, when viewed from the Axiomatic Design/Complexity Theory (AD/CT) framework, it helps contain and manage the ensuing design complexities. Likewise, from the other end of the spectrum, diverse industries have been collecting and cataloging successful Design Patterns (ĐP) in a bottom-up sense. A Design Pattern formally captures the solution to a recurrent design problem in a given field. There is now enough critical mass from both these streams-of-thought to effectively engage in an intellectual mash-up. This paper helps motivate, plan and initialize such a deliberate mash-up. Part 1 surveys both approaches from a knowledge architecture perspective to help frame the top-down approach as the V-model; the bottom-up as the Λ-model; and the integrated as the N-model. It then addresses the central economic issue of our age, namely the proper division of labor between human and machine and locates it in the context of design. Part 2 illustrates the approach using Cybersecurity Patterns.

© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Peer-review under responsibility of the organizing committee of 9th International Conference on Axiomatic Design

Keywords: Axiomatic Design; Design Patterns; Knowledge Architecture Dynamics; Cynefin; Human-Machine Symbiosis

1. Introduction

Mashup's are common in the domains of pop-culture, business, and the internet. Thus we have business-line mashups, music/video-mashups, web-application mashups etc. But mashup's across intellectual streams is a rare occurrence on account of the tower-of-babel syndrome. Also, greater the intellectual distance between two candidate domains, greater is the task to bridge this gap. Thus the gap is all the more significant when considering two antipodal approaches to design such as the (predominantly top-down) Axiomatic Design (AD) approach as spearheaded under the leadership of Prof. Nam Suh and the (predominantly bottom-up) Design-Patterns (ĐP) approach as spearheaded under the leadership of architect Christopher Alexander. But given the richness of these two diametrically opposite approaches, it is well worth the trouble to attempt such a potentially climatic mash-up.

2. Top-Down Axiomatic Design (AD)

What establishes Axiomatic Design (AD) as a top-down framework is the formulation [1]: "*the first step toward developing a solution is the synthesis of the overall solution.*" Here, the two design-axioms provide analytical structuring of the creative space [1]: "*What the design axioms do is to complement and aid the creative process by providing the analytical tool for evaluation of the synthesized ideas so as to enable the selection of only good ideas.*"

And at every stage of the analytical process, the design is decomposed a) laterally and non-hierarchically across *realms* such as customer, functional, physical, process (CR, FR, DP, PV) etc., and b) vertically and hierarchically within each of the above realms. Software Development may also include implementation, testing and integration. Lateral decomposition is sequential, with the functional being the dominant realm; the CRs get formalized and subsumed under the FRs. Integrity of the design is maintained across the realms in a zigzag cross-domain mapping of the nodes at each level of the hierarchy. The Axiomatic Design framework is fundamentally a hierarchical, top-down approach as opposed to a bottom-up approach. The question one may ask here is why is that the design process and the artifacts it produces along the way exhibits a hierarchical pattern in general? Design being a human endeavor, clearly there is a shared underlying epistemological factor driving this pattern. We take this up in Section 4. But now consider the complexity extension to AD.

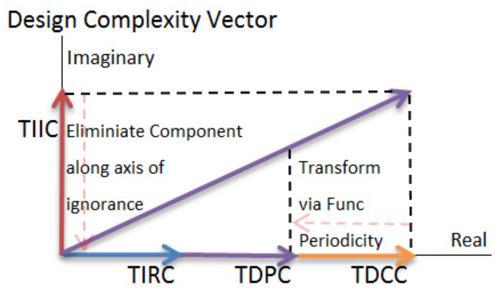
2212-8271 © 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Peer-review under responsibility of the organizing committee of 9th International Conference on Axiomatic Design
doi:10.1016/j.procir.2015.08.010

(a)

Axiomatic Design	Cynefin Model
Time Independent (TI)	Ordered
Real Complexity (TIRC): Match System vs. Design Range Overlap; Decrease bias & FR variance; Insignificant time dependence in either case.	**Simple**: Known Knowns; **Whole = ∑(parts)**; Catalogue & Use Best Practices; *Sense-> Categorize-> Respond;* If operated outside the system range, the design could fall over the edge into chaos.
Imaginary Complexity (TIIC): Correct inadequacies in domain/axiomatics knowledge base. Avoid coupling. Minimize components along the axis-of-ignorance.	**Complicated**: Known Unknowns; **Whole = ∑(parts)**; Catalogue & Use Good Practices; *Sense-> Analyse-> Respond;*
Time Dependent (TD)	Unordered
Periodic Complexity (TDPC): Contain time-dependent real-complexity via naturally ocurring periodic system-resets	**Complex**: Unknown Unknowns; **Whole > ∑(parts)**; Probe & induce pattern-based emergent order; *Probe-> Sense/Analyse-> Respond*
Combinatorial Complexity (TDCC): Transform TDCC to TDPC by introducing functional periodicity to limit complexity growth via forced periodic resets	**Chaotic**: Unknowable; **Whole > ∑(parts)**; Rapidly deteriorating scenario; Use pattern-based heuristics to transform problem to either ordered or the unordered complex; Act-> Sense/Analyse-> Respond

(b): Axiomatic Design/Complexity Theory (AD/CT)



(c) Cynefin

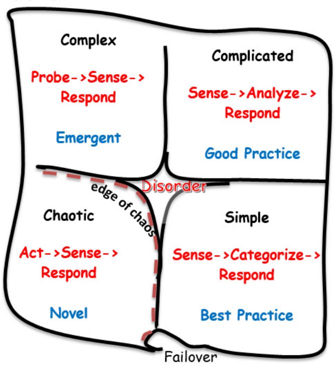


Fig. 1. Complexity Theory: (a) Comparison; (b) AD/CT; (c) Cynefin

3. Complexity-Theory: AD/CT vs. Cynefin

Prof. Suh extended the axiomatic approach (Axiomatic-Design Based Complexity Theory: AD/CT) to cover the complexities of design [3]. There is a close match between the four-squares of AD/CT and David Snowden's Cynefin framework [4] as shown in Fig. 1. What Cynefin calls as Ordered, AD/CT calls as Time-Independent; what Cynefin calls as Unordered, AD/CT calls as Time-Dependent. And as AD/CT clearly indicates, the lack of temporal periodicity is just one type of disorder; others include geometric, chemical, informational, biological, etc. Also AD/CT establishes the fundamental concept of functional periodicity which allows for the zigzag order/disorder tracking/gap-closing across the FR: DP divide. The advantage of AD/CT is that it provides rich quantitative/qualitative tools for structuring/transforming the design complexities; the advantage of Cynefin is that it integrates well with management practices. Also, Cynefin is more sensitive to the role of patterns, emergence and inductive thinking in the context of complex/chaotic domains. We know that biological and social systems have incorporated increasing levels of complexity across time in the struggle for life. Viewed from the perspective of combinatorial complexity, most of the combinations only contribute to increasing disorder. But amongst these are rare mutations and combinations that exhibit emergent properties that help the system scale beyond the current status-quo. It is in this sense that across the immense expanse of time, biological evolution has repeatedly hit the jackpot of emergent properties to produce the richness of life we see today. So the pursuit of complexity therefore does bestow benefits if properly understood. From a design-perspective, the challenge is to use top-down frameworks such as AD/CT to reduce the bottom-up, hit-and-trial implicit in the traversal of such immense combinatorial spaces (as in say, the Nano-world) and deliberately design-for-emergence. Note that contrary to [3], biological systems thrive and have learned to put to good use chaotic, time-dependent combinatorial complexity. These systems have thus put the genetic machinery to good use in searching, finding & propagating winning emergent properties across the combinatoric expanse.

4. Dynamics of Knowledge-Hierarchy

Hierarchies (whether top-down or bottom-up) in design arise from Francis Bacon's governing insight about human artefacts that "*nature to be commanded must be obeyed.*" Human knowledge captures the sum-total of truths/facts gleaned from nature and accumulated across time. Given the relative abundance of concretes to abstractions, human knowledge has a hierarchical shape in its object categories (Fig 2c). The count of perceptuals at the base is wider while abstractions are few. Note that knowledge is also relational in the sense that object categories are networked in the same manner that two friends are networked across the social web; i.e., it is non-hierarchical. And these relationships play a key role in the zig-zag, cross-realm design-matrix mappings.

When a domain (say systems biology) is mapped, there is an overall conical shape to the conceptual network. Knowledge may therefore be seen as a self-similar, inducto-deductive fractal, both in the general as well as in the specifics. Induction is the upward flowing arch involved in creating higher level generalizations. Deduction is the downward flowing arch involved in the application of these generalizations. And since human knowledge is hierarchical, the design trace that leverages this knowledge is likewise hierarchical.

When aligned and composed along domain kinship metrics, one may expect the myriad knowledge conics to exhibit a self-similar fish-scale fractal structure as shown in Fig. 2d. Tools from the burgeoning field of scientometrics [11] may be utilized to map and causally-link the current fragmentary and deeply specialized knowledge architectures. Also since these tools help view and navigate human knowledge as an integrated whole, and since the act of design is fundamentally an act of synthesis, these tools could aid the designer faced with modern complexities. This is especially so given the fragmentary nature of human knowledge today, specifically at the interstices.

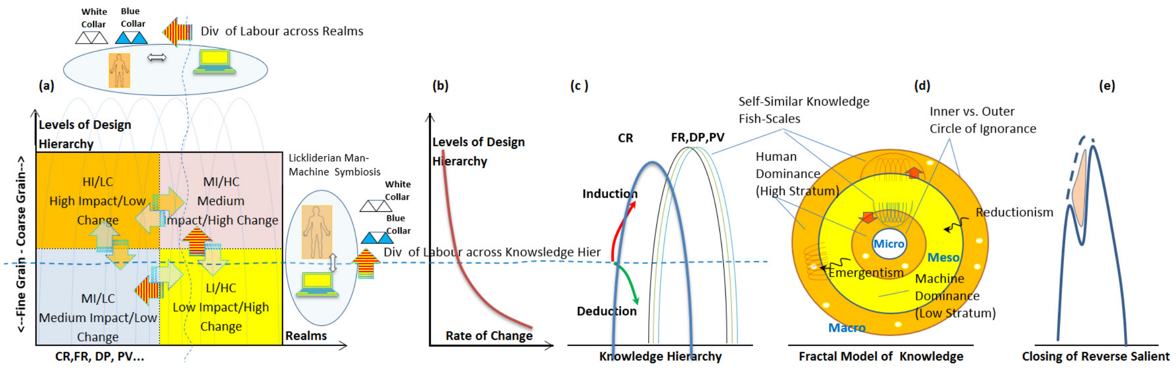


Fig. 2. Knowledge Dynamics: (a) {Realms x Levels} Grid; (b) Rate-of-Change; (c) Knowledge-Hierarchy; (d) Fractal Model; (e) Reverse-Salients

The human scale is the meso-level. Unknowns from the macro-world dominate the outer regions; unknowns from the micro-world dominate the inner regions. Human knowledge is sandwiched between these two circles-of-ignorance.

Reductionism, i.e., whole is the sum of its parts, attempts to reduce all existents to the minimal set at the center. Emergentism, i.e., whole is more than the sum of its parts, attempts to capture systemic properties that evaded the reductionist mapping. Each provides structure and balance to the other. The emergent property, when placed within the context of the vast reductionist knowledge-architectures, turn out to be the abstract formulation of a rare combinatoric that exhibits order in contrast with the rest. These emergent abstractions are rendered as scattered white dots in Fig. 2d.

Now consider the impact of knowledge dynamics. Given the rate of change in the knowledge corpus [2], the dynamics evident in human knowledge architectures can profoundly impact the design-trace as well as the resulting design complexities. Reverse-Salients (Fig. 2e) are lagging knowledge fronts that occur on account of differentials in the rate of knowledge-growth in closely aligned fields, thus leaving gaps that need to be closed. The eventual closing of a given reverse-salient propagates across the knowledge fabric, both causally as well as analogically. Causal propagation occurs in wave-like patterns both radially as well as tangentially. Analogical propagation occurs by seeding the re-evaluation of basic premises in remote fields. Closing of reverse salients at the boundaries (i.e., the contraction of the inner-circle as well as the expansion of the outer-circle) have system-wide implications we associate with paradigm-shifts. An example from the 1860s is described in [12] regarding the age of the earth's crust. Only with the discovery of nuclear fusion in the 1930s was the reverse salient between geologists and physicists resolved in favor of the former. To recap, the reverse salient was first established at the macro level. It was then erroneously rejected by physicists at the micro level. This error was eventually corrected at the micro level with the discovery of nuclear fusion. And when this correction traversed back to the macro, it revised the 100 million estimate into the billions. So, as the inner circle crunches down, the outer circle expands; and vice-versa. While knowledge architecture dynamics is occurring at various levels, the above duetting-play between the inner and the outer is perhaps the broadest, self-correcting, inducto-deductive dynamic pattern. With this background, consider now the Design Patterns approach.

5. Bottom-Up Design Patterns (DP)

A Design Pattern formally captures the solution to a recurrent design problem. The key term to note here is that of recurrence, indicating that problems as well as solution-patterns repeat within and across domains, realms and hierarchical levels. Literature does not provide a convincing answer as to why this is so? Chris Alexander who initiated the Design Patterns approach in the realm of building architecture held that patterns emerge out of human discourse [13]: "*All acts of building are governed by a pattern language of some sort, and the patterns in the world are there, entirely because they are created by the pattern languages which people use.*"

Chris locates the pattern-origins in the machinery of the language that people use. He then decries that the languages have failed us [13]: "*But in our time the languages have broken down. Since they are no longer shared, the processes which keep them deep have broken down; and it is therefore virtually impossible for anybody, in our time, to make a building live.*"

Chris then embarks on re-constituting the broken language towards a pattern language for the modern times. But this begs the question as to why languages have broken down in the first place? What is the underlying process that keeps a language wholesome; and vice versa, what is it that renders it fragmentary? The answer is rooted in philosophy and has to do with the problem-of-induction. As human knowledge grows, it is the induction of higher level concepts that establishes order within the growing corpus. Philosophy has been adrift ever since mid-18th century when David Hume denied the role of induction. David Harriman [12] corrects this error by locating the validity of induction at the systemic total knowledge level. This may also be witnessed in the knowledge dynamic between the aforementioned two circles of ignorance. As the inner circle crunches down, the outer circle expands; and vice-versa. Here the validity of induction is being asserted primarily in an asymptotic, self-correcting sense, with the overall impetus being towards holistic coverage. And by grounding pattern-discovery in the human act of induction, one may avoid the mistake of reification of language as the source, and in favor of the laws of nature as the true source of patterns.

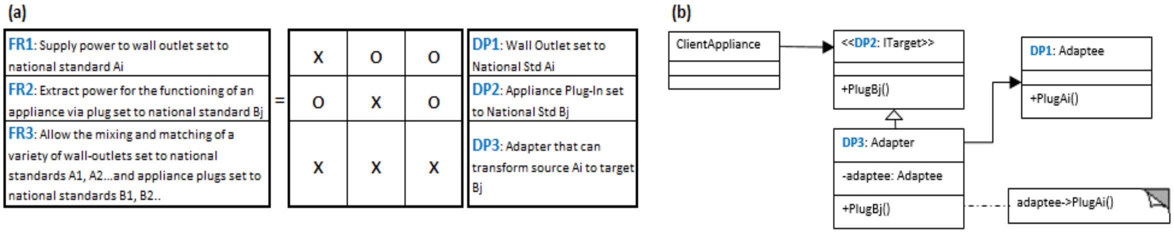


Fig. 3. Object Adapter: (a) Axiomatic Design Approach; (b) Design Patterns Approach

To get a sense of how to map design-patterns to AD, consider the simple example of the Object Adapter Pattern as first portrayed in [14]. Part 2 covers an extensive example from the domain of Cybersecurity. Adapters are ubiquitous across the technological landscape where parts that conform to different national/technological standards nevertheless needs to fit. Adapters help bridge the technological gap. Fig. 3a is the AD version, while Fig. 3b is the Object-Oriented UML/ÐP version.

The AD version captures the FR↔DP mappings both at the high-level as well as for further decompositions such as would be required for FR3↔DP3. Also, the couplings are explicit in the design-matrix, whereas they are implicit in the UML diagram. This allows for the axiomatic-design machinery to be put to good use in the analysis of complex designs. But most critically, it provides a common framework for the composition/decomposition of broader patterns.

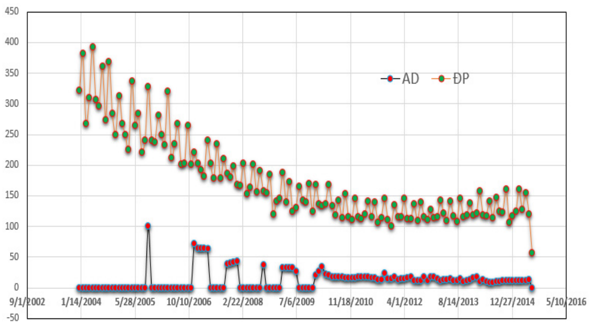


Fig. 4. Google Trends: Axiomatic Design (AD) vs Design Patterns (ÐP)

6. Rationale for AD/CT-ÐP mashup

The modern practice of design-patterns has hit a few stumbling blocks [15]:

- There is no authoritative patterns-index. Indexing on FRs instead of DPs could help resolve this problem
- As the modern world becomes increasingly complex, the ÐP approach finds it difficult to establish broader patterns. Also, as Cynefin shows, using simplistic solutions for complex problems often leads to chaos. Using the AD framework could help in the composition/ decomposition of broader patterns.
- Absent a shared, higher-level patterns vocabulary, design quickly devolves into low stratum [9] disagreements. The greatest value-add from patterns is not at the low level; instead it is in enriching the shared understanding at the higher conceptual level. The discipline of AD can help defer the inevitable gravitational pull of implementation pragmatics sufficiently long to erect the requisite vocabulary.

Google-trends (Fig. 4) captures the waning interest in ÐP over the years. Without an over-arching theoretical framework to help bring order amongst the proliferating design patterns across various domains, the designer is lost in the proliferating ÐP search space. Also, it is difficult to abstract and leverage the patterns from one domain into another. An AD/CT-ÐP mash-up could help structure the ÐP space.

Fig. 4 also captures the relative position of AD/CT vis-à-vis ÐP. While rich in its theoretical/practical offerings, it takes time and effort for any top-down framework to scale. Design being a practical discipline has to make sense in the domain language. Establishing a common design vocabulary, both at the abstract as well as at the fine-detail level is a time consuming affair. AD/CT could thus benefit from the rich inductive base that ÐP provides. An AD/CT-ÐP mash-up could therefore help AD/CT in enriching its base and achieving scale. Furthermore, as illustrated in the above example, these two approaches are not fundamentally antagonistic to each other, given their antipodal origins. Where ÐP is silent (i.e., design theory), AD/CT is rich; where AD/CT has yet to establish the requisite domain-specific vocabulary, ÐP is rich. The top-down approach therefore complements the bottom-up approach and vice-versa. The next section considers the top-down/bottom-up design-process in the domain of software.

7. Top-Down (V) vs. Bottom-Up (Λ) vs. Hybrid (N)

Fig. 5 depicts various top-down/bottom-up software development models. The Waterfall-Model portrays sequential, step-by-step, single-step-at-a-time approach with clearly defined hand-offs and limited feedback between the various steps. Design and development proceed predominantly in a linear fashion from requirements, to high-level-design, to detailed design (with validation and technical testing attached to each step). Each hand-off is a distinct milestone to help clients and project managers track the project. But the client has to wait until the final stage to test-drive the product. Constraints and requirements are assumed to be completely identified and signed-off upfront. Any downstream scope-creep is strongly discouraged via concomitant cost and schedule overruns.

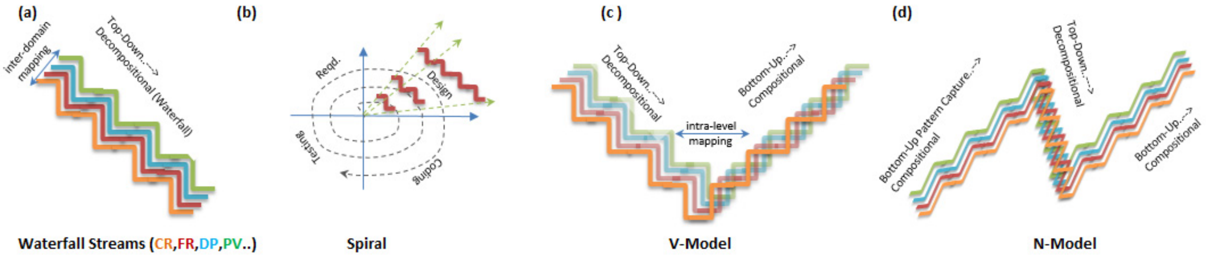


Fig. 5. Top-Down/Bottom-up Models: (a) Waterfall; (b) Spiral; (c) V-Model; (d) N-Model

While the structure and predictability available in the Waterfall model helps handle team-dynamics well (i.e., key-man risk is mitigated), this same rigid structure becomes a liability in the face of relentless change-requests. Framing it in the Axiomatic Design Based Complexity Theory (AD/CT) [3], the Waterfall model is unable to handle the increasing Time-Dependent Complexities (or more generally, the unordered complexities of Cynefin). Also, Waterfall is a documentation-heavy approach that faces scaling issues on account of the resultant information-overload. The demands for documentation increase with the increasing project complexity and concomitant team-size increases; communication pathways scale as $O(n^2)$ with team size n. An integrated AD/CT-ĐP approach could dramatically shift this balance by helping establish a common lingua-franca for design across various domains and realms. A modification is to use a set of smaller, time-shifted water-falls that incrementally and iteratively delivers software in multiple integrated releases. The problem here is that of establishing the requisite "divide-and-conquer" increments with minimal couplings. The Spiral-Model is likewise iterative; but it revisits the whole-as-a-whole while adding finer resolutions towards the actualization of the design. Constant inter-coil radial distances map to constant resources, even as the scope grows across the iterations.

The generic V-Model as first proposed by Paul E. Rook [5] in the context of software Quality Assurance (V/QA) is a modification of the top-down waterfall model. The downward descending arm is top-down and decompositional in nature while the upward ascending arm is bottom-up and compositional. Rook's ascending arm is for testing and validation and includes (in ascending order) unit, integration and acceptance testing—with intra-level mapping across the two arms. In the context of Systems Biology [6], Prof. Suh et al. adopt the V-Model to reverse-engineer the FR↔DP mappings of biological systems in a bottom-up/top-down sense. And in the context of software design [7], Prof. Suh et al. adopt the V-Model to frame the Axiomatic Design of Object-Oriented Software Systems (ADo-oSS) methodology. Here the downward trace is the FR↔DP decomposition, while the upward trace is the bottom-up composition into the requisite class hierarchies with the appropriate interfaces. Note that only the AD based V-models engage in the lateral decomposition between FRs and DPs that could help avoid coupled designs colloquially referred to as the hairball.

It is primarily the inflexibility and rigidity in the face of incessant change-requests that businesses have had to field that has triggered a shift away from all Waterfall models (including all downward-dominant V-variants) and towards Agile.

In direct opposition to the Waterfall model, the Agile Manifesto [8] may be summarized as follows:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

To clarify the "over" clause, Agile does not mean no planning, design, documentation, processes, tools etc. Rather it is a shift in emphasis away from the later and towards the former.

In contrast to the Waterfall model, Agile is an iterative, incremental and flexible approach that can favorably handle incessant change requests. The project starts off with a deliberately simple throw-away design and iteratively adds in the needed complexity over time. Developers work in small close-knit, cross-functional teams on bite-size modules farmed across multiple project sprints. In its iterative nature, Agile is similar to Spiral, except that it is piece-meal and does not iterate on the whole-as-a-whole at any stage of the process.

As highlighted in the manifesto, a working piece of software is preferred in comparison to excessive documentation. Client-feedback & testing occurs in tandem/parallel sprints with minimal lag, often tracking a mini-inverted-V (or Λ) model with a bottom-up, hit-and-trial, ascending leg followed by a more deliberate descending leg. In due course, the requirements stabilize asymptotically as the bite-sized module implementation is iteratively tested and critiqued by the client.

While superior to the Waterfall-model with regard to scope rigidity vs. flexibility, all is not well on the Agile-front. The fundamental problems with Agile is that it does not scale. As the last directive in the Agile Manifesto (i.e., responding to change over following a plan) makes clear, Agile is a tactical, bottom-up, Λ-approach that would face scale-up limitations. While it purports to deal with the incessant change, it does poorly against combinatorial complexity.

In the context of ĐP, Chris Alexander did recognize the holistic nature of the pattern-language, its bottom-up built-up, and its top-down usage [13]. The bottom-up pattern-capture is followed by the top-down, decompositional acts of design which is then followed by compositional acts of design integration. This design-process pattern is depicted as the N-Model in Fig. 5d which is a variant of the V–model. The N-Model is a hybrid inducto-deductive approach that dovetails well with how knowledge itself is created and used. It therefore provides an appropriate meta-level scaffold for attempting an AD/CT-ĐP mashup. When formalized, it may be made explicit along the following 4 steps, with further details in Part 2:

- 1st Leg of /\/: Review the ĐP and explicate the embedded conceptual hierarchy.
- 2nd Leg of /\/**:** Top-Down decomposition & Axiomatic framing of the ĐP; disambiguation of the FR-DP mappings.
- 3rd Leg of /\/: Bottom-Up integrations, including those at the sub-system/system levels.
- Cross-Domain/Realm/Levels/N Integrations: This is where the AD/CT-ĐP mash-up may show unexpected positive results.

Given the impact of change in the context of design, the next section frames it in the larger knowledge architecture and dynamics context we had touched upon earlier.

8. Division-of-Labor Between Human and the Machine

When viewed from the framework of Axiomatic Design Based Complexity Theory (AD/CT) [3], the modern shift towards agility may be viewed as an weak but market-driven attempt at controlling both Time-Independent (real/imaginary) as well as Time-Dependent complexities (predictable-periodic/combinatorial). These complexities can occur across all the lateral realms (customer, functional, physical, process etc.) as well as across all the hierarchical levels in each of the respective realms. The rate-of-change is not homogeneous across the two-dimensional {realm x level} design grid (Fig. 2a). It is most frequent downstream across the lateral domains and at the lower rungs of each of the hierarchies (Fig. 2b). This is because design options and their concomitant change-requests proliferate in the said order. Also, the impact-of-change is not homogeneous across the design grid. It is most pronounced upstream across the lateral realms and at the higher rungs of each of the hierarchies. The above 2-by-2 grid is qualitative and meant to serve as a scaffold for discussing regions of comparative dominance.

Fine-grained uncoupled/decoupled designs with high levels of change-dynamics and low-impact (LI/HC) may naturally fit the Agile development model. And in direct contrast, coarse-grained designs (of any kind) with high-impact and low change dynamics (HI/LC) may naturally benefit from the deliberation and close-coordination of a top-down Waterfall model. Even so, it is more likely to find deeply coupled designs at the fine grain, and fairly uncoupled/decoupled designs at the coarse grain. Why is this so? At the fine grain, as the hierarchical levels increase, the combinatorial design space becomes exponentially all the more unwieldy. Many of the change requests may very well be attempts to recover from earlier coupled-design mishaps at these levels. Without the benefit of the top-down AD-Framework, blindly adopting a hit-and-trial approach would run into O(n!) possibilities, of which only a few patterns might actually work (uncoupled/decoupled). So while Agile works best in an uncoupled/decoupled problem context established at a higher level, chances are that it is handed a deeply coupled problem context to begin with. And if that is not enough, Agilists naturally prefer a bottom-up, Λ-based, hit-and-trial approach as they navigate the design space. Except for AD/CT, currently there does not seem to exist a single, coherent, integrated approach that works across all the levels and change-dynamics. To complete the picture, off-diagonals (Fig. 2a) such as Medium-Impact/Low-Change (MI/LC) may naturally align with a top-down approach; likewise, Medium-Impact/High-Change (MI/HC) may align with the bottom-up approaches.

Over time, the internal borders may shift (as indicated by the internal arrows) to encompass various struggles such as the battle for methodological dominance, changing patterns of dynamism/impact, as well as the changing social/economic/ technological context. But ever since the Moore's law set the modern pace, the dominant trend has been towards ever deepening levels of technological offerings to encompass more and more of the fine grain such as the current challenge of designing at the nano-scale level, while at the same time pushing the human towards higher stratum [9] roles. This shift in the competitive landscape is indicated by the striped red-arrows. So while the internal methodological wars between the various schools of Agilists vs. the Waterfall camps continue unabated, the larger battle will most likely be shaped by those who learn to master the combinatorically exploding vistas of the fine-grained design space and integrate it with the coarse-grain while minimizing coupling. Given the combinatorial complexity at the fine grain, combining AD/CT with the Lickliderian ideal of Man-Machine Symbiosis could prove valuable in navigating and designing both at the coarse-grain as well as (increasingly) at the fine-grain [10]:

> *Man-computer symbiosis…will involve very close coupling between the human and the electronic members of the partnership. The main aims are 1) to let computers facilitate formulative thinking as they now facilitate the solution of formulated problems, and 2) to enable men and computers to cooperate in making decisions and controlling complex situations without inflexible dependence on predetermined programs.*

While the Lickliderian symbiosis can occur in any of the above four-squares, its unparalleled value-add will be in navigating the ever expanding, combinatorically complex design-grid envelope at the fine grain. While everyone could gain, Agilists have far more to gain and/or lose by embracing/not-embracing this potential, given the natural overlap. The Lickliderian symbiosis has been depicted iconically at the side of the diagram with the dotted line indicating the respective relative dominance.

In his visionary essay, Licklider sees the division-of-labor between man & machine as between the higher-level vs the lower-level. The higher-level functions would fall on the orange sections of the above diagram and would include establishing the problem motivation (i.e., the framing of the problem), the overall sense-making, course-correction and supervisory guidance of the design-search, as well as handling low-probability, black-swan events [10]: "*In addition, men will handle the very-low-probability situations when such situations do actually arise....*"

In contrast, Licklider saw the machine execute all the routinizable, lower-stratum operations; yet what is low-stratum vs what is high-stratum is a shifting front. Also, the machine has clear advantages in the context of memory and processing power. For example, Licklider saw the speed of decision making as one of the key demarcations between the human and the machine. Wherever time is of essence (e.g. high-frequency trading), the machines have a strategic advantage. There is also the added reason of the coarse-grain deliberations trying to match the finer-grain pace. Otherwise the impedance mismatch will result in the slowing of the fine-grained pace to match that of the slower coarser grain, which the markets would reject. Licklider's visionary essay predicted such a denouement for the tardy general on the modern battlefront [10]: "*Obviously, the battle would be over before the second step in its planning was begun.*"

As envisioned by Licklider, the advantage is clearly on the side of the machines and their few and short-lived human custodians. But for an overriding majority of human agents, the Lickliderian future looks rather bleak. To be clear, these are not mere Luddite fears. This is the emerging consensus from some of the leading thinkers and shapers of today such as Stephen Hawking who said: "*Artificial intelligence 'could spell the end of the human race.'*"

This concern is captured in Fig. 2a as the red-striped arrows with a question-mark, indicating the possible machine take-over. Both blue-collar as well as white-collar work-force is under pressure to scale upwards on the abstraction stratums; and leftwards towards the FRs. Mindless manual labor and clerical tasks are increasingly being automated. So where exactly (if at all) did the technocratic Lickliderian vision derail? Or are we perhaps misreading Licklider? Given the shifting-front between the human and the machine, it's not clear from Licklider's account as to where exactly human agents ought to be invested in from an educational/humanitarian perspective? Even so, Licklider does provide the following hint [10]: "*Poincare anticipated the frustration of an important group of would-be computer users when he said, "The question is not, 'What is the answer?' The question is, 'What is the question?'*"

Problem-posing as well as dealing with novel black-swan events (as mentioned earlier) are fundamentally inductive actions (Fig. 2c). From an epistemological perspective, the machine is fundamentally restricted to the deductive mode, while the human can reach beyond and induce new knowledge wherever the deductive front is found to be restricted. In contrast, the machine lacks the integrative, sense-making faculty to recognize its own limits and reach beyond. And if at a future date this limit is breached, we are then dealing with the emergence of a sentient being that is no longer a machine. Inductive-dominance vs. deductive-dominance therefore ought to be the fundamental divide between the human and the machine**.** Completing the Lickliderian vision, humans and machines would therefore interact, co-exist and flourish while being deeply entwined across the inducto-deductive knowledge fractal. It would therefore make sense (from an educational/ humanitarian perspective) for the human agents to be invested in the inductive sciences. Currently one may witness this trend in the increasing popularity of data-sciences, which is about pattern and sense-making when dealing with big-data. In this context it is critical to note that design as an act of synthesis is fundamentally inductive in nature. It is therefore to be expected that the problem of design will dominate the human condition.

9. Conclusion

As shown above, there is genuine potential for a mutually beneficial symbiosis between AD/CT & ĐP. We outlined the AD/CT-ĐP mapping with a simple example; but as Aristotle said, "one swallow does not make a summer". Part 2 illustrates the approach with an extended example drawn from Cybersecurity. Also, induction, with design as a dominant inductive pursuit is projected to define the human condition in the human-machine symbiotic future. In this context, a rich set of AD/CT-ĐP mappings could help sensitize us to our humanistic inductive advantages.

References

[1] Suh NP. The Principles of Design. 1st ed. New York: Oxford University Press; 1990
[2] Arbesman S. The Half-Life of Facts. 1st ed. New York: Penguin Group; 2012
[3] Suh NP. Complexity: Theory and Applications. 1st ed. New York: Oxford University Press; 2005
[4] Snowden DE, Boone ME. A Leaders Framework for Decision Making. Harvard Business Review. Nov 2007
[5] Rook PE. Controlling software projects. In: IEEE Software Engineering Journal, 1(1), 1986, pp. 7-16
[6] Jeffrey JD, Lee T, Suh NP. A Function-Based Framework For Understanding Biological Systems. Annu Rev Biophys Biomol Struct. 2004;33:75-93.
[7] Do SH, Suh NP. Object-Oriented Software Design with Axiomatic Design. In: First International Conference on Axiomatic Design Cambridge, MA – June 21-23, 2000
[8] http://agilemanifesto.org/
[9] Jaques E. Requisite Organization. 2nd Ed. Cason Hall & Co Pub; 2006
[10] Licklider, JR. Man-Computer Symbiosis. IRE Transactions on Human Factors in Electronics, HFE-1. 4-11. 1960.
[11] Chen, C. Mapping Scientific Frontiers. Springer. 2013
[12] Harriman D. The Logical Leap. New American Library. 2010.
[13] Alexander C. The Timeless Way of Building. Oxford University Press, USA. 1979
[14] Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. 1994
[15] Manolescu D, Kozaczynski W, Miller A, Hogg J. The Growing Divide in the Patterns World. Software, IEEE Vol.24, Issue:4 July-Aug. 2007