PRE-PRINT
15th Annual Conference on Systems Engineering Research
Disciplinary Convergence: Implications for
Systems Engineering Research

Eds.: Azad M. Madni, Barry Boehm
Daniel A. Erwin, Roger Ghanem; University of Southern California
Marilee J. Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 23-25, 2017

Classifying Emergent Behavior to Reveal Design Patterns

Jack B. Reid[a], *Donna H. Rhodes*[a]
[a]Systems Engineering Advancement Research Initiative, Massachusetts Institute of Technology, E17-361, 77 Mass. Ave, Cambridge, MA, 02139, rhodes@mit.edu

Abstract

Methods for breaking down emergent phenomena into categories typically focus on some measure, qualitative or quantitative, of the degree of complexity of the system. While such categories are useful for clarifying what, exactly, is meant by the word "emergence," they are less useful for developing practical means of identifying, mitigating, or encouraging emergent behaviour. This paper discusses several systems of classification of emergence that focus instead on characterizing either the form of the emergent behaviour or the causal factors that encourage it. These typologies are used as a basis to propose the development of a set of design patterns that are broadly applicable to designing for emergent behaviour in systems of a variety of domains. Case studies are used to illustrate these principles, and further work towards expanding and codifying the principles is discussed.

Keywords: emergence, emergent phenomena, design patterns, design principles, complexity, typology, classification

Nomenclature

Classification, *typology*, and *taxonomy* are three terms used throughout this paper. These terms are commonly used interchangeably. This paper, however, will use Marradi's definitions [1]. *Classification* is the generic term for sorting a set by some defined metric, either quantitative or qualitative. *Systems of classification* or merely *classification* will be used as a generic term, encompassing both typology and taxonomy, as well as any other form of classification. *Typology* refers to a system of classification in which multiple means of division are simultaneously applied. As an example of this, when discussing government systems, a *theocratic, authoritarian government* has two means of classification applied and is the same as an *authoritarian, theocratic government*. A *taxonomy*, on the other hand, arises when multiple means of division are consecutively applied. The modern system of sorting living species, the evolutionary taxonomy, is, as the name suggests, a taxonomy. Thus *Homo sapiens* is not the same as a *sapiens Homo*.

1. Introduction

The word *emergence* as used by systems engineers and scientists can be vague, likely because it is intended to refer to a wide variety of behaviors that are uncommon, hard to predict, often lack defined structure. Nonetheless, these behaviors do seem to share some intuitive similarities that tease at potential generalizable theories regarding them. Many have attempted to more clearly define what emergence should mean, as to enable productive investigation into them and allow those generalized theories to be generated, tested, and proven. Due to the disparate forms of emergence expressed, a natural way to approach this is not to deal with emergence as a whole, but to break it down into categories that can be more readily conceptualized and investigated. Thus classifications of emergence can be found in virtually any discussion of emergent phenomena. This paper surveys several of these methods of classification, discusses the different problems they seek to address, and proposes the development of a set of emergent behavior design patterns

based upon a certain form of typology. The expectation is that these design patterns will enable practicing systems engineers and operators of systems to anticipate potential emergent behavior, more quickly identify it, and have ready access to tools to deal with it. Additionally, these patterns have potential use for education new entrants to the field.

Emergence in this paper is not synonymous with complex systems. The latter typically describes behavior, while the former more typically (though not always) describes some formal aspect of the system. Clearly the two are closely linked with many complex systems exhibiting emergent behavior. Complex Adaptive Systems, for example, are noted for performing a certain emergent form of adaptation (as the name suggests) [2]. Due to this closeness, as well as the fact that neither emergence nor complex systems have definitive, universally agreed upon definitions, and this paper discusses many of these various definitions and means of classification, some overlap may occur.

2. Typologies of Emergence

Many past classifications of emergence have been taxonomies based on the degree of complexity, reducibility, or predictability of the system or behavior. Philosophers in particular tend to favor the latter two measures when constructing taxonomies. Reducibility (i.e. whether system-level emergent behavior is merely the result of complicated interactions of components or is fundamentally *other* from the components) is commonly used to differentiate *nominal* and *strong* emergence in an ontological sense (here ontological refers to the fundamental nature of what emergence is, regardless of whether or not we can ever fully understand it).

Nominal emergence is any form of system property that results from the combined properties of its components [3]. This includes things as simple as the fact that a watch can tell time due to its combination of gears and springs, all the way up to things as complicated as financial markets.

Strong emergence on the other hand refers to a behavior exhibiting "irreducible causal powers," [3] that is, it is an entity in its own right, not explainable in terms of its components, though it may still be dependent on them. The distinction here can be subtle and is not especially relevant here and thus will not be discussed further. Barnes [4] has an excellent explanation for interested readers. It should be noted that while some philosophers argue that conscious thought or life fits into the *strong* emergence category, scientists typically (but not universally) hold that the category is an empty set, as any case of *strong* emergence would be unable to be explained or replicated by the modern scientific method.

Predictability, which is used by philosophers to refer to what is theoretically predictable rather than what is practically or currently predictable, is often used to subdivide *nominal* emergence into further categories like *weak* emergence in an answer to an epistemological question (here epistemological refers to what can be known about emergence, regardless of what it genuinely is). *Weak* emergence is system-level behavior that, while caused by properties and interactions of its components (thus making it fit into the *nominal* category), is not easily explainable by them and requires simulation of potentially extremely high complexity in order to be predicted [3], [5].

Engineers typically are not overly interested in the ontological question of emergence, and often dispense with reducibility as a metric. Additionally, rather than consider predictability in terms of the theoretical (e.g. you have unlimited computing power and infinite computational time), it is instead preferred to think of predictability in a more practical or current sense. When Bjelkemyr et al. use the *weak* and *strong* categories for example, *weak* emergence is defined as that "which can be predicted by experience or extensive modeling and simulation," while *strong* emergence describes properties and behaviors that are displayed by the system, but can't be reduced to known interactions or components, which can't be attributed to an isolated subsystem or part [6]. This shift means that as our scientific understanding and modeling capabilities increase, a behavior could shift from one category to another, unlike in the philosophical typology. Additionally, this typology implies that there is no behavior that engineers and scientists cannot tackle.

Having only two categories in a taxonomy is somewhat limiting, however, and thus Maier expanded these two categories into *simple, weak, strong,* and *spooky* emergence, while making the emphasis on simulation and prediction even more explicit [7]. Basic systems, like mechanical watches, were downgraded to the *simple* category. The *weak* category was reserved for behavior that is consistently predictable using simulations of equal complexity to the actual system, but not with more abstract models. Most systems studied using agent-based models would fall into this category. *Strong* emergence is now a behavior whose general causal chain can be traced down to the component level, but simulations fail to consistently replicate. An excellent example of this is financial bubbles, which are easily explained on evening news, but resist accurate real world predictions and diagnoses. Lastly *spooky* emergence refers

to all system-level behavior that is flatly inexplicable and unpredictable using current scientific knowledge, such as conscious thought.

Note that as these taxonomies have progressed, the emphasis on modeling and prediction has grown in importance. Some go as far as to say that emergence should be defined subjectively as whatever behavior is unexpected to an observer informed of the basic rules of the system [8], [9]. This makes the category that a behavior fits into not merely dependent on current scientific knowledge and engineering methodologies, but on the individual observer of the system. Under such definitions it may occur that, for example, a specific behavior would not be emergent to an experienced system designer, but would be emergent to a novice system operator. This concept is potentially problematic for discussing emergence in a productive manner as certain complexity scientists have noted [2].

Not all engineering-specific systems of classification have focused on predictability, however. Complexity, commonly measured using amount of interaction with extra weight going to unique interactions, is another popular metric for distinguishing categories of emergence. Bar-Yam of the New England Complex Systems Institute based his emergence taxonomy on degree of interactions both among the components of the system, and between the system and its operating environment [10]. Szabo et al. even took the bold step of proposing a quantitative measure of emergence based upon interaction and possible system states [11]. Others have used grammar systems to formally define and quantify emergence in terms of interaction [12], [13]. Bar-Yam argued that these forms of classification enable a better understanding of the possible states of a system and more easily identify behavior dependencies. A robust, formal mathematical definition of emergence would certainly lend towards more accurately determining what aspects of systems lead to emergent behaviors. On the other hand, such a definition may exclude behaviors commonly considered to be emergent and have, as of yet, only been demonstrated on either rather basic systems or systems with few (or one) unique components. Thus the feasibility of applying these definitions to more complicated designed systems, much less systems of systems, remains unproven. Note that these taxonomies should not be confused with similar ways of classifying complex systems by degree of complexity, such as Sheard's typology [14]. While such systems of classification are quite useful, they focus on describing the system as a whole, rather than specific behaviors which is the subject of this research.

The systems of classification (all taxonomies) discussed so far have many uses. For one, they serve to clarify what is meant by "emergent behavior" so that it can have genuine meaning in the systems engineering context other than "I didn't see that coming" (excluding the aforementioned exception to this). Additionally, they help inform design themes for handling emergence, such as robustness, communication, and learning [15], for defining risk of undesirable emergence [16], and general mindset that a systems engineer should have when confronting emergence [17]. That said, these taxonomies are aimed more towards the experienced systems engineer and the development of theory. They do not lend well towards concretely explaining the concept of emergence to a layperson, nor do they immediately suggest concrete ways of either reducing the likelihood of a specific undesirable emergent behavior occurring or diagnosing and addressing such behavior once it has started occurring. Other forms of classification, based more on the type of expression of the emergent behavior can better serve this purpose.

Fromm [18], for example, developed a taxonomy that still uses the basic simple-to-strong structure proposed by Bjelkemyr et al. [6] and Maier [7], but also adds subcategories based on traits of the behavior exhibited, as can be seen in Table 1. Fromm's taxonomy is quite useful for illustrating emergence with real world examples and for discussing potential causes (he does so mostly in terms of feedback loops across different spatial and temporal scales).

Table 1. Fromm's Taxonomy of Emergence [18]

Type	Subtypes	Examples
Type I: Simple/Nominal Emergence without top-down feedback	a) Simple Intentional b) Simple Unintentional	a) Mechanical Watch, Steam Engine b) Pressure, Temperature, Slope of a Sand-pile
Type II: Weak Emergence including top-down feedback	a) Stable b) Instable	a) Flocking, Self-Organization of the Internet b) Financial Bubbles, Demographic Clustering
Type III: Multiple Emergence with many feedbacks	a) Stripes, Spots, Bubbling b) Tunnelling, Adaptive Emergence	a) Zebra Stripe Formation, Financial Market Cycles b) Natural Evolution, Adaptive Systems, Scientific Revolutions
Type IV: Strong Emergence	None	a) Life, Culture

Mogul chose to dispose of the complexity hierarchy taxonomy altogether, instead creating a typology using descriptions of behavioral patterns, such as "unwanted synchronization" or "livelock," to create categories. Additionally, Mogul used the same structure to make a typology of causes of emergent behavior. The typology of behaviors can be seen in Table 2 and the causes in

Table 3. It should be noted that Mogul did not intend this list to be exhaustive, but rather to serve as a starting point to be developed further [19]. While Mogul's typologies are restricted to undesirable emergent behavior (which he called "emergent misbehavior"), they could readily be expanded to include desirable behavior as well. Additionally, this "emergent misbehavior" of Mogul's appears to be a subset of Troncale's "system pathologies." [20]

Table 2. Mogul's Proposed Typology of Emergent Behaviour [19]

Behaviour Types	Description	Examples
Thrashing	When competition over a resource results in switching costs between the sharing parties dominate actual useful work	Computers when insufficient working memory is allocated for the task at hand, a notable instance of this was the IBM/370 mainframe computers [21]
Synchronization	When components who time-varying behaviour normally is uncorrelated becomes correlated.	The frequency of pedestrians steps on the London Millennium footbridge [22], router protocol message synchronization [23]
Oscillation	When the system periodically switches between states due to some feedback loop.	Gliders in Conway's Game of Life, PsEPR herding [19]
Deadlock	When progress stalls due to a circular set of dependencies	Mars Pathfinder software failure [24]
Livelock	When progress stalls because multiple components keep adjusting in such a way as to remain counteracting one another	Two individuals "dancing" in a hallway while trying to get around one another, Ethernet Capture Effect [25]
Phase change	When the system switches between states in an erratic or irreversible manner	Northeast Blackout of 2003 [26], [27], transition from synchronized traffic flow to a traffic jam [28]

Table 3. Mogul's Proposed Typology of Causes of Emergent Behaviour [19]

Type of Cause	Description
Unexpected Resource Sharing	The system designer assumed that separate components had access to separate resources when in fact the resources are shared and insufficient
Massive Scale	The number of communicating components in the system is large enough to give rise to complex global behavior, even if individual components have simple behaviors
Decentralized Control	Distributed systems that lack central controls, and hence suffer from incomplete knowledge and delayed information, can exhibit oscillations and chaos
Lack of Composability	When components are not composable (i.e. lack strictly defined interfaces), modularity may not simplify behavior
Misconfiguration	Beyond literal errors, in complex systems, it is often too difficult for operators to understand global consequences of local configuration choices
Unexpected Inputs or Loads	While not all undesired behavior in response to an unexpected input or load is emergent, sometimes implementers draw the boundaries of "the system as a whole" too close to what they are responsible for implementing.
Information Sharing Delays	Latency makes a system harder to understand and harder to control, which can lead to oscillations and chaos.

Unlike many of the other systems of classification which were developed as taxonomies, Mogul's typology does not seek to place an individual instance of emergent behavior into certain type to the exclusion of others. Rather multiple behavior types may be exhibited in one instance of emergent phenomena. While the matched frequency of pedestrian's steps on the London Millennium Footbridge is listed as an example of synchronization in Table 2, the overall behavior of the can be characterized as an oscillation caused by an expected feedback loop. Applying these "tags" on cases of emergent behavior gives system designers and operators another handle on which to conceptualize their own work, by enabling quick comparisons between similar cases to be made.

3. Design Patterns

This research seeks to generate and compile a set of design patterns to enable systems engineers to quickly recognize emergent behavior in their system, find similar cases in other systems, and identify possible avenues to address the mitigation or inducement of such behavior in the future. In this research, *design patterns* are used to refer to a subset of *design principles*. The latter refers to an abstract rule that produces concrete solutions to design problems. The *design patterns* are a subset of these aimed at specific problems, apply to the design itself (as opposed to the design process), and have some degree of provability or verifiability. Design patterns can be contrasted with *design heuristics*, the other subset of design principles, which are general, unverifiable rules-of-thumb largely aimed at the design process. Design patterns and heuristics have previously been used by Mekdeci [26] as well as Sheard and Mostashari [29] for similar purposes in the field of survivability and resilience. An example of a design pattern that arose based off the design principle "Margin" in Mekdeci's work is shown in Table 4. Though design patterns have been developed for numerous fields and predate the cited researchers [30], this paper uses an altered version of Mekdeci's general format for describing them.

Table 4. Example Design Pattern for Margin from Mekdeci [26]

NAME	MARGIN
TYPE	Operational
DESIGN PRINCIPLE	Margin (Jackson, 2012; Richards, 2009)
PROBLEM	Capability is not being provided adequately.
CONTEXT	Perturbation causes the output of system to be inadequate.
SOLUTION	Have system be capable of producing more output than necessary.
UNINTENDED CONSEQUENCES	May add require additional components, meaning unnecessary size (reducing the strategy of miniaturization), and/or unnecessary weight (making the strategy to mobility difficult to achieve).
EXAMPLE	Extra-long wings on an A-10 aircraft, which generates enough lift in the event that part of the wings are destroyed or damaged.
RELATED PATTERNS	Heterogeneity, Physical Redundancy

When formulating these design patterns, there are certain caveats that should be kept in mind. First, though design patterns are commonly referred to as general solutions to common problems based on a certain design principle, we do not claim that these patterns "solve" emergence. While we disagree with the definition of emergence as the unexpected as some others have done [8], [9], most, if not all, forms of relevant emergence cannot be consistently predicted or designed for. Rather, through observing certain similarities in circumstances surrounding emergent behavior and the forms that certain behaviors take, we aim to expand the ability to at least anticipate it to some degree. So in this context, the phrases "suggested action to alleviate" and "suggested action to encourage" are used instead of "solution."

Similarly, while Mogul referred to his latter typology as a "taxonomy of causes" [19], we argue that the explicit cataloging of strict causal factors of emergent behavior is overly ambitious for what systems scientists are currently capable of (or may ever be capable of). Instead this research seeks to catalogue common circumstances that are conducive to emergent behavior. For instance, a system merely being of massive scale (one of Mogul's listed causes) does not guarantee emergent behavior, nor can it often be truly stated as *the cause of* the emergent behavior. Rather, systems of massive scale tend to *be more likely* to exhibit emergent behavior, and accordingly we use the phrase "causal factor" to describe these traits. We define this in the following way: *A causal factor is any aspect of the system, which, when removed or changed, is likely to reduce the occurrence of emergent behavior, or, when induced, is likely to increase the occurrence of emergent behavior*. Clearly this limits the testability of the design patterns. Nonetheless, it is expected that many of these patterns will be noncontroversial for experienced systems engineers and appear reasonable upon reflection. Moreover, we believe that these patterns will prove useful to practicing systems engineers, particularly those with limited experience with emergent phenomena.

The first step towards developing the proposed design patterns is to generate a typology of emergent behavior, a set of common causal factors, and suggested actions to either alleviate or induce the behavior. The suggested actions will form the basis of each design pattern. The required pieces of information should not be listed independently, but instead linked to one another along with concrete examples.

Table 6 shows a summary of such information for three types of emergent behavior: Synchronization, Phase Change, and Oscillation. While this table contains descriptions and potential consequences of each emergent behavior, it lacks clear explanations of the common casaul factors and the suggested actions. Some of the former are more clearly defined in Table 5, while the latter will be more fully described in the proposed design patterns, discussed later in this paper. A key element is to keep each type well-documented both in its examples and in its suggested actions. For example, Over Design Capacity has been found to be a feasible action in a variety of domains: from traffic jams, where the transition from synchronized flow to congestion cannot occur over a certain density [31], [32], to online communication, well-illustrated by the PsEPR herding case where the initial herding behavior and oscillation from one server to the next cannot occur unless the number of clients is above a certain threshold [19]. Note that the common casaul factor "density" exists in both of these examples as well. These can be found both in field specific research (as the above examples were) or in more generalized discussions of emergent behavior, such as De Wolf's and Holvoet's

proposal of two specific design patterns for dealing with emergence resulting from decentralised coordination: gradient fields and market-based control [33].This general system of cataloging is not without precedence and has been used previously to link cause, effect, solution, and examples for different types of perturbations [34].

Table 5. Descriptions of three types of causal factors

Causal Factor	Explanation	Examples
Density	As the energy or material density of a system increases, feedback loops are accelerated and strengthened, nonlinear behavior becomes more apparent, and likelihood of a perturbation increases.	Spontaneous Traffic Jam[28]; Laminar vs Turbulent fluid flow; Locust Swarms[35]; PsEPR herding [19]
Frequency Matching	Two or more known feedback loops or oscillations that were expected to have different frequencies are instead found to have mutually reinforcing frequencies.	Vortex Induced Vibration, London Millennium Footbridge Vibration [36]
Perturbation / Nucleation Site	There exists some temporally and spatially limited variation to input to the system, either endogenous or exogenous, that can agitate local nonlinearities, potentially initiating a chain-reaction across the system	Laminar vs Turbulent fluid flow; Spontaneous Traffic Jam[28];

Table 6. Descriptions, Causal Factors, and Suggested Actions for three types of emergent behaviour

Emergent Behavior	Synchronization	Phase Change	Oscillation
Description of Emergent Behavior	Components of the system or of the environment whose time-varying behavior is expected to be uncorrelated become correlated	The system switches between two or more states in an irregular, unintended, or irreversible way	The system or some set of components periodically switch back and forth between two or more states in some repeating pattern.
Consequences	Can increase the load on a component or the system as a whole, alternately may result in greater efficiency of a system that is normally random	Can cause the system to act in unpredictably, making control and monitoring more difficult. Can also dramatically increase or decrease system performance.	The rapid state changes can inhibit productive function or cause undue loading to some parts of the system.
Common Causal Factors	Unexpected Inputs; Decentralized Control; Frequency Matching; Unexpected Feedback Loops, Density	Density; Perturbation/Nucleation Site	Unexpected Feedback Loops, Frequency Matching
Suggested Action(s) to Alleviate	Over Design Capacity; Induce Randomness; Introduce Perturbations	Remove Perturbations; Over Design for Capacity	Decoupling; Induce Randomness
Suggested Action(s) to Encourage	Minimize Perturbations; Market-Based Control; Gradient Fields	Induce Perturbations; Add Energy	Introduce Time Delays
Example	Frequency of pedestrian steps on the Millennium footbridge [22]; router protocol message synchronization [23]; PsEPR herding [19]; Laminar fluid flow	Spontaneous Traffic Jam[28]; Laminar vs Turbulent fluid flow; Cascading Network Failure [26], [27]; Locust Swarms[35]	Vortex Induced Vibration, London Millennium Footbridge Vibration [36]; PsEPR herding [19]; Gliders in Conway's Game of Life

Once the above catalog has been created (or at least a substantial start has been made), it is then possible to use the suggested actions as design patterns. Table 7 shows an example of such a design pattern for "Over Design Capacity." In a full digital catalog, the various causal factors, suggested actions, and examples could be linked to the expanded explanations in order to facilitate easy exploration by the reader. Similarly, the related patterns could also be linked.

Table 7. Example proposed design pattern

Over Design Capacity / Increase Capacity	
Emergent Behavior Types Addressed	Synchronization; Phase Change
Context	Emergent behavior, potentially exacerbated by the material or energy density in the system, results in higher than expected load on some component or the system as a whole
Action	Allow for extra load capacity than would be necessary should no emergent behavior occur.
Unintended Consequences	May result in induced demand; will likely require additional resources (particularly physical space) that may be costly or not be available
Example	Increased memory capacity resulted in trashing behavior like that seen in the IBM/370 become less common [21]; opening the shoulder of a highway as an additional lane can (at least temporarily) reduce traffic [37]
Related Patterns	Margin, Heterogeneity, Physical Redundancy

While this format is useful for a reader to quickly reference information about specific behaviors, it is limited in its ability to search in the opposite direction (e.g. a designer has noticed that their system-as-designed contains some unexpected feedback loops and want to understand what kind of behaviors might occur if this not addressed before implementation). To serve this need, a cross-mapping table, relating causal factors and suggested actions to emergent behavior has been generated and is shown in Table 8.

Neither the design patterns nor the cross-mapping table are intended to be exhaustive. Instead they are intended to serve as a structure for evolving a full catalog.

Table 8. Emergent Behaviour Cross-Mapping

		Common Causal Factors						
		Density	**Perturbation / Nucleation Site**	**Unexpected Feedback Loops**	**Frequency Matching**	**Unexpected Inputs**	**Decentralized Control**	**Massive Scale**
Suggested Actions to Alleviate	**Over Design Capacity (Margin)**	Synchronization		Synchronization	Synchronization	Synchronization	Synchronization	
	Induce Randomness	Synchronization		Synchronization Oscillation	Synchronization Oscillation	Synchronization	Synchronization	
	Induce Perturbations	Synchronization Phase Change	Phase Change	Synchronization Oscillation	Synchronization Oscillation	Synchronization	Synchronization	
	Decoupling				Oscillation			
	Remove Perturbations	Phase Change	Phase Change					

There are four primary intended users of these design patterns: systems architects/engineers (the people designing and creating the system), system operators (the people running the system on a day-to-day basis), analysts (those charged with diagnosing problems and rectifying them), and students. Designers can use these patterns to help anticipate emergent behavior and adjust their plans to either encourage or discourage the development of emergent behavior once the system is put into operation.

Both operators and analysts can use these patterns to more quickly identify and react to emergent behavior in their systems. This is line with the Cynefin framework's suggest course of action "Probe-Sense-Respond" in the complex domain [38]. By showing system operators and analysts the common causal factors and common suggested actions to alleviate, they more quickly identify the correct elements of their system to probe, know what they should be looking to sense, and develop a response. As discussed previously, the historical taxonomies have focused on the systems architect's role in working with emergence and been less relevant to the analyst. Previously, this need has been noted by De Wolf and Holvoet, who attempted to remedy it by applying design patterns to a subset of emergent behaviors, specifically decentralized coordination [33].

Lastly, "students", be they grade school or university students or practicing engineers learning about systems engineering, can use these design patterns to get a more concrete grasp on what emergence means in applied use. Given emergence has highly varied meanings across fields (and even within them) as well as its common English definition of "becoming visible over time", the concept can be difficult to teach to students. This is particularly true for those who are not native English speakers as it requires a non-literal interpretation of the colloquial meaning of "emergence." An anecdotal experience suggests this difficulty in explanation is eased through the use of examples from a variety of domains, particularly those with personal or professional familiarity to the student. The proposed catalog of design patterns would formalize this process and allow for more detail and clarity in the explanation. Once a student has been familiarized with these various types of emergent behavior and examples of them, they will be better prepared to consider the systems of classification proposed by Bar-Yam [10], Maier [7], and the others, as well as the more general modes of thought useful for tackling emergence, such as those discussed by Keating [15].

While most of the examples of emergent behavior cited in this paper are those of designed systems, the natural world exhibits innumerable instances of emergent behavior that is also worth considering. Such cases have been thoroughly studied by organizations like the Sante Fe Institute and often demonstrate aspects of direct relevance to system designers, such as evolved modularity [39]. The effort to develop this design pattern catalog is in early stages. Ongoing research continues to investigate additional categories of emergent behavior, as well as identifying case examples of such behavior.

Acknowledgements

The authors gratefully acknowledge funding for this research provided through MIT Systems Engineering Advancement Research Initiative (SEAri, http://seari.mit.edu) and its sponsors. We would also like to thank former SEAri research assistant Lucie Reymondet for her initial investigation in this research effort.

References

1. Marradi, A., "Classification, Typology, and Taxonomy," *Quality and Quantity*, 1990, vol. 24, no. 2, pp. 129–157.
2. Holland, J. H., "Complex Adaptive Systems and Spontaneous Emergence," in *Complexity and Industrial Clusters*, A. Q. Curzio and M. Fortiz, Eds. New York: Physica-Verlag, 2002, pp. 25–34.
3. Bedau, M., "Downward Causation and the Autonomy of Weak Emergence," *Principia*, 2002, vol. 6, no. 1, pp. 5–50.
4. Barnes, E., "Emergence and Fundamentality," *Mind*, 2012, vol. 121, no. 484, pp. 873–901.
5. Seager, W., "Emergence and Efficiency," in *The Mind as a Scientific Object*, 1st ed., C. Erneling and D. Johnson, Eds. New York, New York, USA: Oxford University Press, 2005, pp. 176–190.
6. Bjelkemyr, M., Semere, D., and Lindberg, B., "Definition, classification, and methodological issues of system of systems," *System of Systems Engineering—Principles and Applications*, 2008.
7. Maier, M. W., "The role of modeling and simulation in system of systems development," in *Modeling and simulation support for system of systems engineering applications*, L. B. Rainey and A. Tolk, Eds. Hoboken, NJ, USA: Wiley, 2015, pp. 11–44.
8. Ronald, E. M. A., Sipper, M., and Capcar, M. S., "Design, Observation, Surprise! A Test of Emergence," *Artificial Life*, 1999, vol. 5, no. 3, pp. 225–239.
9. White, B. E., "On Interpreting Scale (Or View) and Emergence in Complex Systems Engineering," in *1st Annual IEEE Systems Conference*, 2007, pp. 1–7.
10. Bar-Yam, Y., "A Mathematical Theory of Strong Emergence Using Multiscale Variety," *Complexity*, 2004, vol. 9, no. 6, pp. 15–24.
11. Szabo, C., Teo, Y. M., and Chengleput, G. K., "Understanding Complex Systems: Using Interaction as a Measure of Emergence," in *Proceedings of the 2014 Winter Simulation Conference*, 2014, pp. 207–218.

12. Kubík, A., “Toward a Formalization of Emergence,” *Artificial Life*, 2003, vol. 9, pp. 41–65.
13. Baas, N. A. and Emmeche, C., “On Emergence and Explanation,” *Intellectica*, 1997, vol. 25, no. 2, pp. 67–83.
14. Sheard, S., “A Complexity Typology for Systems Engineering,” in *Twentieth Annual International Symposium of the International Council on Systems Engineering*, 2010.
15. Keating, C. B., “Emergence in System of Systems,” in *System of Systems Engineering: Innovations for the 21st Century*, M. Jamshidi, Ed. Hoboken, New Jersey: John Wiley & Sons Inc., 2009, pp. 169–190.
16. Ferreira, S., Faezipour, M., and Corley, H. W., “Defining and Addressing the Risk of Undesirable Emergent Properties,” in *IEEE International Systems Conference*, 2013.
17. Mcconnell, G. R., “Emergence : Still a Challenge for the Systematic,” in *INCOSE International Symposium*, 2008, pp. 720–734.
18. Fromm, J., “Types and Forms of Emergence,” *Arvix*, 2005.
19. Mogul, J. C., “Emergent (mis)behavior vs. complex software systems,” in *Proceedings of the 2006 EuroSys conference on - EuroSys ’06*, 2006, vol. 40, no. 4, p. 293.
20. Troncale, L., “Would A Rigorous Knowledge Base in Systems Pathology Add Significantly to the Systems Engineering Portfolio?,” in *Proceedings of the Conference on Systems Engineering Research*, 2011.
21. Denning, P. J., “Thrashing: its causes and prevention,” in *Proceedings of the Fall AFIPS Joint Computer Conferences*, 1968, pp. 915–922.
22. Eckhardt, B., Ott, E., Strogatz, S. H., Abrams, D. M., and Mcrobie, A., “Modeling walker synchronization on the Millennium Bridge,” *Physical Review E*, 2007, vol. 75.
23. Floyd, S. and Jacobson, V., “The synchronization of periodic routing messages,” *IEEE/ACM transactions on networking*, 1994, vol. 2, no. 2, pp. 122–136.
24. Reeves, G. E., “What Really Happened on Mars? The Authoritative Account,” *Microsoft Research*, 1997. [Online]. Available: http://research.microsoft.com/en-us/um/people/mbj/Mars_Pathfinder/Authoritative_Account.html.
25. Ramakrishnan, K. K. and Yang, H., “The Ethernet capture effect: Analysis and solution,” in *Local Computer Networks, 1994. Proceedings., 19th Conference on*, 1994, pp. 228–240.
26. Mekdeci, B., “Managing the Impact of Change Through Survivability and Pliability to Achieve Viable Systems of Systems,” Massachusetts Institute of Technology, 2013.
27. Glass, R. J., Beyeler, W. E., and Stamber, K. L., “Advanced Simulation for Analysis of Critical Infrastructure: Abstract Cascades, the Electric power grid, and Fedwire 1 Advanced Simulation for Analysis of Critical Infrastructure 1,” Albuquerque, 2004.
28. Helbing, D., “Traffic and related self-driven many-particle systems,” *Reviews of Modern Physics*, Dec. 2001, vol. 73, no. 4, pp. 1067–1141.
29. Sheard, S. and Mostashari, A., “A Framework for System Resilience Discussions,” in *INCOSE International Symposium*, 2008, pp. 1243–1257.
30. Alexander, C., Ishikawa, S., and Silverstein, M., *A pattern langugage: towns, buildings, construction*. Oxford University Press, 1977.
31. Tadaki, S.-I. *et al.*, “Critical Density of Experimental Traffic Jam,” in *Traffic and Granular Flow ’13*, M. Chraibi, M. Boltes, A. Schadschneider, and A. Seyfried, Eds. New York City, NY: Springer International Publishing, 2014, pp. 505–511.
32. Kerner, B. S. and Konhauser, P., “Cluster effect in initially homogeneous traffic flow,” *Physical Review E*, 1993, vol. 48, no. 4, pp. 2335–2338.
33. De Wolf, T. and Holvoet, T., “Design Patterns for Decentralised Coordination in Self-organising Emergent Systems,” in *Engineering Self-Organising Systems*, S. A. Brueckner, S. Hassas, M. Jelasity, and D. Yamins, Eds. Berlin: Springer, 2007, pp. 28–49.
34. Mekdeci, B., Ross, A. M., Rhodes, D. H., and Hastings, D. E., “A taxonomy of perturbations: Determining the ways that systems lose value,” in *2012 IEEE International Systems Conference, Proceedings*, 2012, pp. 507–512.
35. Buhl, J. *et al.*, “From Disorder to Order in Marching Locusts,” *Science*, 2006, vol. 312, no. 5778, pp. 1402–1046.
36. Dallard, P. *et al.*, “The London Millennium Footbridge,” *The Structural Engineer*, 2001, vol. 79, no. 22, pp. 17–33.
37. Formby, B., “Once TxDOT opened up SH 161 shoulders, traffic started sailing,” *The Dallas Morning News*, Dallas, TX, 14-Apr-2016.
38. Kurtz, C. F. and Snowden, D. J., “The New Dynamics of Strategy: Sense-making in a Complex-Complicated World,” *IBM Systems Journal*, 2003, vol. 42, no. 3, pp. 462–483.
39. Solé, R. V and Valverde, S., “Spontaneous Emergence of Modularity in Cellular Networks,” Sante Fe, 2007-06–013, 2007.