Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 34 (2015) 276 – 283

www.elsevier.com/locate/procedia

9th International Conference on Axiomatic Design – ICAD 2015

Axiomatic Design/Design Patterns Mashup: Part 2 (Cyber Security)

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

Part 1 established the theoretical framework for undertaking a top-down/bottom-up mash-up between the Axiomatic Design/Complexity Theory (AD/CT) and the Design Patterns (ĐP) framework. It synthesized an integrated N-model from the top-down V-Model and its antipodal/bottom-up Λ-model. Part 2 illustrates the approach in the domain of Cybersecurity. Recent trends in Cybersecurity indicates that our modern socio-technical systems are increasingly vulnerable to both external as well as internal threats. As the cost of defending our cyberstructures as well as the payoffs from successful attacks keeps rising, the cost of launching an attack simultaneously keeps decreasing. Cyber warfare is fundamentally asymmetric. Designing and managing modern socio-technical systems will demand greater appreciation for the language of threat-patterns and its defence. This paper applies the N-model to arrive at a generic pattern-language compositional technique using the Axiomatic Design Matrix. It then shows how one may splice the FR↔DP mappings with the FR↔ĐP mappings. Transcending the technical and into the socio, it is increasingly clear that wars of the future will be fought in the crucible of each individual human mind. Cognitive Biases are the patterns of weaknesses in this domain. The paper illustrates that patterns of cognitive biases can lead to patterns of cyber insecurity.

Keywords: Axiomatic Design; Design Patterns; Cyber Security; Pattern Language; OODA;

1. Introduction

Part 1 covered the theoretical underpinnings of the Axiomatic-Design/Design-Patterns (AD/CT-ĐP) mash-up along the N-model. The approach is formalized as a 4 stage process as shown below:

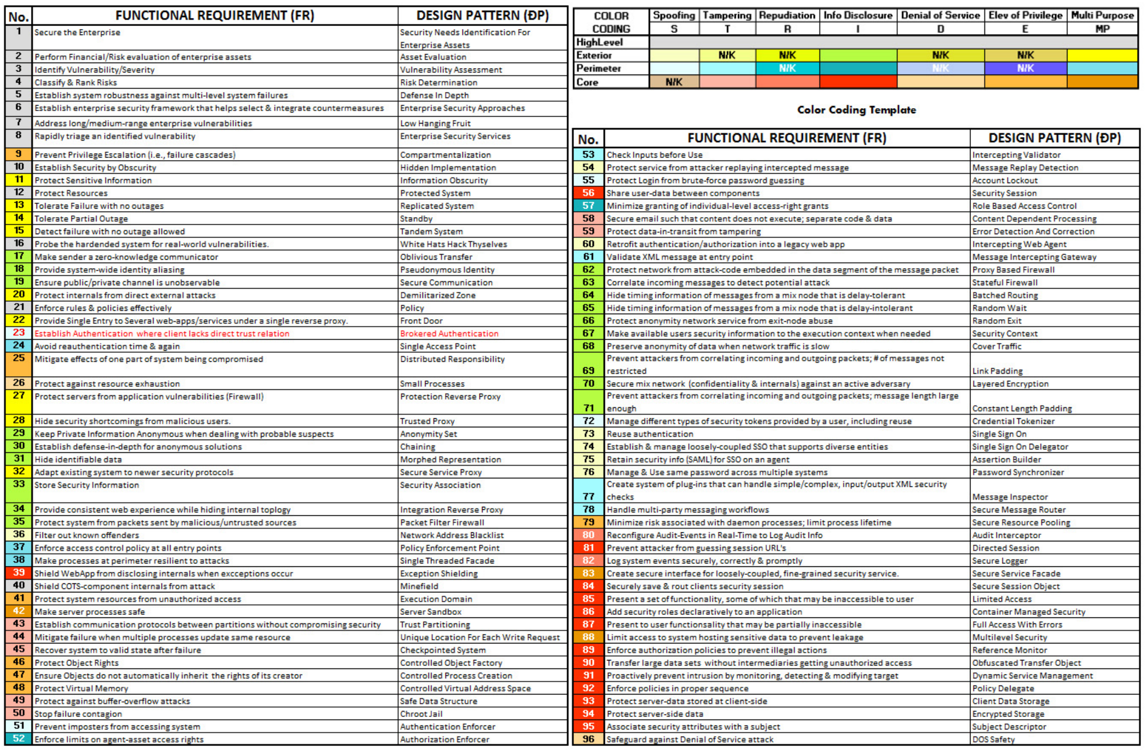
- 1st Leg of /\/**:** Review the ĐP and explicate the embedded conceptual hierarchy.
- 2nd Leg of /\/: Top-Down decomposition & Axiomatic framing of the ĐP; disambiguation of the FR↔DP mappings.
- 3rd Leg of /\/: Bottom-Up integrations, including those at the sub-system/system levels.
- Cross-Domain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

Part 2 illustrates the approach using Cybersecurity Patterns. Section §2 reviews some of the recent high profile cyberattacks. Sections §3 to §6 cover the above N model. Section §4 puts forth a generic pattern-language compositional technique using the Axiomatic Design Matrix. Section §7 transcends the technical and into the social to make the case that cyber security patterns of attack ultimately originate in the realm of patterns of cognitive biases. And as information agents, we leave cybertraces of these patterns to our detriment. Boyd's OODA framework [15] provides a way to overcome these failures both at the individual as well as at the state level.

2. High-Profile Cyberattacks

Recent high-profile cyberattacks on public and private institutions have highlighted the growing vulnerabilities of our socio-technical systems. These include the attack on the U.S government (Office of Personnel Management-OPM: sensitive information on 21 million individuals stolen/June-2015), Primera Blue Cross (11 million individuals/March 2015), Anthem (80 million individuals/February 2015), Sony Pictures (Corporate Data Breach/November 2014), Home Depot (56 million individuals/September 2014), JPMorgan Chase (76 million individuals/July-August 2014), eBay (233 million individuals/May 2014), Google (5 million/September 2014), Yahoo Mail (273 million individuals/January 2014), Target (40 million individuals/December 2013), etc.

doi:10.1016/j.procir.2015.08.011



COLOR CODING	Spoofing	Tampering	Repudiation	Info Disclosure	Denial of Service	Elev of Privilege	Multi Purpose
	S	T	R	I	D	E	MP
HighLevel	⬜	⬜	⬜	⬜	⬜	⬜	⬜
Exterior	🟨	N/K	N/K	🟩	N/K	N/K	🟨
Perimeter	🟦	🟦	N/K	🟦	N/K	N/K	🟦
Core	N/K	🟥	🟥	🟥	🟧	🟧	🟧

Color Coding Template

No.	FUNCTIONAL REQUIREMENT (FR)	DESIGN PATTERN (ĐP)
1	Secure the Enterprise	Security Needs Identification For Enterprise Assets
2	Perform Financial/Risk evaluation of enterprise assets	Asset Evaluation
3	Identify Vulnerability/Severity	Vulnerability Assessment
4	Classify & Rank Risks	Risk Determination
5	Establish system robustness against multi-level system failures	Defense In Depth
6	Establish enterprise security framework that helps select & integrate countermeasures	Enterprise Security Approaches
7	Address long/medium-range enterprise vulnerabilities	Low Hanging Fruit
8	Rapidly triage an identified vulnerability	Enterprise Security Services
9	Prevent Privilege Escalation (i.e., failure cascades)	Compartmentalization
10	Establish Security by Obscurity	Hidden Implementation
11	Protect Sensitive Information	Information Obscurity
12	Protect Resources	Protected System
13	Tolerate Failure with no outages	Replicated System
14	Tolerate Partial Outage	Standby
15	Detect failure with no outage allowed	Tandem System
16	Probe the hardended system for real-world vulnerabilities.	White Hats Hack Thyselves
17	Make sender a zero-knowledge communicator	Oblivious Transfer
18	Provide system-wide identity aliasing	Pseudonymous Identity
19	Ensure public/private channel is unobservable	Secure Communication
20	Protect internals from direct external attacks	Demilitarized Zone
21	Enforce rules & policies effectively	Policy
22	Provide Single Entry to Several web-apps/services under a single reverse proxy.	Front Door
23	Establish Authentication where client lacks direct trust relation	Brokered Authentication
24	Avoid reauthentication time & again	Single Access Point
25	Mitigate effects of one part of system being compromised	Distributed Responsibility
26	Protect against resource exhaustion	Small Processes
27	Protect servers from application vulnerabilities (Firewall)	Protection Reverse Proxy
28	Hide security shortcomings from malicious users.	Trusted Proxy
29	Keep Private Information Anonymous when dealing with probable suspects	Anonymity Set
30	Establish defense-in-depth for anonymous solutions	Chaining
31	Hide identifiable data	Morphed Representation
32	Adapt existing system to newer security protocols	Secure Service Proxy
33	Store Security Information	Security Association
34	Provide consistent web experience while hiding internal toplogy	Integration Reverse Proxy
35	Protect system from packets sent by malicious/untrusted sources	Packet Filter Firewall
36	Filter out known offenders	Network Address Blacklist
37	Enforce access control policy at all entry points	Policy Enforcement Point
38	Make processes at perimeter resilient to attacks	Single Threaded Facade
39	Shield WebApp from disclosing internals when excceptions occur	Exception Shielding
40	Shield COTS-component internals from attack	Minefield
41	Protect system resources from unauthorized access	Execution Domain
42	Make server processes safe	Server Sandbox
43	Establish communication protocols between partitions without compromising security	Trust Partitioning
44	Mitigate failure when multiple processes update same resource	Unique Location For Each Write Request
45	Recover system to valid state after failure	Checkpointed System
46	Protect Object Rights	Controlled Object Factory
47	Ensure Objects do not automatically inherit the rights of its creator	Controlled Process Creation
48	Protect Virtual Memory	Controlled Virtual Address Space
49	Protect against buffer-overflow attacks	Safe Data Structure
50	Stop failure contagion	Chroot Jail
51	Prevent imposters from accessing system	Authentication Enforcer
52	Enforce limits on agent-asset access rights	Authorization Enforcer
53	Check Inputs before Use	Intercepting Validator
54	Protect service from attacker replaying intercepted message	Message Replay Detection
55	Protect Login from brute-force password guessing	Account Lockout
56	Share user-data between components	Security Session
57	Minimize granting of individual-level access-right grants	Role Based Access Control
58	Secure email such that content does not execute; separate code & data	Content Dependent Processing
59	Protect data-in-transit from tampering	Error Detection And Correction
60	Retrofit authentication/authorization into a legacy web app	Intercepting Web Agent
61	Validate XML message at entry point	Message Intercepting Gateway
62	Protect network from attack-code embedded in the data segment of the message packet	Proxy Based Firewall
63	Correlate incoming messages to detect potential attack	Stateful Firewall
64	Hide timing information of messages from a mix node that is delay-tolerant	Batched Routing
65	Hide timing information of messages from a mix node that is delay-intolerant	Random Wait
66	Protect anonymity network service from exit-node abuse	Random Exit
67	Make available users security information to the execution context when needed	Security Context
68	Preserve anonymity of data when network traffic is slow	Cover Traffic
69	Prevent attackers from correlating incoming and outgoing packets; # of messages not restricted	Link Padding
70	Secure mix network (confidentiality & internals) against an active adversary	Layered Encryption
71	Prevent attackers from correlating incoming and outgoing packets; message length large enough	Constant Length Padding
72	Manage different types of security tokens provided by a user, including reuse	Credential Tokenizer
73	Reuse authentication	Single Sign On
74	Establish & manage loosely-coupled SSO that supports diverse entities	Single Sign On Delegator
75	Retain security info (SAML) for SSO on an agent	Assertion Builder
76	Manage & Use same password across multiple systems	Password Synchronizer
77	Create system of plug-ins that can handle simple/complex, input/output XML security checks	Message Inspector
78	Handle multi-party messaging workflows	Secure Message Router
79	Minimize risk associated with daemon processes; limit process lifetime	Secure Resource Pooling
80	Reconfigure Audit-Events in Real-Time to Log Audit Info	Audit Interceptor
81	Prevent attacker from guessing session URL's	Directed Session
82	Log system events securely, correctly & promptly	Secure Logger
83	Create secure interface for loosely-coupled, fine-grained security service.	Secure Service Facade
84	Securely save & rout clients security session	Secure Session Object
85	Present a set of functionality, some of which that may be inaccessible to user	Limited Access
86	Add security roles declaratively to an application	Container Managed Security
87	Present to user functionsality that may be partially inaccessible	Full Access With Errors
88	Limit access to system hosting sensitive data to prevent leakage	Multilevel Security
89	Enforce authorization policies to prevent illegal actions	Reference Monitor
90	Transfer large data sets without intermediaries getting unauthorized access	Obfuscated Transfer Object
91	Proactively prevent intrusion by monitoring, detecting & modifying target	Dynamic Service Management
92	Enforce policies in proper sequence	Policy Delegate
93	Protect server-data stored at client-side	Client Data Storage
94	Protect server-side data	Encrypted Storage
95	Associate security attributes with a subject	Subject Descriptor
96	Safeguard against Denial of Service attack	DOS Safety

Fig. 1. List of 96 FRs & DPs based on [2]

Consider the OPM case; here the attackers gained access to the 127-pages long SF-86 disclosure form detailing highly sensitive security clearance data that an applicant submits. This attack remained undetected for a whole year, spanning July-2014-June 2015. Commenting on the breach, FBI Director James Comey said [5]:

If you have my SF-86, you know every place I've lived since I was 18, contact people at those addresses, neighbors at those addresses, all of my family, every place I've traveled outside the United States since I was 18. If I had substantial contact with any non-United States person, it's on there, along with the contact information of that person. Just imagine you were a foreign intelligence service and you had that data, how it might be useful to you. So it's a big deal.

While the cost of defending our cyberstructures as well as the payoffs from successful attacks keeps rising, the cost of launching an attack simultaneously keeps decreasing [7]. Also these are mass-customizable security breaches that have the potential to combine social-media, data-mining and precision targeting. A Reuters report summarizes the OPM incident as follows [6]:*"The OPM breach gave hackers access to U.S. government job applicants' security clearance forms detailing past drug use, love affairs, and foreign contacts that officials fear could be used for blackmail or recruiting."*

3. /\/: CyberSecurity ĐP Catalog (1st-Leg)

A bottom-up inventory of the prevalent Cyber-Security patterns have been attempted by various authors [2, 4 and 8]. Bunke et al. in [8] account for 415 security patterns (1997-2012) through digital-search and other means. And after culling for duplicates, 364 were deemed unique. Even so, duplicates may have been retained. In contrast, Hafiz et al. in [2] account for a limited set of 96 security patterns (1997-2012) put together using meticulous semantic mappings. Furthermore, this set is organized around a purpose-based, FR-centric framework, namely defending against the STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege) threat-model. In contrast, Bunke et al. [8] use a DP-centric application-domain approach (example: network, embedded systems, distributed systems etc.) for classifying the various ĐPs. Following Louis Sullivan's dictum "*Form Follows Function*" which is part and parcel of the AD/CT framework, it is logical to attempt the AD: ĐP mash-up using the work by Hafiz et al. [2].

System Security issues may be broken into Core, Periphery and External Zones. Core-Security deals with securing the inner sanctum sanctorum of the system which includes data, processes, and the internal hardware. As an example, it includes securing against DoS (Denial-Of-Service: wherein the limited internal resources get swarmed) attack by continuous monitoring and parsimonious resource utilization. Perimeter Security deals with the *"Quo Vadis?"* aspect of the system that checks and enforces the authentication and authorization of anything that crosses the system boundary. External Security deals with securing system assets that are currently in transit outside the system boundary. The above triple-zone demarcation may sound obsolete in the current context of the immense commons that is opening up (such as Cloud

Computing, BYOD, IoT etc.) and threatening to erode the very concept of a system identity and boundary. It is true that boundaries may shift, ownership may entail joint-ownership, and the "system" may consist of myriad agents working in temporary concert that shifts with time; but at any given point in time, the "system" is expected to exhibit a bounded nature along the lines shown above. Also given the relative size of the External zone, it is expected to dominate as is already evident in the number of ĐPs dedicated to this zone. And as the larger ecosystem evolves, so likewise should the threat model, which in AD-language means studying the evolving FR-realm.

Fig. 1 lists the 96 ĐPs and their inferred FRs (as they are not explicated in [2]). The cells are color coded based on STRIDE as well as Core, Periphery & External. Note that the Design Pattern (ĐP) is different from a Design Parameter (DP) in the sense that the former provides a recurring solution template to a recurring problem context; it is assembled bottom-up by studying an ensemble of paradigm cases prevalent in the industry. In contrast, the Design Parameters (DPs) are defined in a top-down sense, are specific to the problem at hand, and are not driven by exiting industry-wide norms. So how should a bottom-up, FR↔ĐP approach dove-tail with a top-down FR↔DP approach? We take this up next in the context of a Pattern Language.

4. /\/: Pattern-Language & Axiomatic Design (2nd-Leg)

Chris Alexander (founder of the Design Patterns approach) was cognizant of the conflict between “design as selection” vs. “design as invention” when he asserted that [9]: “*Those problems of creating form that are traditionally called "design problems” all demand invention.*”

But Chris was also aware of the human difficulties in scaling the combinatorial complexities inherent in the traditional approach [9]: “*It is..not possible to replace the actions of a trained designer by mechanically computed decisions. Yet at the same time the individual designer's inventive capacity is too limited for him to solve design problems successfully entirely by himself.*”

This is the Lickliderian Human-Machine symbiosis problem discussed in Part 1. To solve this logjam, Chris initially proposed a set-theoretic pattern-language that is mathematical enough to engage the machine, while human enough to break out of the patterns when needed [9]:

> *Caught in a net of language of our own invention, we overestimate the language's impartiality. Each concept, at the time of its invention no more than a concise way of grasping many issues, quickly becomes a precept. We take the step from description to criterion too easily, so that what is at first a useful tool becomes a bigoted preoccupation.*

This framework has significant resemblance to the Axiomatic Design framework. Chris terms the hierarchical FR-set as the set of misfits (M) and the linkages between the elements in M as the linkage set (L) which is akin to the Design Matrix (DM) in AD. These two sets together constitute a linear graph G(M,L). He then posits that this graph can be transformed into the design form by creating what he calls the “Constructive Diagram” (CD) which is indeed a holistic visualization of the Design Matrix (DM) at various levels [9]: “*We shall call a diagram constructive if and only if it is both at once - if and only if it is a requirement diagram and a form diagram at the same time.*”

Chris provides ample illustration of the Construction Diagram approach in [9]. And once the top down misfits & linkages graph G(M,L) as well as the Constructive Diagram (CD) is assembled, the designer may then synthesize the realization set (i.e., the DPs) in a top-down fashion. Here there is a key differences between the G(M,L) approach and AD; the G(M,L) approach does not recognize the zigzag mapping that occurs between the FRs and DPs at every stage of the decomposition. Thus the FRs have to be decomposed (top-down) before the DPs may be assembled (bottom-up). This is of course unrealistic; one cannot decompose the FRs into the next level without first mapping it to the DPs at the same level. This is also the reason why the FR linkage set L is presumed to be known without considering the DPs. Chris did not develop the G(M,L) approach further; instead he moved on and proposed the Pattern Language (ĐP/PL) approach which defines relationships between the patterns in a ĐP Catalog [10]:

> *The elements of this language are entities called patterns. Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.*

Here the fundamental problem is in coming up with an appropriate language ĐP/PL that is sufficiently structured to aid in the bottom-up cataloging of the ĐPs, the iterative, zigzag, top-down/bottom-up composition & decomposition of the design. Also it has to be sufficiently flexible to help identify gaps and allow reaching beyond the current catalog of ĐPs where necessary. This is where integrating Chris’s G(M,L) framework [9], his later work on ĐP/PL [10] along with insights from AD may prove fruitful. Here, the hierarchical bipartite zigzag mapping between FRs↔DPs is key. Relations between ĐPs fall into the following categories:

- Is-A: Pattern B is a subtype of Pattern A: A←B
- Uses: Pattern X calls Pattern Y: Y()←X
- Decomposition: A composed of B, C & D: A [B,C,D]
- Alternatives: A & B are mutually replaceable: {A,B}

In [2], Hafiz et al. have used a bottom-up approach to meticulously put together a series of Construction Diagrams (CDs) that capture the relationships between the various CyberSecurity ĐPs. These are then composed to arrive at a holistic CD which serves as the master Pattern Language that finds an appropriate place for each of the 96 ĐPs and the hierarchical/transformational relations between them. This is indeed a Pattern “Language” as it affords the expression of intent and execution both at a high level of abstraction as well as its transformation down into the concretes.

Fig. 2 converts the PL in [2] into a Design Matrix [DM]. The corresponding FRs & ĐPs are as shown in Fig. 1 along with the color-coding shown therein. The DM is sparse with very few off-diagonal terms. Those on the diagonal that tend to be batched together have been demarcated with a bold red boundary. The design naturally evolves from a high-level grey (top-left) to external-in-green, the periphery-in-blue and the core-in-red tones (bottom-right).

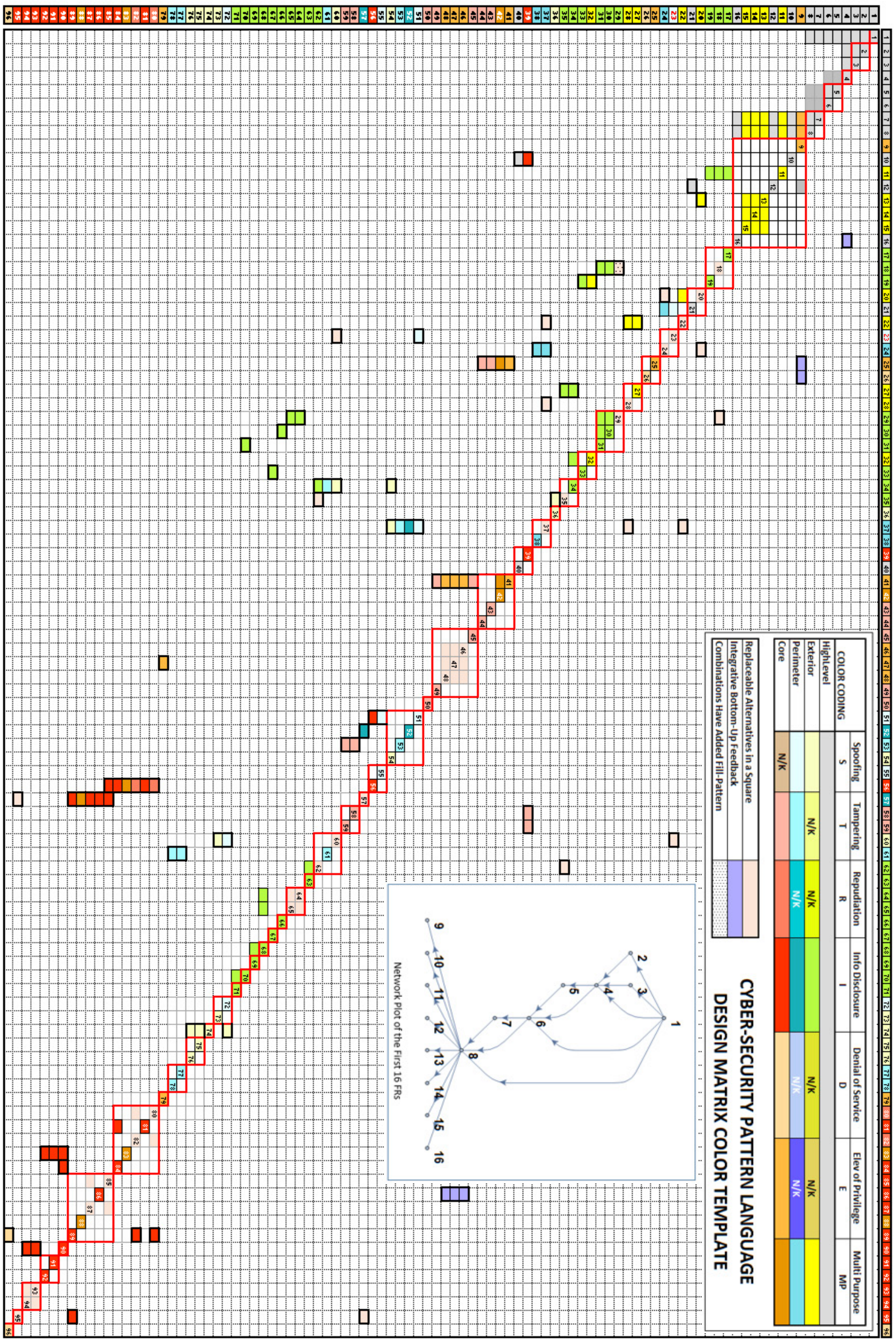

CYBER-SECURITY PATTERN LANGUAGE
DESIGN MATRIX COLOR TEMPLATE
COLOR CODING
Spoofing S
Tampering T
Repudiation R
Info Disclosure I
Denial of Service D
Elev of Privilege E
Multi Purpose MP
HighLevel
Exterior
Perimeter
Core
N/K
Replaceable Alternatives in a Square
Integrative Bottom-Up Feedback
Combinations Have Added Fill-Pattern
Network Plot of the First 16 FRs

Fig. 2. Cyber-Security Pattern Language Design-Matrix based on [2]

The few cells above the diagonal may be easily transformed to render the DM in a lower-triangular form to explicitly satisfy the axioms. Cells in pink are replaceable alternatives (where the FR, ĐP and the corresponding DM rows could be exchanged). Cells in purple allow feedback from lower-level design artifacts to those at the higher level, which asymptotically should fade across time. The DM does indeed provide all the functionality as that of the CDs. Also it completes the "unfinished symphony" that Chris Alexander started-off with in the context of the G(M,L) framework by correcting the stand-alone decomposition/composition of the FRs and ĐPs, and instead using the iterative, zigzag mapping between FRs↔DPs. Also, since the DM matrix can easily be transformed into an incidence matrix for input into a network graphing package (such as Mathematica or Gephi [11]), the various CDs in [2] (both holistic and partial) may be automatically graphed. Fig.2 shows an example of the partial graphing of the first 16 FRs. Also, when viewed as a dynamic network, one may study how the Pattern Language is evolving across time. The DM is the ideal instrument to bring about the proper Lickliderian Human-Machine symbiosis in the context of design towards scale, which indeed has relevance for Agile.

The actual act of design in the context of a ĐP inspired PL involves a top-down traversal from the abstract to the concrete. If all of design were to be articulated via the PL, then there would be mostly "selection" and very little of "invention". It is therefore to be expected that any new FRs (especially at the higher levels) would necessitate splicing in novel, problem-specific FR↔DP mappings into the existing FR↔ĐP mappings. And as these newly created inventions make its way in the collection of similar designs, it may transition into a ĐP.

5. /\/: System Integration (3rd-Leg)

Once the PL has been assembled, it is important to use it as an aid and not as a crutch. Otherwise, as Chris indicated, the "*useful tool becomes a bigoted preoccupation*".

As mentioned earlier, unlike DP, the ĐP is not specific to a given problem. That being the case, how could one ensure that in the case of a specific design problem, the ĐPs being used in the decompositional/compositional phases do provide adequate coverage? How will the ĐP/PL approach assure top-down coverage and bottom-up compatibility? As Chris recognized in [9], the act of design is not merely in finding suitable sub-parts from a patterns cook-book. It instead is in grasping the whole, using composition of cook-book patterns where available, but filling the gaps with new and original one-of-a-kinds that may very well become patterns at a later date. Without such a holistic approach, the design would have holes and lack integrity. This is especially critical in the context of CyberSecurity as the adversary is constantly probing for security holes. Note that in Fig. 2 (color legend), many of the cells are marked "NK"—meaning "not known". These are potential security holes that have not even been formulated. Used in this sense, the PL may be of value in identifying regions of lacunae in the ĐP catalog. Note that these are bottom-up approaches for the system integration of the ĐP-system itself.

Feedback and System Integration also occurs in the actual FR↔DP design. Thus, as noted before, DM cells in purple allow feedback from lower-level design artifacts to those at the higher level, which asymptotically should fade across time. And even though the hierarchical design tree is top-down acyclic, nothing stops us from traversing it in the opposite direction to test and prove that the lower nodes indeed does comply with the design intent set at the higher nodes. These are the many ways in which one may attain System Integration.

6. Cross-Domain Integrations

There is a key assertion made by Hafiz et al. regarding the lack of genericity in the PL [2]: "*There is a pattern language for object-oriented design, and a pattern language for security, and another pattern language for performance.*"

It is true that when the PL is materialized as a set of CDs, it is indeed difficult to find commonalities. However, if the PL is captured as the DM and materialized as an incidence matrix of a network graph, the above ideal of PL genericity ought to be achievable. And as mentioned, this is where the AD/CT-ĐP mash-up may show unexpected positive results both at the tooling level as well as in seeding cross-domain integrations.

For example, the immune system ĐPs that nature has selected across billions of years of evolution of life may have direct relevance for modern Cyber Systems. So likewise when considering ĐPs in the context of 3D-printing and nano-technology. As indicated in Part 1, Prof. Suh et al. adopted the V-Model to reverse-engineer the FR↔DP mappings of biological systems in a bottom-up/top-down sense [12]. When assembled across myriad such ensembles of related natural artifacts, these mappings then become FR↔ĐP mappings that have generic appeal. Nature after all preserves and reuses winning designs in the form of conserved genes that encode successful ĐPs.

7. Cognitive-Biases, OODA, Cyber-Warfare & CASoS

Having made the case for the use of ĐP/PL, consider now the dangers inherent in becoming too wedded to Design Patterns, especially in the context of Cyber-Security. While the scale and scope of the attacks discussed in §2 places it in the realm of Big Data, the defensive measures in terms of detection and year-long response-times have the cadence of wars fought in pre-historic times. This may be related to the fact that the theoretical foundations of cyberwar are not well founded. As Michael Hayden (former Director/CIA) has asserted [14]: *"Rarely has something been so important and so talked about with less and less clarity and less apparent understanding."*

Colonel John Boyd who wrote insightfully about asymmetric warfare does provide the appropriate framework for grasping the problem of CyberSecurity [15] (emphasis added):

> *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 adversary's observation-orientation-decision-action time cycle or loop**. Why? Such activity will make us appear ambiguous*

(unpredictable) thereby generate confusion and disorder among our adversaries—since our adversaries will be unable to generate ***mental*** *images or pictures that agree with the menacing as well as faster transient rhythm or patterns they are competing against.*

Boyd's OODA-loop framework integrates well with AD/CT along with the theory of knowledge hierarchy and dynamics (see Figs. 3a & 3b). The OODA-loop is initiated with the observational phase, or problem perception. This involves sensing signals (both internal and external) from the micro, meso and macro levels (see Part 1). Next is the orientation phase where the problem is abstracted to the right level in the knowledge hierarchy. This is followed by the decision or the design phase. As shown in Fig. 3a, the "what" and the "how" from AD/CT map well with Boyd's OODA framework. Note however that AD/CT provides greater clarity by disambiguating the design process across the appropriate realms such as FRs, DPs, PVs, etc. The action phase of the OODA loop devolves into myriad leaf-level nodes, including the final PV's, the system integration, the testing, the training, etc. Each OODA cycle (both in whole and in parts) feeds the next cycle with the sole purpose of tightening the cognitive decision loop into "fast-transients". The fundamental insight that Boyd put forth was in grasping the role of conflict inherent in nature, how it manifests in human knowledge, and how it shapes our conceptual, hierarchical and fractal decision making processes.

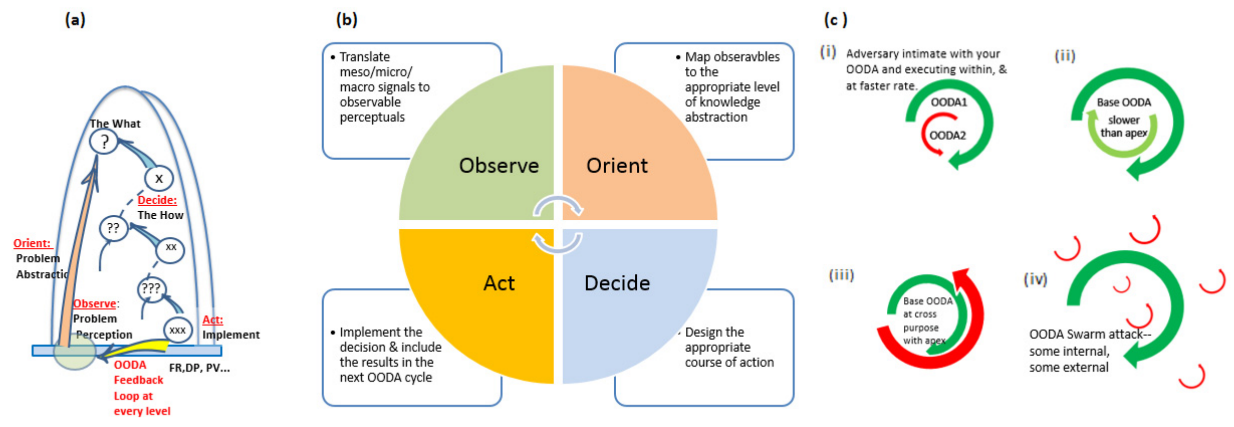

(a)
The What
?
X
Decide:
The How
??
XX
Orient:
Problem
Abstraction
???
Observe:
Problem
Perception
XXX
Act:
Implement
OODA
Feedback
Loop at
every level
FR,DP, PV...
(b)
• Translate meso/micro/ macro signals to observable perceptuals
• Map observavbles to the appropriate level of knowledge abstraction
Observe
Orient
Act
Decide
• Implement the decision & include the results in the next OODA cycle
• Design the appropriate course of action
(c)
(i) Adversary intimate with your OODA and executing within, & at faster rate.
OODA1
OODA2
(ii)
Base OODA
slower than apex
(iii)
Base OODA at cross purpose with apex
(iv)
OODA Swarm attack-- some internal, some external

Fig. 3. (a) Knowledge-Hierarchy & OODA; (b) OODA Loop; (c) Patterns of OODA Loop

Fig. 3c shows some of the patterns of conflict that result when two or more adversaries engage. The green-arcs represent friendly OODA-loops; the red-arcs represent enemy actions. The various CyberSecurity breaches listed in §2 above fall into the pattern shown in Fig. 3c(i). Here the adversary is intimately knowledgeable about the green-teams OODA loop and is synchronized to execute within it, but at a faster cadence. Thus commenting on the OPM breach, S. Gallagher (IT editor at Ars-Technica) noted that [16]:

These attacks have occurred despite the $4.5 billion National Cybersecurity and Protection System (NCPS) program and its centerpiece capability, Einstein. But..Einstein..appears to be incapable of catching the sort of tactics that have become the modern baseline for state-sponsored network espionage and criminal attacks. Once such attacks are executed, they tend to look like normal network traffic.

Fig. 3c(ii) is the case of conceptual friendly-fire where the concrete-level decision-making is happening at a slower rate than the more abstract-layers; this is most likely on account of inadequate levels of automation at the lower levels. The converse of this pattern would be when the abstract layers are executing at too slow a rate to match the rapidly-evolving basal layers; this pattern was considered in Part 1. Fig. 3c(iii) is an extreme case of friendly-fire where the organization is dysfunctional enough to host two antagonistic cultures that are at cross-purposes with each other. Fig. 3c(iv) is the case of the swarm-attack (e.g., DoS), where multiple enemy agents are loosely coordinating scant resources to engage a slower moving target across the overall attack surface. Thus commenting on the origins of the OPM breach, R. Barger, co-founder of ThreatConnect stated that [6]: "*We think it's likely a cohort of Chinese actors, a bunch of mini-groups that are handled by one main benefactor… [and] the group could get software tools and other resources from a common supplier.*"

While patterns in Figs. 3c(i) & 3c(iv) are compelling in capturing the conflicts between nation-states, the friendly-fire patterns in Figs. 3c(ii) & 3c(iii) are equally noteworthy. This is because as a system scales, it is these two seminal patterns inherent in a poorly conceived system that fundamentally compromise the integrity of a system, resulting in an inviting attack-surface for adversaries. Note that these friendly-fire OODA-loops may exist within a given individual or institution. Thus habits of thought and action that form within an individual or an institution can calcify and thwart breaking out of an outdated decision-cycle loop that is dominant. Humans are creatures of habit; and the tools we use (both individually as well as collectively), including tools such as ÐPs can trap us in.

Also, as Nobel laureate Daniel Kahneman has shown [17], we are encumbered with cognitive biases that form patterns of faulty decision-making styles. Our evolutionary bias is towards fast-paced, low-stratum, routine thinking (System 1), while high-stratum, out-of-the-box, complex thinking (System 2) requires intense focus and effort, and is therefore slow. Framing System 1 & 2 within the context of the knowledge-hierarchy dynamics and the proper symbiotic relationship between the human and the machine (as discussed in Part 1), it becomes increasingly clear that System-1 type thinking needs to be relegated more and more to the machine while humans move up the chain towards high-stratum, System-2 thinking. Mental confusion and paralysis of thought happens when Systems-1 and 2 are conflicted, as in Fig. 3c(iii).

From the perspective of CyberSecurity, these inherent biases can be tracked and exploited by adversaries both within and without. For example, phishing-attacks take advantage of our cognitive-biases such as Priming and Confirmation-Bias. Priming sets up conscious and sub-conscious trust-associations (such as the look and feel of a bank web-site) that may be exploited. This may be thwarted by presenting the general website with a custom icon that is individual and personal to the user during the course of a multi-factor authentication. The underlying principle here is that it is not just the user who needs to authenticate; the website also needs to repeatedly establish that it is authentic. Confirmation-Bias is more abstract and internal and alludes to the notion that "*if all you have is a hammer, then everything looks like a nail*"; for example, in CyberSecurity, it could be a socially engineered Trojan sent from a compromised but trusted friend. Patterns of security or the lack of it is thus ultimately a state of the mind; to address it in a fundamental sense requires it to be reduced to patterns discernable in human cognitive biases as shown in [17].

Wars of the future will be fought in the crucible of the individual human mind, but targeted at the mass population-level. Consider the expert assessment of the OPM breach [18]:

- T*he threat could include intruders already in the government whose security credentials were stealthily enhanced during the OPM intrusion, which may have lasted a year before it was detected last April.*
- *It's the digital equivalent of Pearl Harbor. Because people don't see the carnage, they don't recognize that this is the equivalent of an act of war. This is about espionage—Cold War tactics in the modern digital age.*
- *Government is paralyzed. They don't know what to do.*
- *At a minimum, 24 million people have a counter-intelligence problem.*
- *The problems deepen further if the cyber-intruders start to combine that information with data from the social media of targets, which can deepen understanding of their habits, fears, travel plans and much more. It also heightens their vulnerability in travelling outside the U.S.*

The following quote from RUSI makes the stakes abundantly clear [19]: *"The front line exists in the mind of each individual citizen."*

This is where John Boyd's writings indicates that he grasped the essence of the problem from the perspective of knowledge hierarchy and dynamics [20]:

To make..timely decisions implies that we must be able to form mental concepts of observed reality, as we perceive it, and be able to change these concepts as reality itself appears to change....There are two ways in which we can develop and manipulate mental concepts to represent observed reality: we can start from a comprehensive whole and break it down to its particulars or we can start with the particulars and build towards a comprehensive whole. Saying it another way, but in a related sense, we can go from the general-to-specific or from the specific-to-general. A little reflection here reveals that deduction is related to proceeding from the general-to-specific while induction is related to proceeding from the specific-to-general.

He then goes on to prescribe a strategic and deliberate top-down destructive-deduction phase followed by a constructive-induction phase to help continuously rebuild and reshape the knowledge architectures. This is similar to the deliberate mining and gap-closing of the reverse-salients as discussed in Part 1. And it has strategic implications in the modern world of asymmetric conflicts (whether internal or external). This is because establishing conceptually hard-to-reach, knowledge beachheads is the ultimate asymmetry.

In the context of cybersecurity, what it means is that while cybersecurity patterns aggregate current norms and best-practices, attempts to overcome failures such as the OPM-breach will demand breaking current patterns and stepping beyond to establish new patterns at various levels to help close the growing reverse-salients inherent in [7].

Thus in the OPM case, FR23↔ĐP23 is weak: "*Establish Authentication where client lacks direct trust relation."* To understand its weakness, consider the OPM case [6]: "*Both the Anthem and OPM breaches used malicious software electronically signed as safe with a certificate stolen from DTOPTOOLZ Co, a Korean software company, the people close to the inquiry said."*

This can be traced to ĐP23: *Brokered Authentication,* which is a weak ĐP [22]: "*Any compromise of an authentication broker results in the integrity of the trust that is provided by the broker also being compromised."*

Note that the above reverse salient was identified close to 10 years ago, yet it remains open. Closing of these and other similar reverse salients would need a Boydian approach of breaking and reconstituting prevalent ĐPs. This is similar to the motto behind ĐP16: "*White Hats Hack Thyselves"*; except that the White Hats predominantly use heuristics while the Boydian approach frames it in the context of fast-transients to be obtained via purposeful knowledge dynamics. Such transients could include quantum cryptography at the technical level, mass-customized identity-resets with pseudonyms at the government-level, training in identifying and overcoming cognitive-biases at the human-level, etc.

Modern socio-technical systems are massively Complex Adaptive System of Systems (CASoS) [21]. The internet is one such system that includes legitimate as well as illegitimate activities by its various stakeholders. Designing and evolving such systems will involve precipitating and evolving the Pattern Language in each of its constituent CAS-subsystems as well as the overall CASoS-wide Pattern Language (Fig. 4). On account of reductionist cognitive-biases prevalent in modern thinking, reverse-salients often lurk undetected in the interstices between such sub-systems.

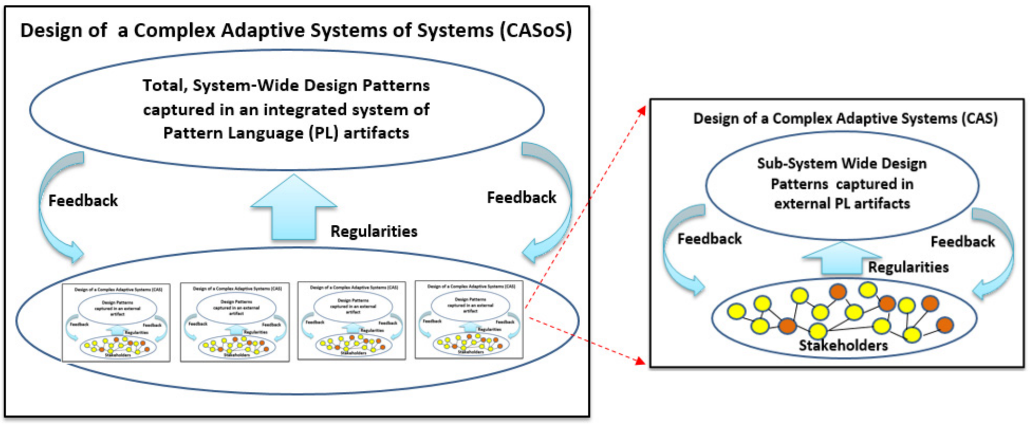

Design of a Complex Adaptive Systems of Systems (CASoS)
Total, System-Wide Design Patterns captured in an integrated system of Pattern Language (PL) artifacts
Feedback
Feedback
Regularities
Design of a Complex Adaptive Systems (CAS)
Sub-System Wide Design Patterns captured in external PL artifacts
Feedback
Feedback
Regularities
Stakeholders

Fig. 4. Design of Complex Adaptive System of Systems (CASoS)

As modern life becomes more and more dependent on such massively complex socio-technical systems, it is imperative that we bring order to the looming chaos.

8. Conclusion

Part 2 illustrated the AD/ĐP mashup in the domain of Cybersecurity. This involved the four-step N-model:

- 1st Leg of /\/: Creating the Patterns Catalog (PC) & piecing together the Patterns Language (PL) along with the Construction Diagrams (CDs), all in a bottom-up fashion
- 2nd Leg of /\/: Integrating PL with the Axiomatic Framework using the far more generic and machine-readable Design Matrix (DM) and then showing how to use this in design by splicing in FR↔DP with FR↔ĐP.
- 3rd Leg of /\/: System Integration at various levels including ĐP, PL, asymptotically vanishing feed-back loops as well as cross-level checks on design intent.
- Cross-Domain Integrations: This step challenged the prevailing view that Pattern Languages are domain-specific. Instead it was argued that the Design Matrix [DM] provides a generic, machine-readable repository for capturing in-domain as well as cross-domain PLs.

Finally in section in §7, the case was made that Cyberwar is being waged at the fine-grain of the individual human mind as the war-front; that cognitive biases are fatal; that Boyd's theory of OODA-loops and fast-transients integrates well with the earlier framework of knowledge-dynamics to help understand why it is important to make and deliberately go about probing and breaking prevalent ĐPs. Also, when placed in the CASoS framework, the case was made that cross-system interfaces are choice spots for probing such weaknesses.

References

[1] Suh NP. The Principles of Design. 1st ed. New York: Oxford University Press; 1990
[2] M. Hafiz, P. Adamczyk, and R. E. Johnson. Growing a pattern language (for security). Onward! 2012 Proceedings of the ACM International Symposium. ACM New York, NY, USA. October 2012
[3] Suh NP. Complexity: Theory and Applications. 1st ed. New York: Oxford University Press; 2005
[4] C.Blackwell, H. Zhu. Cyberpatterns: Unifying Design Patterns with Security and Attack Patterns. Springer. 2014
[5] D. Perera. 21.5 million exposed in second hack of federal office http://www.politico.com/story/2015/07/federal-government-cyber-attack-breach-21-million-people-affect-119918.html July 2015
[6] J. Menn. U.S. Employee Data breach tied to Chinese intelligence http://www.reuters.com/article/2015/06/20/us-usa-data-breach-idUSKBN0OZ20Z20150620. June 2015
[7] http://infosecurityinc.net/wp-content/uploads/2011/07/Consult-Cyber-1Cyber-Threats-Diminishing-Attack-Costs-Increasing-Complexity4.jpg
[8] M. Bunke, R. Koschke, K. Sohr. Organizing Security Patterns Related to Security and Pattern Recognition Reqd. http://www.iariajournals.org/security/ sec_v5_n12_2012_paged.pdf. Int J. On Advances in Security July 2012
[9] C. Alexander. Notes on the Synthesis of Form. Harvard Univ. Press. 1964
[10] C. Alexander. A Pattern Language: Towns, Buildings, Construction. Oxford University Press. 1977.
[11] K. Cherven. Network Graph Analysis and Visualization with Gephi. Packt Publishing. 2013
[12] Jeffrey JD, Lee T, Suh NP. A Function-Based Framework For Understanding Biological Systems. Annu Rev Biophys Biomol Struct. 2004;33:75-93.
[13] Alexander C. The Timeless Way of Building. Oxford University Press, USA. 1979
[14] P. Singer. Cybersecurity and Cyberwar: What Everyone Needs to Know. Oxford University Press. 2014
[15] J. Boyd. Patterns of Conflict. http://www.ausairpower.net/JRB/poc.pdf December 1986
[16] S. Gallagher. Why the "biggest government hack ever" got past the feds. http://arstechnica.com/security/2015/06/why-the-biggest-government-hack-ever-got-past-opm-dhs-and-nsa/ June 2015.
[17] D. Kahneman. Thinking, Fast and Slow. Farrar, Straus and Giroux. 2011
[18] G. Russell Entire US national security system possibly compromised. http://www.foxnews.com/politics/2015/07/23/entire-us-national-security-system-possibly-compromised-by-year-long-cyber/ July 2015
[19] D. Bergman. Psychological Stealth Bombers: The Cognitive Domain as a Physical Battlefield. https://www.rusi.org/downloads/assets/ Psychological_Stealth_Bombers_RDS_Summer_09.pdf June 2009
[20] J. Boyd. DESTRUCTION AND CREATION. http://www.goalsys.com/ books/documents/DESTRUCTION_AND_CREATION.pdf. September 1976
[21] R. Glass et al. A Roadmap for the Complex Adaptive Systems of Systems (CASoS) Engineering Initiative http://www.sandia.gov/ casosengineering/ docs/CASoSEngineeringRoadmap_09.22.08.pdf September 2008
[22] Microsoft Corporation. Brokered Authentication. https://msdn.microsoft.com/en-us/library/ff650014.aspx December 2005