Journal of Database Management
Volume 28 • Issue 1 • January-March 2017

A Framework for Managing Complexity in Information Systems

Mala Kaul, Department of Accounting and Information Systems, College of Business, University of Nevada, Reno, NV, USA

Veda C. Storey, Department of Computer Information Systems, Robinson College of Business, Georgia State University, Atlanta, GA, USA

Carson Woo, Sauder School of Business, University of British Columbia, Vancouver, Canada

ABSTRACT

A particularly difficult, but important, challenge in the design and development of contemporary information systems is dealing with complexity. Although complexity has been richly discussed from various perspectives in the literature, there is limited guidance on how to address complexity in information systems design. This research analyzes different approaches to handling complexity and finds that there exists a plurality of ways in which to address complexity that are dependent upon the given situation. This analysis results in the derivation of a framework for addressing complexity in information systems. The framework explicitly recognizes implications and limitations of decomposition, inner-outer environments, abstractions, and decentralization, and the role of Ontology. Application of the framework is intended to enable information researchers to identify and adapt applicable strategies for managing complexity in any domain.

KEYWORDS

Abstraction, Chaos Theory, Complex Adaptive Systems, Complexity, Decomposition, Inner Environment, Ontology, Outer Environment

INTRODUCTION

An information system is a representation of a real world system (Wand & Weber, 1995). Today's information systems operate in a complex business environment driven by rapid changes in technology and a highly competitive business setting. Information systems must be developed quickly and accurately. Challenges in designing and developing information systems involve understanding what to include or exclude in a design, as well as dealing with the complexity of systems development efforts. The latter involves identifying and deciding how information can be captured and represented (Clarke, Burton-Jones, & Weber, 2013; Hadar, Soffer, & Kenzi, 2014). To be practically useful, of course, an information system must consistently and accurately model applications (Choi, Song, & Han, 2006).

There have been a number of approaches to dealing with complexity in information systems. Notably, are Wand and Weber's (1993) work on decomposition and Simon's (1996) notion of inner and outer environment. These approaches are, generally, based upon the notion of abstraction, although they have not been presented as such. There is, thus, a need to understand and synthesize

DOI: 10.4018/JDM.2017010103

these different approaches in an effort to provide clarification and guidance on their use when dealing with complexity.

The objective of this research is to analyze the work that has been carried out to deal with the notion of information systems complexity, identifying the appropriate use of different approaches for a given situation. The overall contribution is to provide suggestions and guidelines for addressing complexity in designing information systems. Our research proposes that employing various notions of abstraction is useful for this purpose.

The following tasks are involved in carrying out this research.

1. Analyze complexity from the perspective of using abstractions and ontologically-based decomposition.
2. Propose a framework for positioning and conducting research centered on concepts and tradeoffs involved when dealing with complexity from an abstraction perspective (where abstraction is considered a specific mechanism associated with decomposition).
3. Identify existing approaches, both in information systems and other domains (e.g., biology and physics), to addressing complexity. From this we populate our proposed framework for understanding complexity and suggesting when to use a specific approach or a combination of approaches.
4. Illustrate the use of the framework by showing how it can help information systems developers identify and select strategies for managing complexity.

The remainder of the paper is organized as follows. The following section reviews related research. The section, Framework for Complexity in Designing Information Systems, proposes an abstraction-based framework for dealing with complexity. The fourth section discusses the results of doing so. The final section concludes the paper, and proposes areas for future research.

RELATED RESEARCH

The notion of complexity has been richly examined in the literature from a variety of perspectives, including algorithmic complexity (in the form of mathematical complexity theory and information theory), deterministic complexity (in the form of chaos theory and composite theory), and aggregate complexity (Manson, 2001). Algorithmic complexity is primarily concerned with the problem of complexity that makes it difficult to describe systems' characteristics. In the form of mathematical complexity theories, algorithmic complexity calculates the effort required to solve complex problems within information theory (Chaitin, 1992). Complexity is addressed through the simplest computation that can reproduce system behavior. Deterministic complexity posits that the interaction of a few variables creates largely stable systems that are prone to sudden discontinuities (Manson, 2001). Aggregate complexity finds holism and synergies resulting from the interaction of system components. In this form of complexity, the properties of the whole system are different than the sum of the constituent parts; therefore, such systems have emergent qualities that are not separable from the properties of the components (Baas & Emmeche, 1997) .

This section first defines complexity with respect to designing information systems. It then examines complexity from the different perspectives found in the literature. First, though, the main concepts of complexity, emergence, and abstraction are defined and described.

It is, generally, not easy to explain complexity, although a great deal of work has been carried out to try to specify the characteristics of a complex system (Ladyman, Lambert, & Wiesner, 2013). For the purpose of this paper, our definition is limited to the context of designing and developing information systems. The definition proposed by Simon fits this context well: "Roughly, by a complex system I mean one made up of a large number of parts that have many interaction. …, in such systems

the whole is more than the sum of the parts in the … sense that, given the properties of the parts and the laws of their interactions, it is not a trivial matter to understand the properties of the whole" (pp.183-184). If a decomposed part is still too complex, it can further be decomposed, thus forming a hierarchy of systems and subsystems. After a decomposition, in addition to designing and developing the subsystems, we also need to understand the emergent behavior of the system resulting from the interactions of the subsystems.

For emergence, we consider the work of Wand and Weber (1989) who adapt Bunge's (1977; 1979) Ontology to the analysis of information systems. The world is made of *things* that possess properties. Things can associate to form *composite* things. A property of a composite thing that is also a property of a component is termed an *inherited* property. A property of a composite not possessed by any of its components is termed an *emergent* property. A composite thing must possess emergent properties. The emergent behavior of a composite thing is the changes of its emergent properties. Emergence is the cause of complexity, with abstraction a way to reduce this complexity.

In discussing the relationships between idealization and abstractions, Woods and Rosales (2010) use the abstraction definition of Gale (1998). For the purposes of this paper, we adapted the same definition. Abstraction is a process "in which some features are chosen to be represented, and some rejected for representation [...]" (Gale, 1998, p. 167). In our context, this means abstraction is a mechanism for reducing complexity, either through the suppression of detail and focusing on the general aspects through some formal representations, or alternately, by using it as a high level, manageable view that can be subsequently decomposed to provide greater detail.

Complexity in Information Systems

In design science research, complexity is part of the notion of a "wicked" problem. Only through iterations and development will the problem and solution emerge. The problem of complexity is further articulated by who note that in order to capture the complexity of the problem in such a way that the artifact appropriately addresses the problem requirements, it may be helpful to atomize or decompose the problem conceptually. Similarly, Lewis (2012) proposes that decomposability provides a way to uncover simple answers to complex problems and lead to elegant, simple designs. Simon (1991) describes decomposability within the context of a hierarchy of semi-independent or "nearly decomposable systems". In design-science research, complexity has been also discussed within the context of the more theoretical or knowledge outcomes. For example, suggest that theories at a very high level of abstraction often make their relationship to design, and suggestions for design, difficult to discern.

Inner/Outer Environment

Simon (1996) proposed that we can manage complexity by partitioning a complex system into inner and outer environments where we can have very little or no understanding of the inner environment and build artifacts to interface the inner and with the outer environments. In this case, the complexity is reduced to developing the artifacts and studying the outer environment. The general strategy is to deal with complexity by abstracting away the inner and outer environments and focusing on the design of the interface that fulfills the interaction between the inner and out environments. This strategy is general and can be used to explain other approaches, including the types of abstraction presented below.

Abstraction

Another way of addressing complexity is by abstraction. Data abstractions has long been recognized in information systems design, especially in the areas of database design with the work of who introduced the notion of data abstractions, identifying the main concepts as being generalization/specification (is-a), part-whole (part-of), and association (set theory). Abstraction can alleviate complexity because one can focus on a module and its relationship to other modules without being concerned with how the other modules will function. For example, when modeling a part-time student class, which is

a specialization of a student class, one needs only to deal with the special situations of a part-time student, and need not be concerned with how students, in general, behave. Modeling the concept of a part-time student in this manner can be considered as the outer environment where the super-class, student, is the inner environment, and the artifact is the IS-A relationship. Abstraction can also be viewed as a special case of decomposition in that, when we model the details (e.g., going from a specialized class to a generalized one, or going from the whole to the parts), we are decomposing a complex system into sub-systems.

In this paper, generalization is considered to be a decomposition of specialized classes because a specialized class inherits all properties and behaviors from the generalized class and, thus, has more properties and behaviors than its corresponding generalized class. These special cases make the class complex if generalization is not used.special cases that make the class complex if generalization is not used.

Data abstractions and related research have long been applied and analyzed with respect to database design methodologies (e.g., Storey, 1991, 1993; Teorey, Yang, & Fry, 1986). They have implications for deriving and using rules for their employment. Most of the rules developed provided guidance for the assignment of cardinalities for conceptual modeling which are then transformed into logical models for use in implementations. In this way, abstractions serve as a mechanism to deal with the emergent properties of systems. For example, modeling a specialized class from one or more (multiple inheritance) generalized classes or modeling the whole from several parts are a way to form an emergent system using sub-systems.

Decentralization

Decentralizing systems is another way to deal with complexity. Although it can be viewed as a form of decomposition, or even a form of top-down abstraction (whole-part), it can also represent bottom-up modeling using simulation and complex adaptive systems. That is, one can model individual agents, actors, or modules, and the emergent behaviors of executing all of them together can be simulated. In the case of complex adaptive systems, the individual agents can also include adaptive capabilities such as learning and adjusting its own actions. When emergence can only be identified through simulations, philosophers refer to this type of emergence as weak[1] emergence (Bedau, 1997).

Decomposition and Bunge-Wand-Weber (BWW) Ontology

The Bunge-Wand-Weber (BWW) Ontology provides meta-modeling that is strongly influenced by the philosophy of science. They argue that, in order to achieve a better understanding of a system, it needs to be decomposed, and propose a formal approach to doing so.

The BWW approach, has, at the highest level, one object. Obviously, if the details of this object are not known, then the BWW decomposition will not be applicable. This object is decomposed by defining and applying rules regarding what should, or should not, be included in the decomposition, when to terminate the decomposition, and related matters. More specifically, Wand and Weber (1990b) defined *decomposition* as a set of subsystems of a system where: a) every element in the composition of the system is included in at least one of the subsystems in the set; b) the (set) difference between the union of the environments of the subsystems and the composition of the system equals the environment of the system; and c) each element in the structure of the system is included in at least one of the subsystems in the set. Simply stated, a decomposition of a system is a set of subsystems of the system such that the composition of the system equals the union of the compositions of the subsystems in the set (Weber, 1997).

The BWW Ontology pre-supposes a level of modeling where the nature of a system is general. For example, to manage complexity, abstractions are needed. To make abstraction precise, the BWW Ontology provides the structure and formalisms for systems. However, it does not provide specific mechanisms for the actual abstractions (e.g., specialization/generalization, whole-part). Other theories or frameworks are needed to handle the specialized cases. For example, Parsons and Wand (2008)

apply classification mechanisms from Cognitive Science in their instance-based work. The BWW Ontology is then used to provide a structure and formalism to the instance-based framework. It is effective in doing so and is considered a structural approach.

Applications in Database Management

Some specific examples of adoption and use of abstractions that deal with database design and data warehousing include the following: specialization and generalization (e.g., (Parsons & Wand, 2008)). Akoka, Comyn-Wattiau, and Prat (2001) propose incorporating abstractions into data warehousing, asserting that generalizations and mappings need to be examined for the role they can play in modeling, with rules and constraints specified. classify syntactic and semantic aggregation rules as well as user preference rules. Semantic aggregation rules are based upon the semantics of the elements of the conceptual multidimensional meta-model; syntactic aggregation rules express mathematical properties; and user preference provide aggregation rules.

Philosophy of Science

Philosophers have been studying how researchers in natural sciences (e.g., biology and physics) develop and study complex systems. Insights from the philosophy of science, then, should provide insights for this research. Techniques in natural sciences can be divided into two general categories: idealizations and abstractions (Woods & Rosales, 2010).

- **Idealization:** Consider, for example, general relativity. Planets are spherical but never *perfectly* spherical. When one investigates the consequences of Einstein's equations of the gravitational field, one sets up an idealized system composed of two perfectly spherical bodies. In that way, one can take advantage of the symmetries involved. Alternatively, consider a frictionless plane. Even a fully polished surface will not be frictionless, but one can idealize it to be one. In economics, the concept of a "rational agent" is also an idealization in that it assumes an agent that can handle infinite amount of information in the calculation of utility functions.
- **Abstraction:** Abstraction involves selecting things we want to represent and ignoring others as a means to reduce complexity. It is consistent with the abstraction It is consistence the abstraction notion discussed above.in the section above on abstraction.

As stated by Woods and Rosales (2010): "Idealizations are expressed by statements known to be false. Abstractions are achieved by suppressing what is known to be true. Idealizations, we might say, over-represent empirical phenomena, whereas abstractions underrepresent them" (p. 3). Abstraction means leaving something out. Idealization means perfecting; for example, making things perfectly spherical or infinite. Idealization enables the abstraction to be performed. This notion of idealization and abstraction is consistent with abstraction mechanisms as generally employed in information systems design and development such as specialization/generalization, whole-part, etc. To illustrate, below are examples from biology and physics on how idealization and abstraction are used to deal with complexity.

In biology, much research addresses population genetics. The complexity challenge is that, within the evolution of a given population, both random facts and natural selection are relevant and important. Populations are idealized to be an infinite size, so, in this manner, random factors can be neglected. For abstraction, with respect to population level, evolution occurs both at the level of ecological interactions among different phenotypes, and at the level of genetic interactions. Population ecology assumes populations to be genetically homogeneous, and ignores (abstracts from) the underlying genetic detail. On the other hand, population genetics assumes constant population sizes and ignores (abstracts from) ecological interactions. In both cases, the abstracted interactions may not be evident, but are not included in the respective mathematical models.

Similar, relevant situations can be found in physics. A crucial step towards Newton's laws of motion was Galileo's notion of a frictionless motion down an inclined plane. This led to the idea of a "uniform rectilinear motion" which became the basis for a law of inertia which asserts that, in the absence of forces, a body moves uniformly in a straight line (rectilinear motion) or remains at rest. This is referred to as "Galilean idealization" because the surface on which motion occurs is idealized to be perfectly uniform. This notion also leads to abstraction in the sense that no other forces are considered to be at play, when, in reality, they are.

Mathematical physics in the Newtonian style proceeds by idealizing massive bodies to be "material points" or "mass points." In the course of his modelling, Newton sets up the problem as a gravitational system of two bodies. That is, a two-body system is employed as a good approximation for the rest of the universe. Here, idealization (point masses) is combined with abstraction. No other forces; no other bodies are involved.

These examples from biology and physics utilize both idealization and abstraction at the same time in order to study the corresponding complex system.

FRAMEWORK FOR COMPLEXITY IN DESIGNING INFORMATION SYSTEMS

This section derives a framework, shown in Figure 1, for understanding and handling complexity that is inherent in designing complex information systems. The framework is derived from synthesizing the important, relevant perspectives to handling complexity from prior literature and proposes that complexity can be addressed by abstraction and decomposition.

According to Simon (1996), a complex system is hierarchical and [2]near decomposable. Abstraction and the strategy of inner and outer environments are ways to handle complexity. The most commonly used abstractions are generalization and specialization. Generalization takes the commonality (properties and methods) of two or more classes and places them into a super-class. On the other hand, specialization takes a super-class and specializes it into a sub-class by inheriting properties and methods from the super-class and adding additional properties and/or methods. This means we can model abstraction from top to bottom, or from bottom to top, where top is the specialized classes and bottom is the super-class. This forms the causation dimension of the complexity framework.

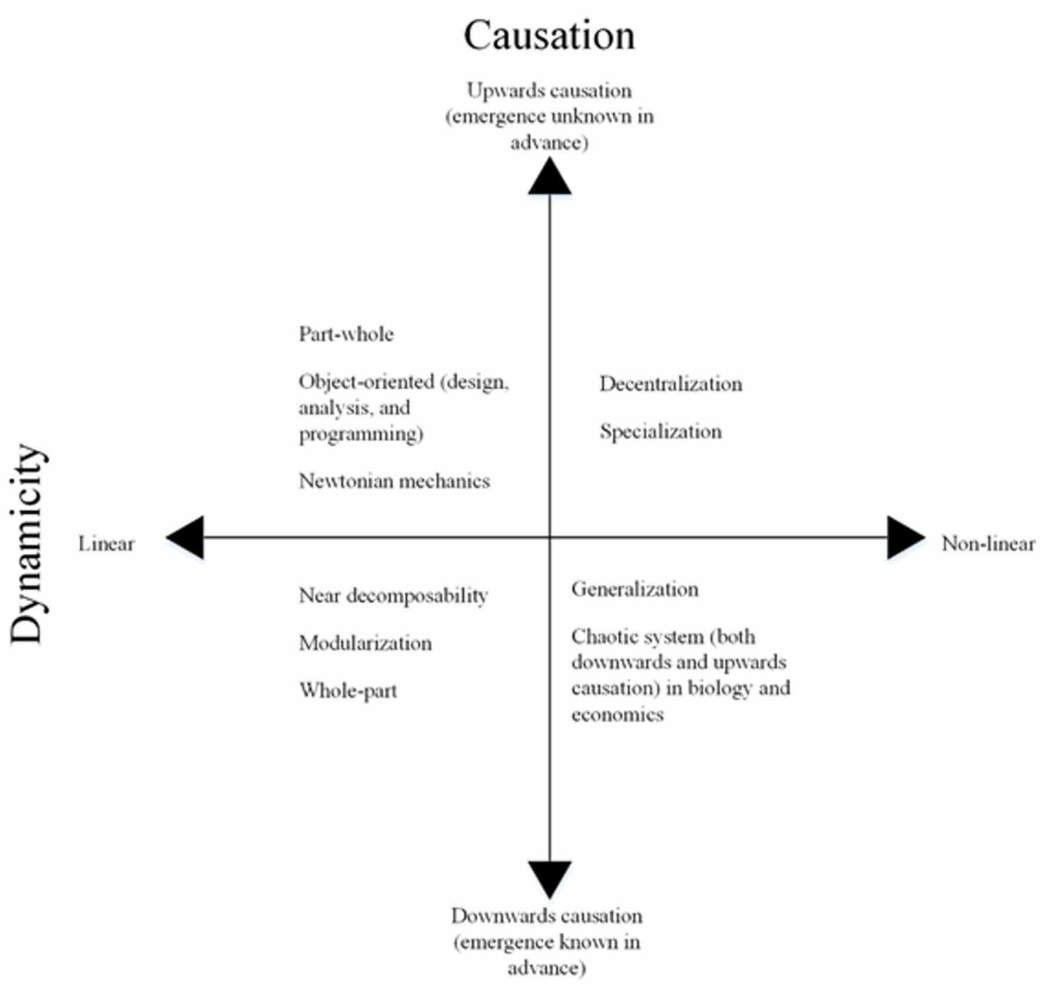
Causation, for the purposes of this research, means causality, but not in the general / social science sense of cause and effect. Causation can come from a top-down or bottom-up approach to performing or doing something (Ladyman et al., 2013). A top-down approach involves a situation in which one knows how the "whole" (e.g., an enterprise information system) operates. Then, the whole system can be decomposed into smaller units and each unit understood individually. Another example could be employing an SAP system to perform data analytics on a company's performance, but needing to examine the individual units to highlight any anomalies.

Bottom-up causation, in contrast, means that one does not know how the whole works, although each of the individual components can be identified and it is known how each component works. Here, we concentrate on the design and development of the components. For example, the failure of an information system might trigger the modification or development of a new system; frustration with an existing system might lead to the acquisition of a bottom-up "Shadow IT" system (Chua & Storey, 2017).

These top-down and bottom-up causations are consistent with Simon's (1996) view of complexity. This is because Simon considers weak emergence to bridge the top-down and bottom-up causations since we might need both to obtain a complete picture of the system or theory. That is, certain emergent properties or behaviors might not be obtained when going from top down (or bottom up) and require bottom-up (or top-down) modeling to see them.

The conception of the second axis of the framework (dynamicity) came from examining the actual usage of super-classes and sub-classes resulted from generalization or specialization. A super-class can

Figure 1. Framework for understanding complexity



be an abstract class in that it has no instances; all instances belong to leaf nodes (the most specialized sub-classes) of the class hierarchy. In this paper, the focus is *not* on abstract classes. Instead, if we know more about the properties and methods of an instance, then more specialized sub-classes can be created into which the instance can fit. Conversely, the less we know about the properties and methods of an instance, the more generalized super-class the instance can be assigned to. This creates a dynamic environment in that identifying the class to which an instance belongs depends upon how much we know about the instance itself. If, at a later point, more information becomes available, then the instance can change its class automatically. This leads to our consideration of dynamicity in handling instances in Figure 1. Linear means no dynamicity or the result is deterministic (i.e., for the same instance, the outcome is the same). Non-linear corresponds to dynamicity or allows for exception handling (i.e., the same instance running through a system can have different results). In the case of decentralization, taking complex adaptive system as an example, the same instance can have different results depending on what each component in the system learned and adapted based upon past experience.

Note that specialization is the reverse of generalization. Similarly, whole-part is the reverse of part-whole. Whole-part (has-a) starts from the whole to identify the parts, whereas part-whole (part-of) considers the parts first, and then combines them together to form the whole.

The dimension of linear vs. non-linear dynamicity in Figure 1 is of the same spirit as some existing work in complexity. Tight-coupling is the same as our linear decomposition. However, Roberts' (1990) complexity is more than our non-linear decomposition, which simply provides the ability to react dynamically. Similarly, the analytical dimension corresponds to our linear decomposition. The behavioral dimension is more general than our non-linear decomposition, similar to Robert's complexity factor. Although the complexity-tight coupling and behavioral-analytical dimensions are useful in understanding complexity, when developing information systems, we are more restricted to mechanisms we can incorporate into systems. The non-linear dynamicity in our framework better reflects this restriction.

The philosophy of science also addresses complexity from the perspective of downwards causation and upwards causation, as well as linear and non-linear dimensions (Ladyman et al., 2013). Ladyman et al. (2013) state that "a system is linear if one can add any two solutions to the equations that describe it and obtain another, and multiply any solution by any factor and obtain another. Nonlinearity means that this superposition principle does not apply" (p.36). Note that downwards causation and upwards causation is fluid in that it is a continuum where some mechanisms might not clearly fit into just one cell. The same holds for linear and non-linear.

Thus, the complexity framework in Figure 1, is deduced from the literature and intended to provide a useful way to visualize and analyze mechanisms to deal with complexity. In particular, abstraction constructs (e.g., specialization/generalization) as found in the information system literature fit this framework well.

DISCUSSION

The two dimensions for understanding complexity in Figure 1 can be used to provide suggestions and guidelines for addressing complexity in designing information systems. That is,

- If emergence of the whole system is known in advance and the system behavior is rather predictable, then near decomposition, modularization, and/or whole-part can be used to address complexity.
- If emergence of the whole system is unknown in advance but the system behavior can be derived from its components, then part-whole can be used to address complexity.
- If emergence of the whole system is known in advance but the system behavior can be dynamic (e.g., depending on how much information we know about an instance), then generalization can be used to address complexity.
- If emergence of the whole system is unknown in advance and the system behavior can only be understand through simulations, then decentralization and/or specialization can be used to address complexity.

Nevertheless, the Figure 1 framework can also be used to learn from other domains and provide additional insights. They are discussed in the remaining of this section.

Since the philosophy of science (e.g., Ladyman et al. 2013) also discusses the same two dimensions as noted above, it should be helpful to identify natural sciences approaches to managing complex systems. Some of these approaches (e.g., Newtonian mechanics – upward causation and linear) might not be useful for designing information systems; however, others might be more applicable. This leads to the identification of the chaotic system that has been using in biology and economics to reduce complexity.

The idea of a chaotic system was proposed by Li and Yorke (1975). Chaotic systems exhibit sensitivity to initial conditions. A small variation in initial conditions of the dynamical systems can lead to bifurcations in the time-evolution of the system. This system, depending on how one uses it, can be downwards causation or upwards causation. Simon (1996) also discusses complexity and chaos, with the idea being that a chaotic system can help to reduce complexity.

Specifically, Simon highlights how in chaos research (non-linear dynamics), thanks to the use of computer simulation models, certain general patterns were discovered. For example, "numerical computations of simple nonlinear systems revealed unsuspected invariants ('universal numbers') that predicted, for broad classes of such systems, at what point they would change from orderly to chaotic behavior. Until high-speed computers were available to reveal them, such regularities were invisible" (p.177).

Complexity can be minimized through a chaotic system. Investigation on "chaos" in dynamical systems has revealed that deterministic (non-linear) systems exhibit sensitivity to initial conditions in their time evolution. If the initial conditions vary, the dynamical behavior becomes complex (chaotic): bifurcations, periodic orbits, limit cycles, etc. There are two main insights. (a) This type of behavior is commonly associated with stochasticity, such as stochastic noise in a system, although chaos theory indicates that this is wrong. (b) That type of dynamical complexity exhibits patterns so then one knows when to expect "chaos" in the dynamical evolution of a system. This becomes a situation where complexity is understood and reduced in the sense that one can rule out stochasticity in the face of erratic behavior in a data set (e.g., time series). Alternatively, one cannot assume that wild fluctuations are the result of stochastic factors with checking for chaos.

Although chaos theory is known to the information systems field it is not as widely used as abstraction mechanisms such as generalization/specialization. The Bunge-Wand-Weber (BWW) Ontology (Wand & Weber, 1989, 1990a, 1990b) is very good in providing structure and formalisms but lacks specificity which is found in chaos theory. Given the importance of the BWW Ontology, we, therefore, suggest that chaos theory could be more widely employed if the BWW Ontology can be used to structure and formalize it for adaption to information systems.

Some further observations can be made from Figure 1 and the complexity work conducted in natural science. First, note that real-world systems are non-linear, but linearity can be used to understand non-linearity and vice versa. Second, idealization and abstraction are ways in which to preserve linearity. They can, therefore, also can be considered as a way to manage non-linearity. Finally, computer modeling, specifically conceptual modeling and computer simulations, can make the study of non-linear systems possible.

CONCLUSION

This paper has proposed a framework for recognizing and understanding complexity in information systems. The work is motivated, in part, by the recognition that today's businesses and society operate in a complex world and that information systems are continuously being developed to help streamline and automate some of this complexity. It was also inspired by the work of Simon (1996) whose understanding of inner and outer environments has been proposed as a way to deal with complexity. Wand and Weber (1989, 1990a, 1990b) insights on decomposition provided further guidance on how complexity might be approached. Then, with further grounding in the philosophy of science, the derived framework is based on the important concept of abstraction which appears in various forms.

There does not appear to be much guidance for explicitly understanding specifically *how* to deal with complexity in information systems design. Therefore, the intended contribution of this research is the framework for recognizing different kinds of complexity to provide a meaningful first step for managing complexity. This paper extends work in the area of abstraction and decomposition to provide a systems view. Note that complexity has been especially recognized, but perhaps in a different form, by research in design science that deals with wicked problems. However, despite such recognition

of complexity both in the development of the artifact as well as in the development and application of knowledge at an abstract/theoretical level, there has been limited discussion in the design science literature on *how to deal with complexity.*

Finally, complexity is a problem that will continue to challenge designers and researchers as we develop and apply more complicated applications that, in some way, reflect the everyday complexity of business and society. Future research is needed to apply the framework to a number of problems in different application domains, both to understand the nature of the environment in which an information system must operate and to assess whether the framework can be used to identify and then help to effectively managing complexity.

ACKNOWLEDGMENT

This work was supported by J. Mack Robinson School of Business, Georgia State University and the Natural Sciences and Engineering Research Council of Canada (NSERC). Special thanks to Alirio Rosales, for assisting with concepts from philosophy of science, Yair Wand and Ron Weber, for inspiration on this research, and the reviewers and the Editor-in chief.

REFERENCES

Akoka, J., Comyn-Wattiau, I., & Prat, N. (2001). *Dimension hierarchies design from UML generalizations and aggregations.* Paper presented at the International Conference on Conceptual Modeling. doi:10.1007/3-540-45581-7_33

Baas, N. A., & Emmeche, C. (1997). On emergence and explanation. *Intellectica*, *25*(2), 67–83.

Bedau, M. A. (1997). Weak emergence. *Noûs (Detroit, Mich.)*, *31*(s11), 375–399. doi:10.1111/0029-4624.31.s11.17

Bunge, M. A. (1977). Treatise on Basic Philosophy: Ontology I: The Furniture of the World. Dordrecht: D: Reidel Publishing Company.

Bunge, M. A. (1979). Treatise on Basic Philosophy: Ontology II: A World of Systems. Dordrecht: D: Reidel Publishing Company.

Chaitin, G. J. (1992). Information-theoretic incompleteness. *Applied Mathematics and Computation*, *52*(1), 83–101. doi:10.1016/0096-3003(92)90099-M

Choi, N., Song, I.-Y., & Han, H. (2006). A survey on ontology mapping. *SIGMOD Record*, *35*(3), 34–41. doi:10.1145/1168092.1168097

Chua, C. E. H., & Storey, V. C. (2017). Bottom-up enterprise information systems: Rethinking the roles of central IT departments. *Communications of the ACM*, *60*(1), 66–72. doi:10.1145/2950044

Clarke, R., Burton-Jones, A., & Weber, R. (2013). *Improving the Semantics of Conceptual-Modeling Grammars: A New Perspective on an Old Problem*. Academic Press.

Gale, G. (1998). Idealization in cosmology: A case study. In N. Shanks (Ed.), *Idealization IX: Idealization in Contemporary Physics* (pp. 165–182). Amsterdam: Rodopi.

Hadar, I., Soffer, P., & Kenzi, K. (2014). The role of domain knowledge in requirements elicitation via interviews: An exploratory study. *Requirements Engineering*, *19*(2), 143–159. doi:10.1007/s00766-012-0163-2

Ladyman, J., Lambert, J., & Wiesner, K. (2013). What is a complex system? *European Journal for Philosophy of Science*, *3*(1), 33–67. doi:10.1007/s13194-012-0056-8

Lewis, K. (2012). Making sense of elegant complexity in design. *Journal of Mechanical Design*, *134*(12), 120801. doi:10.1115/1.4023002

Li, T.-Y., & Yorke, J. A. (1975). Period three implies chaos. *The American Mathematical Monthly*, *82*(10), 985–992. doi:10.2307/2318254

Manson, S. M. (2001). Simplifying complexity: A review of complexity theory. *Geoforum*, *32*(3), 405–414. doi:10.1016/S0016-7185(00)00035-X

Parsons, J., & Wand, Y. (2008). Using cognitive principles to guide classification in information systems modeling. *Management Information Systems Quarterly*, 839–868.

Roberts, K. H. (1990). Some characteristics of one type of high reliability organization. *Organization Science*, *1*(2), 160–176. doi:10.1287/orsc.1.2.160

Simon, H. A. (1991). *The Architecture of Complexity Facets of Systems Science*. Boston, MA: Springer US. doi:10.1007/978-1-4899-0718-9_31

Simon, H. A. (1996). *The sciences of the artificial*. MIT Press.

Storey, V. C. (1991). Meronymic relationships. *Journal of Database Management*, *2*(3), 22–36.

Storey, V. C. (1993). Understanding semantic relationships. *The VLDB Journal—The International Journal on Very Large Data Bases, 2*(4), 455-488.

Teorey, T. J., Yang, D., & Fry, J. P. (1986). A logical design methodology for relational databases using the extended entity-relationship model. *ACM Computing Surveys*, *18*(2), 197–222. doi:10.1145/7474.7475

Wand, Y., & Weber, R. (1989). *An ontological evaluation of systems analysis and design methods*. Academic Press.

Wand, Y., & Weber, R. (1990a). An ontological model of an information system. *IEEE Transactions on Software Engineering*, *16*(11), 1282–1292. doi:10.1109/32.60316

Wand, Y., & Weber, R. (1990b). *Toward a theory of the deep structure of information systems*. University of British Columbia, Faculty of Commerce and Business Administration.

Wand, Y., & Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. *Information Systems Journal*, *3*(4), 217–237. doi:10.1111/j.1365-2575.1993.tb00127.x

Wand, Y., & Weber, R. (1995). On the deep structure of information systems. *Information Systems Journal*, *5*(3), 203–223. doi:10.1111/j.1365-2575.1995.tb00108.x

Weber, R. (1997). *Ontological foundations of information systems*. Coopers & Lybrand and the Accounting Association of Australia and New Zealand Melbourne.

Woods, J., & Rosales, A. (2010). *Virtuous Distortion Model-based reasoning in science and technology*. Springer. doi:10.1007/978-3-642-15223-8_1

ENDNOTES

1 Bedau (1997) considered strong emergence to be scientifically irrelevant (p.376), thus outside the scope of this paper.

2 In this paper we use the term decomposition similar to Simon's notion of "near decomposable."

Mala Kaul is Assistant Professor of Information Systems in the College of Business at the University of Nevada, Reno. Her research focuses on information systems design methodology, evaluation methods, cybersecurity and privacy issues, and health information technology. She has extensive industry experience as an information systems professional in a wide variety of verticals including manufacturing, finance and hospitality. She received her BCom and MA in Industrial and International Economics from Kanpur University, India, and her MBA and PhD degrees from the Managerial Sciences Department and the Computer Information Systems Department, respectively, in the Robinson College of Business at Georgia State University.

Veda C. Storey is the Tull Professor of Computer Information Systems and Professor of Computer Science at the J. Mack Robinson College of Business, Georgia State University. Her research interests are in conceptual modeling, ontologies, intelligent information systems, big data, and design science research. She serves on the Steering Committee of the International Conference of Conceptual Modeling where she is an ER Fellow.

Carson Woo is Stanley Kwok Professor of Business, Sauder School of Business, University of British Columbia. His research interests include conceptual modeling, systems analysis and design, and requirements engineering. Dr. Woo is editor of Information Technology and Systems Abstracts Journal at the Social Science Research Network (ITS-SSRN), and serves or has served on several editorial boards, including ACM Transactions on Management Information Systems, Business & Information Systems Engineering Journal, and Requirements Engineering. He serves as Vice-Chair (2017-2018) and later as Chair (2019-2020) of Conceptual Modeling Conference Steering Committee and has served as President of Workshop on Information Technology and Systems (WITS), Inc. (2004-2006) and chair of the ACM Special Interest Group on Office Information Systems (SIGOIS) 1991-1995.