Complications with Complexity in Requirements

JOHN LESLIE KING and CARL P. SIMON, University of Michigan

Requirements engineering must recognize the difference between complicated and complex problems. The former can lead to successful solutions. The latter should be avoided because they often lead to failure. As a starting point for distinguishing between complicated and complex, this article offers six characteristics of complex problems, with examples from economics, logistics, forecasting, among others. These characteristics make it easier and more systematic to recognize complexity during requirements elicitation and formulation.

Categories and Subject Descriptors: D.2.1 [**Software**]: Requirements/Specifications

General Terms: Design, Management, Reliability

Additional Key Words and Phrases: Complicated, complex, requirements, information systems, software

ACM Reference Format:
John Leslie King and Carl P. Simon. 2015. Complications with complexity in requirements. ACM Trans. Manag. Inform. Syst. 5, 3, Article 13 (January 2015), 12 pages.
DOI: http://dx.doi.org/10.1145/2629375

1. INTRODUCTION

According to the Institute of Electrical and Electronics Engineers (IEEE), professionals must not lead clients into thinking that work can be accomplished when it cannot and must ensure that their work meets ethical standards, such as avoiding harm to innocent third parties [IEEE 2003]. Moreover, project management professionals must deal with complexity in requirements [IEEE 2011]. For this article, *requirements engineering* describes the activity of eliciting, evaluating, and explicating what is required to meet stakeholders' needs and preferences in developing software to solve the stakeholder problems. The term *requirements expert* describes the person responsible for that role. The requirements expert must represent the interests of multiple stakeholders in the project, not only the customer or the software development team. The purpose of the work is to make it easier to honor such responsibilities by explaining how complexity affects requirements and, by extension, the systems development process The work is far from done—the landscape is constantly changing, and requirements experts must keep abreast of the change.

Complexity in software was discussed before the NATO meetings on software engineering [Naur and Randell 1969; Randell and Buxton 1970]. Literally hundreds of articles have touched on the matter since that time. Boehm [1984] considers complexity to be among the most important drivers of software cost. Brooks [1987] calls complexity an inherent property of the "irreducible essence" of modern software systems, along with conformity, changeability, and invisibility. Special concern has been focused on

Authors' addresses: J. L. King, School of Information, 3447 North Quad, University of Michigan, Ann Arbor MI 48109; email: jlking@umich.edu; C. P. Simon, Center for Study of Complex Systems, 321 West Hall, University of Michigan, Ann Arbor MI 48109; email: cpsimon@umich.edu.

DOI: http://dx.doi.org/10.1145/2629375

requirements in the systems development process. Nguyen and Swatman [2003] argue the importance of understanding and, when possible, reducing complexity of the requirements model for managing the requirements engineering process. Milne and Maiden [2012], citing Bergman et al. [2002], note that optimal solutions to complex problems are almost impossible to achieve in a reasonable amount of time. Project failure is most often attributed to poor project management, but poor requirements play a role in failure [Verner et al. 2008].

Requirements are important in the management information systems field, having been cited with respect to issues including agile software development [Cao et al. 2010], generic design goals and requirements [Schmidt-Rauch and Schwabe 2011], executive information systems [Marx et al. 2013], and IT-business alignment [Choi et al. 2013; Ullah and Lai 2014]. Requirements are not the only place to look for improvements in systems development, and complexity is not the only challenge to be faced in requirements. Complexity in requirements is a major challenge, and deserving of ongoing attention.

2. BASIC DISTINCTIONS

It is important to distinguish between *the software system itself* and *the problem the system seeks to affect*. The primary focus of this article is on the problem that the requirements expert seeks to address. The system's complexity must *follow* the complexity of the problem being addressed [Ashby 1956]. The requirements expert must ascertain whether problems to be addressed are too complex for a software system to handle.

The word *system* in requirements engineering typically describes a solution process, algorithm, or computer program. Generally speaking, a system is a set of states or compartments with designated connections, often diagrammed as arrows, that link these states. The requirement expert's flow chart is often a good example of a system, but the problem under consideration might also be embedded in a system that must be considered carefully before proceeding. For example, finding an effective vaccination program is embedded in the transmission system of a disease spread among a population. Finding an effective taxation process can be embedded in a larger economic system. Solution algorithms are systems, as are the problems they tackle. Ignoring the system in which the problem is embedded can lead to disastrous unintended consequences. The pest control agent DDT was designed to save important crops but led to serious environmental damage. The arms race to build more effective drugs to cure diseases led to the evolution of dangerous drug-resistant bacteria.

What makes a problem "complex"? A theoretical computer scientist might point to the class of P and NP problems: a problem is in class P (polynomial time; NP would be nonpolynomial time) if, as the size n of the problem increases, the time $T(n)$ to solve the problem is bounded by a polynomial in n. A problem not in class P would definitely be considered complex. Unfortunately, in requirements engineering, this observation does not help us much because of the difficulty of determining whether or not a problem is in class P. P-class problems are part of a famous unsolved problem in mathematics: does $P = NP$? A million dollar prize is offered for a mathematical solution to this problem.

Harel [2004] is a very readable computer science introduction to design, complexity, P versus NP, and other aspects of algorithmics. Mitchell [2009, Chapter 7] presents a list of technical and mathematical definitions of complexity, including entropy, algorithmic information content, thermodynamic depth, and fractal dimension. Such theoretical computer science and physics definitions of complexity can be useful but are seldom of practical use to requirements experts tackling specific problems.

A more down-to-earth approach defines a system in terms of the number of interdependencies or linkages among the basic components of the problem system. However,

abstract systems with many components connected to all other components can be complicated but are not automatically complex. For example, a large assignment problem with arrows joining all possible assignments can be solved by the simplex algorithm of linear programing in a straightforward way.

The distinction between complicated and complex is not new. Years ago, Cilliers [1998, 2000] noted that entangled problem elements might make something *complicated* and argued that if that problem can be understood, a reliable system can be built. In contrast, for a problem so *complex* that it cannot be understood, it will be difficult to build a system for it. Cilliers recommended that the requirements expert deal only with the complicated and stay away from the complex. The ability to distinguish between complicated and complex is key. Cilliers focused on the view of the beholder of the problem, not the features of the problem, so better training for requirements experts should make it possible to address more complex problems. This work argues that the requirements expert must focus on the problem that the system is to affect, provide guidance to distinguish between complex and complicated, and explain the risks tied to the distinction.

3. DISTINGUISHING BETWEEN COMPLICATED AND COMPLEX

If the problem under consideration has an economics foundation, such as choosing an optimal investment portfolio, marketing a series of products, or devising an efficient taxation system, the requirements expert in search of the underlying system might look to the basic neoclassical model that underlies teaching and research in microeconomics. We turn to this system to define characteristics of simple, complicated, and complex systems. Microeconomics textbooks use strong simplifying assumptions that make economic analysis amenable to mathematical analysis. Other fields that consider populations of individuals, including epidemiology, ecology, and business, often have very similar simplifying assumptions at their foundation.

Consider some of the important simplifying assumptions from economics. Economic agents and firms are homogeneous (e.g., representative consumers or representative firms). Things are at equilibrium (microeconomics is often called *general equilibrium theory*). There are no idiosyncratic organizational structures to worry about among consumers or within firms. Traders choose counterparties randomly rather than through their social networks. Everyone is perfectly rational, and there is no learning or adaptation to feedback. Consumer preferences and firm production strategies do not change and are not related to the actions of other consumers or firms. (So much for education and advertising.) Microeconomic assumptions are rarely connected to macroeconomic outcomes. And there are no significant uncertainties because the system is deterministic. These assumptions lead to the fundamental theorem of welfare economics: "markets work."

The economy is a system in such views, composed of subparts that make up a whole. A system presented to undergraduates with two consumers, two goods, and two firms is *simple*, solvable with planar geometry techniques. As one adds orders of magnitude more commodities, consumers, and firms, advanced calculus and numerical methods might be necessary to find general equilibrium prices in what is now a more complicated problem. The problems remain relatively simple until one introduces heterogeneity, dynamics, complex connectiveness, adaptation to feedback loops, emergence, and uncertainty that *real* economies exhibit. These components can lead to very different outcomes not seen in the simplified approach. These six characteristics separate simple and complicated systems from complex systems. We will examine each in turn.

1. *Heterogeneity.* Heterogeneity of the agents in the system can add complexity to a system quickly. Anything involving humans as agents who make choices according

to varying information and preferences invokes heterogeneity. Agents can be male or female, young or old, black or white, honest or thieving, rational or impetuous, thoughtful or mean, risk averse or risk loving, left brained or right brained, and so forth. These often matter for outcomes. A realistic model of a complex problem has to recognize heterogeneity. In the case of economics, there is heterogeneity among agents, firms, products, services, and so on. Problems involving portfolio choice or product marketing require attention to heterogeneity.

2. *Dynamics, especially sensitive nonlinear dynamics.* Complex problems are *dynamic*. States change over time; the current state influences future states. Mathematically, a dynamic is usually written as $x(n+1) = F(x(n),\ n,\ a)$ so that tomorrow's state $x(n+1)$ is determined by today's state $x(n)$, and possibly by the season n and the background state of the world a. Equilibrium can occur, but only when the dynamic settles down so that $x(n+1) = x(n)$. The path to equilibrium might be more important than equilibrium per se.

   Different kinds of dynamics have different effects. The simplest dynamic is the *linear* dynamic of money in a savings account paying annual interest at rate r; each year's principal multiplied by $(1+r)$.

$$\frac{x(n+1)-x(n)}{x(n)} = r \quad \text{or} \quad x(n+1) = (1+r)\ \times x(n). \tag{1}$$

   The long run solution is simply multiplicative: $x(n) = (1+r)^n \times x(0)$.

   More realistically, especially in economics, ecology, epidemiology, and population studies, the growth rate is a *decreasing* function of current size:

$$\frac{x(n+1)-x(n)}{x(n)} = r - bx(n) \quad \text{or} \quad x(n+1) = (1+r)x(n) - bx(n)^2. \tag{2}$$

   This case leads to a quadratic equation, sometimes called a *logistic equation*, which is a core dynamic in population studies. Some logistic equations give quickly convergent time series. For example, taking $r = .5$ and $b = 1.5$ in (2) yields the "logistic curve" in Figure 1. However, taking $r = 3$ and $b = 4$ yields the much less predictable dynamic in Figure 2. The dynamic in Figure 1 is robust and immune to errors; it converges to equilibrium value 0.3333... from any positive starting point x_0. The series in Figure 2 is sensitive to the slightest change in initial conditions: if $x_0 = 0.332$, $x_{10} = 0.216...$; but if $x_0 = 0.333$, $x_{10} = 0.999...$ (All of the x_n's lie between 0 and 1.) Even computer rounding errors can produce major changes in predicted outcomes.

   What separates the predictable logistic equation in Figure 1 from the chaotic logistic equation in Figure 2? The dynamics of chaotic systems are much more expansive; nearby objects move farther apart each time period. In mathematical terms, points in any one interval at one time n are mapped into a much larger interval in the next time period $n+1$. The predictable logistic equation in Figure 1 has an underlying growth rate of 50%; that of the chaotic logistic equation in Figure 2 is 300%.

3. *Complex connectiveness*. In simple systems, agents, whether people or objects, have connections that can be as orderly as a Cartesian grid or as random as gas molecules bouncing randomly around a room. Neoclassical economics models rarely include the opportunities and constraints that family structures or business organizations bring to the table. In complex systems, agents are connected in possibly intricate "networks," and changes in those networks can lead to changes in the way that information is handled, as well as in outcomes. Systems for halting the spread of HIV, of smoking, of gang activities, or of terrorist cabals have to concentrate on the idiosyncrasies of the underlying networks.

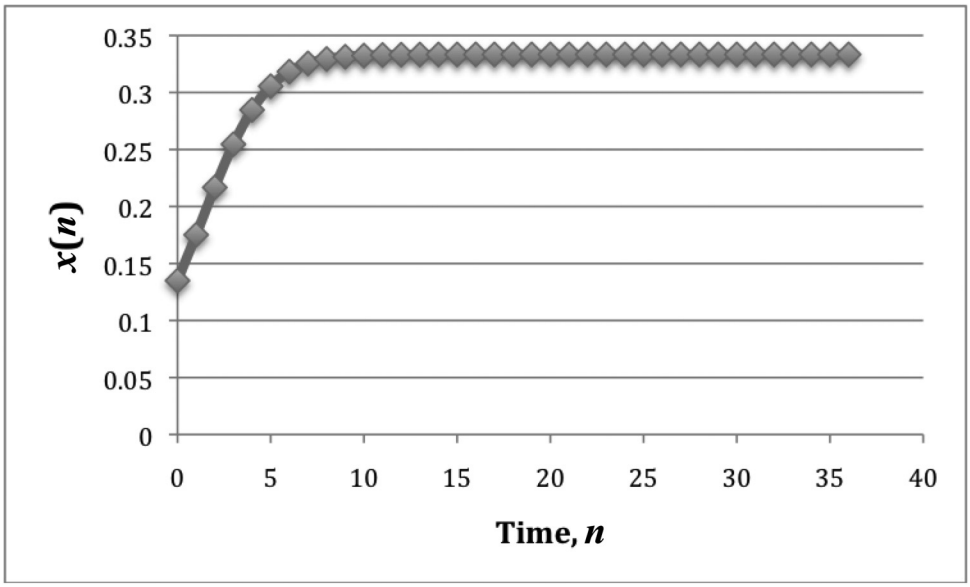

Fig. 1. The dynamics of $x(n+1) = 1.5\,x(n)(1-x(n))$.

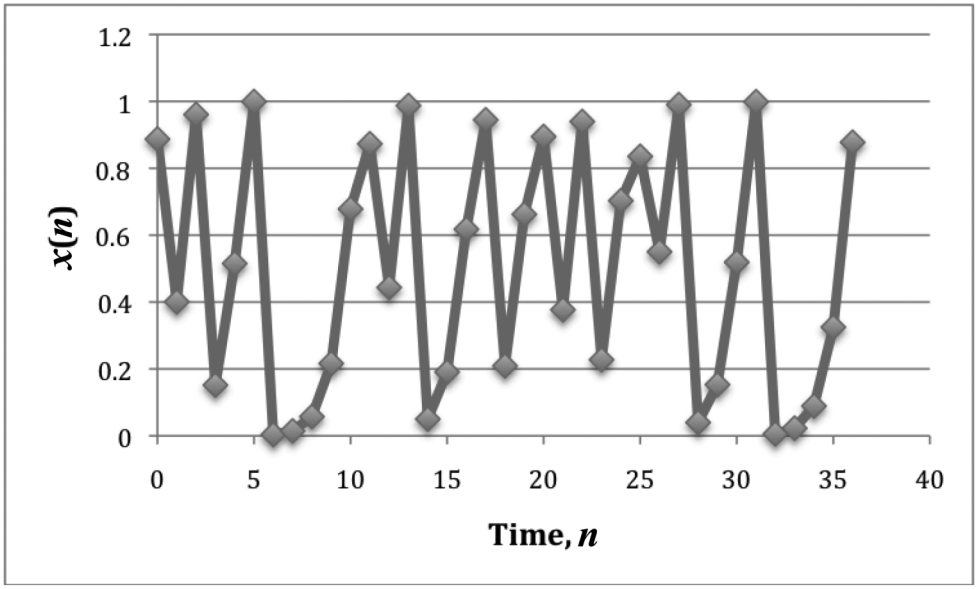

Fig. 2. The dynamics of $x(n+1) = 4\,x(n)(1-x(n))$.

4. *Adaptation to feedback loops.* Intelligent agents learn from experience, and they change their behaviors and preferences to avoid loss while returning to reward. In economics, successful firms attract other firms to an industry. In sports, successful teams adapt to the competition. Engineers who broadcast their predictions of a system can change the system for better or worse as the system adapts to the predictions, as discussed later.

5. *Emergence: the whole is greater than the parts.* Micro changes can affect macro outcomes, so the connections between disparate parts and layers of a complex problem are important. Predicting hurricanes or earthquakes requires understanding of interactions of atmospheric air flows and stresses on tectonic plates, just as predicting a game score requires knowledge of the strengths and weaknesses of individuals and their matchups. Understanding and coping with the housing depression following collapse of the housing bubble in 2007–2008 required understanding individual lenders' and borrowers' activities. Such understanding can be difficult to obtain.

   In complex systems, diverse, dynamic, networked, adapting agents can form a whole that is much more than the sum of its parts. Good examples are the human body with its nervous, immune, coronary, and pulmonary components, or a working economy with its actions and interactions among component firms and consumers. Some complexity presentations (e.g., see Mitchell [2009] and Boccara [2004]) focus on the related property of "self-organization," in which large networks of components operating under simple rules give rise to organized collective behavior without an external or internal controller or leader.
6. *Uncertainty/Stochasticity.* It is difficult to predict precise price moves on the stock market, atmospheric temperature changes, the moods of coworkers, or the performance of sports teams. Uncertainties must be considered, and forecasts should be within ranges when possible. The temperature at a given location 24 hours from now probably will not match a predicted point estimate like 20.00000. . . degrees, but might match a range estimate "between 15 and 25 degrees." It is even more likely to match a probabilistic range estimate such as "80% probability of being between 15 and 25 degrees."

Distinguishing between complicated and complex is largely judgmental. Something with most of the above six characteristics is probably complex, and something with few or none might be complicated rather than complex. A simple Lego structure has none of the six properties. A real house with many more parts might be more complicated, but it is not much different. People build houses and cars and jumbo jets all the time, despite how complicated they are. Even weaknesses in components and human error in combining inputs does not change the fact that such things can be built within tolerances. Linear and quadratic programing algorithms can schedule a week's worth of airline flights for minimized fuel costs, short passenger wait times, and gathering flight personnel from different cities. Outcomes that are the sums of the individual parts might present relatively complicated problems, but these can be solved by well-designed systems.

A linear programming solution to the airline scheduling problem gives macrolevel solutions: the number of required planes and personnel in each city, say, every hour. On the other hand, if one requires more microlevel data, such as the identities of the airline personnel and airplanes, while expecting a nearly perfect assignment procedure, things can get complex. Examples of complexifying requirements include health and family characteristics of personnel to take illness absences into consideration; dynamics and uncertainties of both the status of the equipment and the weather, which might lead to delayed or canceled flights; networks of friendships or enmities among personnel, which could impact the smoothness of service; and adaptation strategies of personnel, customers, and competitors, which could change supply and demand for airline services over time. Including such information may lead to more useful and practical algorithms, but it is likely to turn the problem from a solvable complicated problem to an unsolvable complex problem.

The challenge of complexity lies in the fuzzy and sometimes fluid boundary between complicated and complex. Once again, if a problem has relatively few of the six

characteristics of complex systems, it is probably complicated and might not be too complex to tackle. If it has many or all of the six characteristics, it might be too complex. Much depends on the quality of the requirements expert's understanding of the six characteristics and the risks that they pose, but the requirements expert needs to know the difference between complicated and complex.

4. OTHER DEFINITIONS OF COMPLEXITY

The six characteristics of complexity noted earlier are consistent with definitions of complexity in other nontechnical discussions of the subject. Sometimes the language used varies, but the concepts are similar. Page [2011, Chapter 1] uses the same six criteria that we present. A text that should be of special interest to requirements experts is Axelrod and Cohen [1999], which frames complexity as "variation, interaction, and selection." These are closely related to our concepts of heterogeneity, dynamics/connectiveness, and adaptation/emergence. Boccara [2004] characterizes "complexity" by (1) a large number of interacting agents, (2) emergence, and (3) self-organization.

Holland's *Hidden Order* [1995] characterizes complex adaptive systems by the properties of aggregation, nonlinearity, flows, and diversity, and the mechanisms of tags, internal models, and building blocks. Holland's four properties are similar to our emergence, dynamics, and heterogeneity; the three mechanisms are specific ways for treating connectiveness, adaptation, and dynamics. Holland's *Emergence* [1998] and, as noted earlier, the guided tour of complexity of Mitchell [2009] focus on connectiveness, adaptation, and emergence. Overview papers defining complexity might use "diversity" and "multiplicity" (our heterogeneity) and "interdependence" (our connectiveness) to define complexity, but nonlinear dynamics also plays a role in discussion of sensitive dependence on initial conditions (see Sargut and McGrath [2011]). Marcus et al. [2014] and Simon and Koopman [2005] characterize complexity by the same six characteristics as presented in this article.

5. EXAMPLES OF COMPLEX SYSTEMS

The boundary between complicated or complex can be fluid. Improved knowledge makes things that were once complex "merely" complicated. Improved knowledge in chemistry, physiology, genetics, and other fields has enabled the pharmaceutical industry to "design" drugs for specific effects rather than just study the effects of hundreds or thousands of chemical compounds on tissue samples, cell lines, or laboratory animals. More accurate instrumentation has improved short-run weather prediction. Improving knowledge can accelerate this change within the time frame of the project itself. The boundary can also be somewhat under the control of the requirements expert. By selecting only the simple (if complicated) part of the problem for system application, some headway can be achieved even on a complex problem. For example, bringing a spacecraft down within millimeters of a target might be too complex, but if the requirements expert ensures that acceptable accuracy is "within a hundred meters," it might be possible to build a system that works. Despite the benefits from rapidly improving knowledge or the ability to limit the scope of the project, the requirements expert must consider the six criteria shown earlier and determine the risks involved. Some examples provide guidance on how to do this.

Consider an office sports pool following baseball. Suppose that the client desires a system that will predict game outcomes accurately. Baseball outcomes depend on the success of individual players and their interactions, and some of this is predictable within limits. A "good" major league batter will hit safely only 30% of the time, on average, but performance on a given day will depend on concentration, muscle tone, the opposing pitcher, the weather, and luck. Winning requires, on average, at least two batters per inning getting hits *if* that team's pitcher is holding the other team's batters

in check. Feedback comes in as batters adjust to pitching, and fielders change positions based on batter behavior and their pitcher's performance.

It is easy to see that emergence and stochasticity are at issue; what about "connectedness"? Baseball players bat in assigned order, and action is mainly between the pitcher and catcher. A batted ball rarely involves more than two defensive players. Baseball is certainly complex, but less complexly "connected" than American football, where success of a play depends on each of 11 players performing assigned roles well, or basketball where success depends on constant interactions among 5 players. Even if baseball is not as complex as other sports, the right response to the client is that baseball is "too complex" to build a system that will accurately predict game outcomes. (As they say in sports, "That's why we play the game.") A system might be built to give *some idea* of likely outcomes, but probably not enough precision to guarantee winning the office sports pool.

The odds of predicting the winner in every game of the March NCAA men's basketball tournament are about 9.2 quintillion to one. In 2014, Quicken Loans, with the backing of famed financier Warren Buffett, offered a billion dollar prize to anyone who could beat those odds—a tempting proposal for building a prediction algorithm. The heterogeneity and internal dynamics of the players and teams, the complex interconnections within and between teams, the many adaptations that players and coaches make during the tournament, and the impact of random hot streaks and cold streaks—all suggest the complexity of building a solution system for predicting the winners that would emerge from all individual player and team characteristics. Indeed, by the end of the first week of the 2014 tournament, it was announced that no entrant won the prize.

Public authorities, insurance companies, and many others would like the ability to predict long-run weather accurately. Unfortunately, that is for now "too complex," for reasons the MIT mathematician/meteorologist Edward Lorenz [1963] made clear in the 1960s. His mathematical model of air movements showed that small changes in initial conditions can produce dramatically different weather outcomes. His model (later reduced to a version of the chaotic logistic equation described in Figure 2) was the basis of his talk to the American Association for the Advancement of Science on *Predictability: Does the Flap of a Butterfly's Wings in Brazil Set Off a Tornado in Texas?* [Lorenz 1972]. Even though weather prediction *has* improved dramatically since the 1960s as a result of improved measuring equipment, mathematical models, and computers, long-run weather prediction (e.g., 10 days out) is almost impossible. Lorenz's "butterfly effect," heterogeneous surfaces (e.g., mountains, deserts, cities, pavements, water bodies), the complex connectedness of weather forces on and above the ground, feedback loops among levels of the atmosphere, emergence, and other factors affect weather patterns.

The economy is also complex. Anyone who "beats the market" will reap great rewards, but heterogeneities among investors and investment vehicles, connections among actors (e.g., banks, brokerages, businesses, and bureaucrats), confounding effects of national and international economic events, and natural stochasticities that result from human decision making make the problem complex. That's *before* considering the weather. Weather studies alone suggest that economic dynamics are chaotic: Mandelbrot [1963] found that price changes in cotton markets followed thick-tailed distributions with theoretically infinite variance. The underlying dynamic might be more than nonlinear; it might be chaotic [Silver 2013]. Small errors can make a big difference.

Strange feedback loops are also illustrated by the economy. New competitors entering a market can disturb the behavior of established firms that find it difficult to maintain past successes. Exuberant consumer experiences cause bubbles to collapse. Backlash (a negative feedback loop) against people wearing black and white this year might

(or might not) produce a preference for bright colors next year. Feedback to forecasts can even negate their own predictions: a 2006 prediction that the U.S. housing market would continue to boom accelerated home buying, higher prices, and the collapse of the housing bubble mentioned earlier. Huge measurement errors in economic data can also make accurate forecasting almost impossible.

Improved systems might reduce the spiraling cost of health care, but the human body can be complex. The body involves many heterogeneous subsystems, linked by complex networks. Specialized cells form millions of tiny thin-walled air sacs called *alveoli* and make up the lung. Blood and lymph carry amino acids, electrolytes, oxygen, and carbon dioxide in the circulatory system, but not with perfect predictability. Body temperature is stabilized and homeostasis is maintained by the endocrine system, but again, not with perfect predictability. Much of the body is controlled by neurons and synapses that make up the brain and nervous system, which is not very predictable. The Krebs cycle in general provides energy, and the feedback loops of the immune system adapt to fight disease, but this is not very predictable. The body is a dynamic self-organizing complex adaptive system and usually (but not always) avoids chaotic dynamics in biorhythms and heartbeats.

As if that were not enough, an invisible cloak of heredity covers the body's operation. The health of one's ancestors might be required to predict the future health of any individual accurately. We assume that health is not affected by weather or some other complex factor, but this is only an assumption. Prediction in health is difficult! James Fixx was 52 years old when he died of a heart attack, 7 years after his *Complete Book of Running* promoted exercise for longevity [Fixx 1977]. Prediction within limits is possible: smoking tobacco correlates with shorter life expectancy, but some smokers live a long time. This has contributed to the difficulties of smoking cessation programs.

6. BOUNDARIES AND RISK

Following the guidelines of the IEEE, requirement experts must not promise what they cannot deliver and must do no harm in what they do deliver. The boundary between complicated and complex is fuzzy, but one might sensibly *manage* the risks effectively rather than eliminate risk altogether. Thieves work full time to overcome security in transaction processing systems, so clients seldom insist on technical security alone to eliminate risk. They use a combination of security measures, audit procedures, mechanisms to facilitate reporting of problems (e.g., customer hotlines to report unexpected charges to a credit card), prosecution of offenders, and protocols for handling losses (e.g., forgiving customer losses). They adopt insurance against loss, learn from their experiences, and improve performance whenever they can. They manage risks.

Ultimately, the job of the requirements expert is to ensure that the client understands the risks involved in a proposed project. Complex projects can make sense if everyone recognizes and accepts the risks involved, but the client must recognize that "everyone" can be a large set. Institutional review boards were established in universities to remind researchers that "everyone" includes the subjects of their studies (humans, laboratory animals, etc.), not just researchers.

Table I provides guidance. Columns represent problems to be addressed by systems, examples of systems, and the likelihood of client expectation failure or third-party suffering from system malfunction. Rows provide examples. Risk increases from top to bottom. In some cases, risk can be lowered by eliminating possible suffering from third parties altogether (e.g., in research projects that do not involve human or animal subjects). In some cases, the inherent risks of the problem are so significant that they cannot be avoided (e.g., in systems to control the launch of nuclear weapons during conflict). Risk assessment is usually judgmental, requiring the requirements expert to

Table I. Risks Associated with Systems

Problems the System Affects	Examples of Systems	Risk of Client Expectation Failure	Risk of Suffering by Third Parties
Well-understood problems where client expectations can be met and third parties are unlikely to lose due to system malfunction.	Research-oriented systems with low potential to affect human or animal subjects	Low	Low
Enough complexity to result in client expectation failure but low likelihood of losses to third parties.	A system that does something new for the client, with low potential for harm to third parties	High	Low
Client expectations can be met, but third parties can suffer if systems malfunction.	Short-run weather forecasting that might cause third parties to suffer if inaccurate	Low	High
Enough complexity to result in client expectation failure and third-party losses from malfunction.	Forecasting effects of a vaccine with serious side effects that combats a serious disease	High	High

explore manageable risk as opposed to the often-impossible goal of complete avoidance of risk.

7. IMPLICATIONS FOR PRACTICE

How easy or difficult is it to create a system that addresses the key objectives at an acceptable level of risk of failure or of harm? The six characteristics of complexity and the table of risks should help the requirements expert let the client know whether a given system can be completed or if third parties might suffer from system malfunction. "Creep" (scope creep, mission creep, requirements creep, etc.) usually enters through change processes and should be avoided as the project proceeds.

Can the problem that the system addresses be sufficiently understood to allow for sensible requirements to be specified? If so, *is* it understood at this level? Engaging incompletely understood problems is risky. Systems work can be restricted to a tractable part of the larger problem, but such decisions should be made early and adhered to. Proceeding on the assumption that incompletely understood elements will be "figured out" in the course of the work is risky.

Ancient Greeks saw human failure as usually due to one of two forces: Fate and Hubris [Green 1973]. Failure due to Fate was not necessarily a failing of the individual. On the other hand, Hubris (later described as "overweening pride" that comes before the fall) was not as readily forgiven and was often punished by Fate. People should know better than to engage in Hubris. If someone says that the problem is complex, the requirements expert should ask what is meant by that term. *Complicated* rather than *complex* might bring a correction of terminology to suffice. Hubris is more grave. Clarity about the problem and how solutions might work is needed. Clients satisfied with less ambitious solutions that are "good enough" will be content with "work-arounds," such as having the system halt algorithmic execution to let human intervention take over (provided that appropriate processes for human judgment are in place). Simply hoping that the system works right violates psychological, social, or legal contracts. The requirements expert should engage early and work against this.

8. IMPLICATIONS FOR RESEARCH

New knowledge can help move complex toward complicated and can aid in the creation of methods that improve the likelihood that requirements are handled well in system development. Systems to support science or engineering research are a special case. It

is acceptable for unknowns to spill over into development as long as stakeholders do not suffer and there is an opportunity to learn more about the topic. Close collaboration between the researchers and requirements experts might be needed to answer the question of how systems development can best support research. Sometimes systems are *part* of the research endeavor, in which case commercial, off-the-shelf products warranted to work reliably and predictably are not a good model.

"Clear and unambiguous" requirements seldom arise unless the problems to be affected are quite simple. Learning by doing is common in the requirements phase when the question is *how* to build systems that accomplish objectives. Once sufficient knowledge is acquired, satisfactory systems can be built inexpensively and predictably. Innovative organizations pushing frontiers often face challenges in initial stages where work is more like research than development.

9. CONCLUSION

Systems development has advanced over the past half-century due to the development of techniques to address requirements problems. The work is not done, however. Cillier's observation from two decades ago remains salient: *complicated* and *complex* are different. It is usually easier to create systems that affect simple problems, even if they are complicated. Problems that are complex are hard to address. The requirements expert must assess the characteristics of complicatedness and complexity and their relative risks when executing professional responsibility.

ACKNOWLEDGMENTS

The authors thank the associate editors and anonymous reviewers for their guidance, and our friend Michael Cohen for his inspiration to find the practical in the complex.

REFERENCES

William R. Ashby. 1956. *An Introduction to Cybernetics.* Chapman and Hall, London.

Robert Axelrod and M. Michael Cohen. 1999. *Harnessing Complexity: Organizational Implications of a Scientific Frontier.* Free Press, New York, NY.

Mark Bergman, John L. King, and Kalle Lyytinen. 2002. Large-scale requirements analysis revisited: The need for understanding the political ecology of requirements engineering. *Requirements Engineering* 7, 3, 152–171.

Nino Boccara. 2004. *Modeling Complex Systems.* Springer, New York, NY.

Barry W. Boehm. 1984. Software engineering economics. *IEEE Transactions on Software Engineering* 10, 1, 4–21.

Fred P. Brooks. 1987. No silver bullet: Essence and accidents of software engineering. *IEEE Computer* 20, 4, 10–19.

Lan Cao, Balasubraniam Ramesh, and Tarek Abdel-Hamid. 2010. Modeling dynamics in agile software development. *ACM Transactions on Management Information Systems* 1, 1, 5.

Jae Choi, Derek Nazareth, and Hemant Jain. 2013. The impact of SOA implementation on it-business alignment: A system dynamics approach. *ACM Transactions on Management Information Systems* 4, 1, 3.

Paul Cilliers. 1998. *Complexity and Postmodernism: Understanding Complex Systems.* Routledge, London, UK.

Paul Cilliers. 2000. What can we learn from a theory of complexity? *Emergence* 2, 1, 22–33.

James Fixx. 1977. *The Complete Book of Running.* Random House, New York, NY.

Peter Green. 1973. *A Concise History of Ancient Greece.* Thames and Hudson, London, UK.

David Harel. 2004. *Algorithmics: The Spirit of Computing* (3rd ed.). Addison-Wesley, Reading, MA.

John Holland. 1995. *Hidden Order: How Adaptation Builds Complexity.* Addison-Wesley, Reading, MA.

John Holland. 1998. *Emergence: From Chaos to Order.* Addison-Wesley, Reading, MA.

IEEE. 2003. *Guidelines for Professional Employment: A Framework for Communication.* IEEE, New York, NY.

IEEE. 2011. IEEE Guide—Adoption of the Project Management Institute (PMI®) *Standard: A Guide to the Project Management Body of Knowledge (PMBOK® Guide)* (4th ed.). IEEE, New York, NY.

Edward Lorenz. 1963. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences* 20, 2, 130–141.

Edward Lorenz. 1972. *Predictability: Does the Flap of a Butterfly's Wings in Brazil Set off a Tornado in Texas?* Talk to American Association for the Advancement of Science, November 29, 1972.

Benoit Mandelbrot. 1963. The variation of certain speculative prices. *Journal of Business* 36, 4, 394–419.

Steve Marcus, D. Pincus, and Yoram Vodovotz. 2014. *Systems Modeling: Overview, Relevance, and Advantages to Advance the Study of Resilience*. McGowan Institute for Regenerative Medicine Preprint, University of Pittsburgh, Pittsburgh, PA.

Frederick Marx, Jorg H. Mayer, and Robert Winter. 2013. Six principles for redesigning executive information systems: Findings of a survey and evaluation of a prototype. *ACM Transactions on Management Information Systems* 2, 4, 26.

Alastair Milne and Neil Maiden. 2012. Power and politics in requirements engineering: Embracing the dark side? *Requirements Engineering* 17, 83–98.

Melanie Mitchell. 2009. *Complexity: A Guided Tour*. Oxford University Press, New York, NY.

Peter Naur and Brian Randell (Eds.). 1969. Software Engineering. Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany. Scientific Affairs Division, NATO, Brussels, Belgium.

Lemal Nugyen and Paul A. Swatman. 2003. Managing the requirements engineering process. *Requirements Engineering* 8, 55–68.

Scott Page. 2011. *Diversity and Complexity*. Princeton University Press, Princeton NJ.

Brian Randell and John N. Buxton (Eds.). 1970. Software Engineering Techniques. Report of a conference sponsored by the NATO Science Committee. Scientific Affairs Division, NATO, Brussels, Belgium.

Gökçe Sargut and Rita G. McGrath. 2011. Learning to live with complexity. *Harvard Business Review* 89, 9–10, 68–76.

Susanne Schmidt-Rauch and Gerhard Schwabe. 2011. From telesales to tele-advisory in travel agencies: Business problems, generic design goals and requirements. *ACM Transactions on Management Information Systems* 2, 3, 17.

Nate Silver. 2013. *The Signal and the Noise: Why So Many Predictions Fail—But Some Don't.* Penguin, London, UK.

Carl Simon and James Koopman. 2005. A complex systems approach to understanding the HIV/AIDS epidemic. In *Mathematics for Industry: Challenges and Frontiers. A Process View: Practice and Theory*, David R. Ferguson and Thomas J. Peters (Eds.). S.I.A.M, Philadelphia, PA, 200–221.

Azmat Ullah and Richard Lai. 2014. A systematic review of business and information technology alignment. *ACM Transactions on Management Information Systems* 4, 1, 4.

June M. Verner, Jennifer Sampson, and Narciso Cerpa. 2008. What factors lead to software project failure? In *Proceedings of the 2nd International Conference on Research Challenges in Information Science (RCIS '08)*. 71–80.

Received April 2013; revised March 2014; accepted April 2014