2009

A Theory Of Complex Adaptive Inquiring Organizations: Application To Continuous Assurance Of Corporate Financial Information

John Kuhn
University of Central Florida

A THEORY OF COMPLEX ADAPTIVE INQUIRING ORGANIZATIONS: APPLICATION TO CONTINUOUS ASSURANCE OF CORPORATE FINANCIAL INFORMATION

by

JOHN R. KUHN, JR.
B.S. University of Central Florida, 1994
M.B.A. University of Pittsburgh, 1997

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Management Information Systems
in the College of Business
at the University of Central Florida
Orlando, Florida

Summer Term
2009

Major Professors: James F. Courtney, Paul Cheney

ABSTRACT

Drawing upon the theories of complexity and complex adaptive systems and the Singerian Inquiring System from C. West Churchman's seminal work *The Design of Inquiring Systems* the dissertation herein develops a systems design theory for continuous auditing systems. The dissertation consists of discussion of the two foundational theories, development of the Theory of Complex Adaptive Inquiring Organizations (CAIO) and associated design principles for a continuous auditing system supporting a CAIO, and instantiation of the CAIO theory. The instantiation consists of an agent-based model depicting the marketplace for Frontier Airlines that generates an anticipated market share used as an integral component in a mock auditor going concern opinion for the airline. As a whole, the dissertation addresses the lack of an underlying system design theory and comprehensive view needed to build upon and advance the continuous assurance movement and addresses the question of how continuous auditing systems should be designed to produce knowledge – knowledge that benefits auditors, clients, and society as a whole.

ACKNOWLEDGMENTS

First and foremost, I dedicate this dissertation to my loving and supportive wife, Tracey, two beautiful daughters, Alyssa and Jillian, and of course, my parents. Without their positive attitudes and unwavering support I never would have reached this point. I feel the deepest gratitude and have the greatest regard for my dissertation co-chairs, Drs. Paul Cheney and James Courtney. The two of them helped me through a potentially disastrous situation that would have nipped my academic career in the bud. With their encouragement and support, I persevered. My PhD endeavor has been one of the most challenging yet rewarding experiences of my life, one that I would choose again over and over. I would be remiss without expressing my thanks to Dr. Robin Roberts who, in conjunction with Drs. Cheney and Courtney, convinced me to leave public accounting and pursue a career in academia. Finally, I wish to thank the other members of my dissertation committee (Dr. Bonnie Morris, Dr. Mihir Parikh, and Dr. Ross Hightower) for providing me with insightful comments to constantly improve this dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE: INTRODUCTION

The massive corporate frauds and the related bankruptcies of the late 1990's and early 2000's such as Enron, WorldCom, Adelphia, and others sent shock waves through the financial markets resulting in wide-spread concern over stronger corporate governance, improved internal controls, more transparent and timelier corporate reporting, and expansion of assurance activities – particularly in the area of information technology (IT) controls over financial reporting. The federal government intervened by passing the "Public Company Accounting Reform and Investor Protection Act" (commonly referred to as the "Sarbanes-Oxley Act" or "SOX" after the congressmen sponsoring the legislation). Many consider SOX the most significant corporate governance legislation since the securities laws passed in the 1930's (Hoffman and Rowe 2007). The Act highlights many of the inherent flaws of the corporate financial reporting process and the associated attestation framework. However, the Act does not address one major issue: the attestation timeframe. The traditional once a year audit of corporate financial statements hinders timely and relevant assurance reporting (Vasarhelyi and Halper 1991). The Securities and Exchange Commission (SEC) is currently working towards a mandate that all registrants (primarily public companies) report in the eXtensible Business Reporting Language (XBRL) format (Cox 2008) that will, along with many other advantages, set the stage for continuous reporting and continuous assurance.

Continuous assurance of corporate financial information will require unique information systems (IS), referred to as continuous auditing applications/systems, to meet the needs of the assurers (public accounting firms) as well as other involved parties such as executive management of the organizations producing the financial statements, owners of these organizations, consumers, regulatory authorities, financial market participants, creditors, etc. These individuals and organizations in essence create a dynamic, connected network of actors that act/react to the behaviors of each other and the influences of external forces. For instance, environmental factors may affect the availability of key resources. The combination of drought in Australia, flooding in Europe, and the increased production of biofuels (farmers more interested in growing corn and soybeans rather than hops) have resulted in a shrinking supply of hops forcing beer breweries in Minnesota to find alternative and more expensive sources; experts predict some varieties of hops will increase in price by 400 percent and local beer prices may rise as much as 15 – 20 percent (Dyslin 2007). Continuous auditing systems therefore must be designed to support constantly changing environments, generate new knowledge, and provide decision support in an increasingly complex and connected world.

As a relatively new field, only minimal research has touched on the design of continuous auditing systems and the interaction of the various parties involved in and affected by the attestation process. Diverse views of and approaches to system design exist but, in general, many IS researchers consider system design as the central focus of the IS discipline (Markus, Majchrzak, and Gasser 2002). C. West Churchman (1971) considered design as a thinking process that involved the selection of an alternative from several possible alternatives in order to attain some goal and that design was an activity used to better the human condition (Parrish 2008). As noted in the American Institute of Certified Public Accountants (AICPA) Code of Professional Conduct, the audit profession shares Churchman's view of serving humanity:

“Members should accept the obligation to act in a way that will serve the public interest, honor the public trust, and demonstrate commitment to professionalism.” (American Institute of Public Accountants 2008)

This research study strives to address the lack of an underlying system design theory and comprehensive view in order to build upon and advance the continuous assurance movement. Broadly, can information systems help auditors gain a better understanding of the complex environment in which their audit clients operate in order to properly opine on reported financial statements in a timely manner? More specifically (and the research question of this study), “How should continuous auditing systems be designed to produce knowledge – knowledge that benefits auditors, clients, and society as a whole”? To answer this question, I developed a comprehensive, system design theory specifically for continuous auditing systems based on the theoretical underpinnings of complexity theory and the system requirements for Churchman’s (1971) Singerian Inquiring System (SIS). Subsequently, I instantiated this general model by creating an agent-based simulation model of the airline industry that includes agents representing a specific airline (the audit client), key competitors, consumers, the general economic environment, etc. Simulation results of the specific model facilitated analysis of the selected organization’s ability to continue operations and can act as a decision aid for the auditor’s assessment of this organization’s ongoing viability – a requirement of generally accepted auditing standards (GAAS).

The remainder of this research study consists of three distinct essays (i.e. chapters 2 - 4) with a final chapter tying everything together and continues as follows. First, I present supporting background information and a review of related literature on continuous assurance/auditing, complexity theory, complex adaptive systems, and simulation and agent-based modeling. The next essay examines the key tenets of Churchman’s (1971) SIS, discusses how complex adaptive systems theory and SIS can inform one another, and presents a set of design principles for knowledge management (KMS) and decision support systems (DSS) that support a Complex Adaptive Inquiring Organization (CAIO). The final essay offers background information on the auditor’s going concern opinion, details the design of an agent-based simulation model of the airline industry based on the CAIO organizational learning theory that can be developed into a CA application, and presents simulation results that are intended to support an auditor’s going concern assessment of a particular client in the airline industry.

CHAPTER TWO: LITERATURE REVIEW

Continuous Assurance and Continuous Auditing

Background

The existing approach to auditing corporate financial statements contains two inherent flaws that have raised concerns in both academia and practice. First, fraud, by nature, is meant to deceive and extant research highlights the inability of auditors to consistently detect financial fraud through existing manual procedures (Kuhn and Sutton 2006). Pincus (1989) utilized a checklist of fraud indicators, similar to ones used in practice developed from the SAS No. 82 risk categories. Surprisingly, participants not using the checklist identified more fraud situations than those with access to the list. Hackenbrack (1993) also incorporated a checklist of potential fraud indicators in an experimental setting and noted differing perceptions of the level of fraud risk for each indicator across the participants.

Continuing the research stream of auditor risk assessment, Asare and Wright (2004) examined the impact of alternative risk assessment approaches. The study found that auditors relying on the standard risk checklist generated lower risk assessments than those without a list suggesting use of a pre-defined checklist results in a less effective diagnosis of fraud. Second, the nature of the traditional attestation framework hinders timely and relevant assurance reporting (Vasarhelyi and Halper 1991). Performing an audit only once a year may result in fraudulent activity going undetected for up to a year or more and key stakeholders must wait an entire audit cycle to determine if the financial numbers released by an organization are correct in all material respects and that appropriate internal controls are in place to detect financial fraud. Critical decisions by investors, creditors, management, and others therefore may be determined from out-of-date information and conjecture rather than current, audited facts. This situation raises the risk of making less than optimal decisions.

Concerns over these weaknesses, in combination with the corporate frauds of the early part of this millennium, fostered the continuous assurance and auditing movement that many consider a viable approach to help identify potential audit concerns earlier in the process. In a joint study performed by the Canadian Institute of Chartered Accountants and American Institute of Certified Public Accountants, continuous auditing is defined as ‘‘a methodology for issuing audit reports simultaneously with, or a short period of time after, the occurrence of the relevant events’’(CICA/AICPA 1999). Over time, continuous assurance proponents have refined the related terminology and now recognize two distinct types: ‘‘continuous auditing’’ and ‘‘continuous assurance.’’ Alles et al. (2002) presented the distinction as follows:

> “[Continuous auditing] is best described as the application of modern information technologies to the standard audit products ... Continuous auditing is another step in the path of the evolution of the financial audit from manual to systems-based methods ... By contrast, continuous assurance sees continuous auditing as only a subset of a much wider range of new, nonstatutory products and services that will be made possible by these technologies.”

Consistent with this description, the remainder of this research study focuses on the technique of continuous auditing as a tool in an overall continuous assurance framework. Going forward in this manuscript, ‘CA’ will mutually represent the terms ‘continuous auditing’ and ‘continuous assurance”.

Continuous Auditing Movement: Yesterday, Today, and the Future

Many in the continuous auditing field consider the work of Vasarhelyi and Halper (1991) to be the groundbreaking study that initiated the movement. The authors developed a framework for a CA approach called the Continuous Process Auditing Methodology (CPAM). CPAM consists of three primary phases: measurement, monitoring, and analysis that together represent the foundation for the CA process. The measurement phase involves the identification of key metrics that will be compared to a set of standards; examples include account balances from financial statements, cost center expenses, and system configuration settings. Once metrics have been derived, the underlying data need to be examined in the second (monitoring) phase where the continuous audit system compares the metric data to the standards programmed into the system (referred to as "analytics") on a real or near real-time basis, hence the term "continuous." Any significant differences between the metrics and analytics automatically trigger alarms to notify auditors of any potential issues. In the final analysis phase, auditors review the nature of the alarms and investigate as they deem appropriate.

Vasarhelyi et al. (2004) referred to this process of auditing transactions nearer to the event on an exception basis as "analytic monitoring" and described new assurance technologies that can facilitate the CA tasks of observing events as they happen, triggering alarms on exceptions, drilling down to transactional detail, integrating data across multiple and distinct processes, and performing repeated tests at minimal cost. Some of the tools the authors discussed include continuity equations, tagging data accuracy, time series analysis, and data taps. Additional studies examined various advanced information technologies that may be applied to CA such as data warehouses (Rezaee et al. 2002), public cryptography (Ricart 2007), and text mining (Gray and Debreceney 2006). Kuhn and Sutton (2008) issued a call for more sophisticated analytics that apply artificial intelligence techniques or mathematical algorithms to improve the detection capabilities and effectiveness of continuous monitoring systems.

Over the past 20 years, two methodologies of CA emerged: embedded audit modules (EAM) and the monitoring control layer (MCL). EAM are software components built directly into an IS to monitor activities (Groomer and Murthy 1989). In contrast, the MCL approach consists of an external software application that interfaces with the system under audit in order to perform monitoring functions (Vasarhelyi and Halper 1991). EAM has received the greatest initial attention with a focus on building audit functionality into existing enterprise resource planning systems (Debreceny et al. 2003; Debreceny et al. 2005). Inherent limitations exist, however, that prevent broad implementation of EAM, particularly for external audit firms (Kuhn and Sutton 2008). For instance, implementing foreign code into an existing system may result in adverse affects to overall system performance, maintaining EAM code in all the connected systems of a large organization can be an arduous task, and legal issues may arise if the EAM code an external auditor operates and maintains in a client’s IS causes problems to operability of the system. For these and numerous other reasons, the MCL approach has gained traction. Two recent studies analyzed the design of CA systems for specific organizations, one for the monitoring of system configuration settings and controls (Alles et al. 2006) and another for the monitoring of transaction level data (Kuhn and Sutton 2006). Both studies argued that MCL rather than EAM is the most efficient and effective approach to CA.

Very few actual, full-scale CA systems have been developed to date. The majority have been small implementations for very specific business functions or basic internal controls and none by any external auditor firms. Vasarhelyi and Halper (1991) implemented an auditing procedure to examine the accounts payable function at Bell Labs as did Coderre (2006) for the Royal Canadian Mounted Police. Vasarhelyi (2005) and Kuhn and Sutton (2006) examined the Enron and WorldCom frauds, respectively, and demonstrate hypothetically how continuous assurance would have helped detect the fraudulent behavior. In both studies, the CA system proposed incorporates information external (i.e. from the environment) and internal (e.g. historical time series data) to the organization in order to develop and revise the standards to which the metrics are compared. For example, Kuhn and Sutton (2006) illustrated how "sweeping in" key financial ratio data from the telecommunications industry (e.g. operating expenses as a percentage of total revenue) and comparing to WorldCom ratios would have highlighted a significant deviation from both industry trends and WorldCom's own historical averages. Neither study provided a supporting theoretical foundation for the proposed system design. Alles et al. (2006) offered one of the most widespread implementations in their pilot project to monitor business process controls for the United States (U.S.) IT internal audit function of Siemens Corp. Of the approximately 200 audit action sheets (i.e. audit steps) for an audit of a Siemens SAP system, the project team determined about 25% could be automated with CA, saving significant time and money (Haardoefer 2005). Although Alles et al. (2006) detailed the architecture designed and rules implemented, once again the underlying theoretical design principles are not discussed.

The current SEC initiative for mandated continuous reporting of financial information and the focus of SOX on tighter internal controls over financial reporting and more transparency create an even stronger impetus to develop and implement CA functionality. CA can provide the external auditor a method of providing a more comprehensive, efficient, and effective audit by automating the testing of the full population of transactions for specific audit areas rather than evaluating only smaller samples due to the time constraints of manual processing. Boundless opportunities exist for researchers and practitioners alike to determine the optimal design, implementation approaches, and aftereffects of CA system use.

The environment in which organizations operate has become increasingly complex. Internal and external parties interact with one another creating a dynamic, connected network where the actions of one individual or organization can significantly affect the livelihood and behavior of others regardless of time and space dimensions. Auditors must be cognizant of this fact and consider the implications of external forces on their clients' operations. The remainder of this essay introduces the theories of complexity and complex adaptive systems (CAS) then discusses the techniques of simulation and agent-based modeling designed to analyze complex, organizational phenomena.

Complexity

Views and Theories of Complexity

Complexity, what does it truly mean? Depends who you ask. The Merriam-Webster Online Dictionary defines the word as "a whole made up of complicated or interrelated parts" (Merriam-Webster 2008). This understanding held true until about 15 years ago (Laurent and Koch 1999) when a new, deeper interpretation emerged distinguishing *complexity* from *complicated* (Reitsma 2003; Mikulecky 1999). A *complicated* system (and the original view of complexity) can be completely and accurately described regardless of the number of individual components (Reitsma 2003); "complication is a quantitative escalation of that which is theoretically reducible" (Chapman 1985). A *complex* system, on the other hand, cannot be fully understood by analyzing the components (Cilliers 1998); the whole is greater than the sum of its parts where the dynamics of real systems arise from traits of the individuals and their environment (Siebers and Aickelin 2007). The core of complexity science lies in understanding the indirect effects that arise from the interactions of system components.

Collectively, complexity science and research have taken the "anti-reductionist" approach to analyzing phenomena. Chapman (1985) argued that if the world can be explained in a reductionist manner then complexity is not qualitatively different from simplicity but merely quantitatively different. The main task for complexity science is to explain how relatively stable, aggregated, macroscopic patterns emerge from local interactions of numerous lower level entities (Srbljinovic and Skunca 2003). Over time, the anti-reductionist view of complexity has become the dominant approach but as Edmonds (1999) stated, many techniques under the banner of complexity inappropriately apply the concept of complexity when describing complicated or difficult systems.

Within the ever-expanding circle of anti-reductionist complexity advocates, a wide variety of opinions exist as to the particulars of complexity science. Complexity theories originated with the mathematical models of weather systems built by meteorologists (Lorenz 1993) and quickly spread to other natural science disciplines such as biology, chemistry, physics, etc. (Styhre 2002) and eventually to the social sciences. Due to the breadth of the disciplines employing complexity, many competing, contradictory, and confusing definitions of complexity arose and still exist today (Horgan 1995; Corning 1997; Richardson and Cilliers 2001; Parellada 2002; Manson 2003; Burnes 2005). Overviews of complexity research identify a myriad of complexity sciences (Matthews et al. 1999), a whole series of fields of study (Thrift 1999), and acknowledgement that no over-arching theory of complexity exists (Cilliers 1998).

Manson (2001) attempted to introduce some order by classifying the body of research (theories and models) into three, not mutually exclusive, groups of complexity research: algorithmic, deterministic, and aggregate. Algorithmic complexity theory consists of two aspects. The first relates to the calculation of the effort to solve a mathematical problem that is considered so complex that it is unsolvable; examples include spatial statistics and geographic information science (Manson 2003). The second views complexity as the simplest computational algorithm that can reproduce system behavior and facilitates condensing system component interactions into simple measures. This approach has been used to understand the limits on the thermodynamic cost of computations (Zurek 1989) and computing the minimum distance of a linear code (Vardy 1997).

Similar to algorithmic complexity, deterministic complexity relies on the use of mathematics but incorporates key aspects of both catastrophe and chaos theory. Catastrophe theory attends to systems that experience large, sudden changes in the state of the system as a result of a small change in an element and/or attribute. The system can go from stable to total disequilibrium very quickly. A commonly cited example is the threatened dog that either suddenly moves to attack or panics and flees (Simon 1996).

Chaos theory addresses dynamic systems that constantly transform themselves in an irreversible, evolutionary manner (Bechtold 1997; Haigh 2002), are sensitive to initial configuration conditions, yet possess an underlying order allowing short-term understanding and forecasting. However, the complex patterns of behavior (i.e. non-linearity) of these systems prevent long-term prediction as behavior is not proportional to the multiples (Fitzgerald and van Eijnatten 2002); chaotic systems are not subject to the laws of cause and effect (Burnes 2005; Pascale 1999). Aggregate complexity, the most commonly employed view popularized by the Santa Fe Institute (a private research organization focusing on complexity), takes the core principles of catastrophe and chaos theory even further than deterministic complexity by focusing on the interaction of system components rather than constructing models at the macro level. Note, a significant difference between complexity theories, particularly the aggregate view, and catastrophe and chaos theories, lies in the nature of the system with the latter concerned only with *closed*, isolated systems while the former on *open* ones. The aggregate view of complex systems truly epitomizes the anti-reductionist adage "the whole is greater than the sum of its parts." The varied and intertwined network of relationships among the system components extend beyond simple feedback into higher order, non-linear processes not amenable to modeling with traditional techniques (Constanza et al. 1993). This notion of aggregate complexity underlies the theory of complex adaptive systems that lies at the heart of this particular study. The next section delves further into CAS.

Complex Adaptive Systems

The theory of complex adaptive systems arose from the complexity theories spawned in the natural sciences to develop mathematical models of systems in nature. Although considered one stream of complexity research, many variations of the definition and key premises of CAS exist. A quote from John H. Holland, one of the original researchers in the area, best depicts the general principles underlying CAS:

> "A Complex Adaptive System is a dynamic network of many agents (which may represent cells, species, individuals, firms, nations) acting in parallel, constantly acting and reacting to what the other agents are doing. The control of a CAS tends to be highly dispersed and decentralized. If there is to be any coherent behavior in the system, it has to arise from competition and cooperation among the agents themselves. The overall behavior of the system is the result of a huge number of decisions made every moment by many individual agents" (Waldrop 1993).

CAS examples include economies, social systems, ecologies, cultures, politics, technologies, traffic, weather, etc. (Dooley 1997). In order to adequately comprehend and utilize a theory that spans such a wide array of disciplines with varied interpretations, Choi et al. (2001) developed a comprehensive framework of CAS elements and attributes depicted in Figure 1. The framework consists of three interacting and intertwined main foci each with a subset of additional components: 1) internal mechanisms, 2) co-evolution, and 3) environment. The remainder of this section reviews these foundational concepts and principles in more depth along with examples of related research from social science domains (predominately business) as they represent the supporting lens in which this study examines CAS.

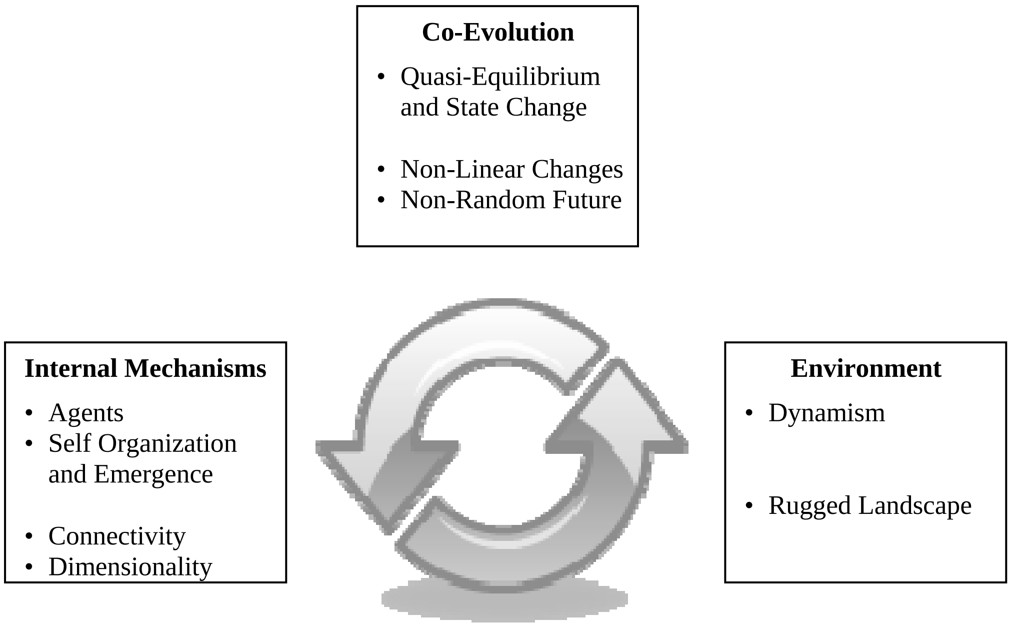


Figure 1: Underlying principles of complex adaptive systems. Adapted from Choi et al. (2001).

Internal Mechanisms

<u>Agents and schema</u>

Agents represent the building blocks of CAS and are semi-autonomous units that seek to maximize some measure of goodness, or fitness, by evolving over time where fitness corresponds to the general well-being of the system. Giddens defined agency as the ability to intervene meaningfully in the course of events. The term ‘agent’, to an extent, can be viewed synonymously with the term ‘actor’ commonly used in IS theories and research (see Actor Network Theory in Johnston and Gregor (2000). Therefore, by definition, a system must include agents that can impact the state of the system by their actions in order to be considered a CAS – a definitive characteristic differentiating CAS from complicated systems. Examples of agents in a social CAS include individuals inside organizations, organizations comprising a profession, or even a profession operating in a global marketplace. The latter two illustrate a network of multiple CASs functioning in concert. Defining agents and CASs thus depends entirely upon the perspective of the onlooker.

CAS agents interact with other agents, both within their own system as well as with the environment which may include other CASs and their respective agents, commonly referred to as meta-agents (Benbya and McKelvey 2006). The exchange of information and resources between agents facilitate the generation of schema (Schein 1992) defined as the norms, values, beliefs, and assumptions shared among the collective that dictate the manner in which agents interpret information and perform actions. Organizational leaders often declare formalized mission statements, create codes of conduct, ethics statements, etc. that represent core values and guide the behavior of agents, in particular, the interaction between employees and other stakeholders (e.g. customers, vendors, and other related parties). Uhl-Bien et al. (2007) developed Complexity Leadership Theory as a leadership paradigm that focuses on enabling the learning, creative, and adaptive capacity for knowledge-producing organizations.

Within the bounds of these “rules of behavior” and shared values, agents strive to increase the fitness of their system, both locally and globally. The actions of agents can result in non-linear impacts to the local system and network of systems depending on the connectedness of the system(s); a more connected system will generally experience larger ripple effects throughout as agents interact in a dynamic fashion. Complex system behavior, therefore, can occur when multiple non-linear processes interact (Choi et al. 2001).

Self-organization and emergence

Self-organization refers to the emergence of a pattern of order from a simple set of rules governing agent behavior in a connected network without the intervention of a central controller (Anderson 1999; Luoma 2006; Mason 2007) that allows limited chaos (Frederick 1998). The self-organization process occurs from the bottom up through the interactions and inter-relationships of agents at the lowest level creating new structures or behaviors unintentionally. These emergent phenomena seem to have a life of their own with their own rules, laws, and possibilities (Goldstein 1994; Zimmerman et al.1998) and can be observed holistically as ordered patterns that emerge from aggregate individual behavior (Fuller and Moran 2001) that generate infinite variety and unpredictability.

> "One consequence of emergent complexity is that you cannot see the end from the beginning … it is less comforting to put oneself at the mercy of this process with the foreknowledge that we cannot predict the shape that the future will take. Emergent complexity creates not one future but many ... Design for emergence never assumes that a particular input will produce a particular output [but the] design creates probabilistic occurrences that take place within the domain of focus. Period. Greater precision is neither sought nor possible." (Pascale 1999)

Reynolds (1987) presented the phenomenon of flocking birds as an illustration of the self-organization process. The flocking pattern (i.e. the new structure) occurs not because of a predetermined plan or unilateral control by the lead bird. The pattern emerges from the actions of individual birds (agents) acting upon three simple rules based on local information: each bird 1) keeps a minimum distance from the other birds, 2) flies at the same speed as other birds, and 3) moves towards the center of the flock. The individual birds behave according to their own local rules of interaction and a self-organized, coherent pattern emerges for the system as a whole.

From an organizational perspective, Burnes (2005) offered insight into the usefulness of CAS theory:

> "By reducing the workings of the natural world to mathematical models and simple order-generating rules, complexity theories have an attractive elegance, especially for those of us who seek to understand the complexity of the organizational world."

Individual managers cannot predict or plan long-term outcomes (Wilkinson and Young 1998; Frederick 1998; Kelly and Allison 1999; Mason 2007), but can adapt simple rules to manage movement of the aggregate (i.e. the system) between stability and chaos (Lewin 1993; Mason 2007). The aforementioned examples of mission statements, codes of conduct, and ethics statements embody the underlying principles of simple rules that guide agent behavior, rather than directly controlling the agent interaction. Examples of self-organization and emergence in the business setting include development of new strategies (Conner 1998) and marketing tactics (Forrest and Mizerski 1996), self-directed teams (Gault and Jaccaci 1996), growth of strategic alliances (Wilkinson and Young 1998), and investor herd behavior that frequently is attributed to irrationality but in actuality results from rational, local interactions and their non-linear consequences (Andreoni and Miller 1995).

Connectivity

A key premise of CAS theory involves the concept of connectivity – the linkages of agents inside a system with each other and to neighboring systems. Different elements (agents, meta-agents, other CASs) continuously interact producing intertwined reactions nearly impossible to anticipate or trace afterwards (Luoma 2006). As the number of agents increase, the volume and layers of relationships, both direct and indirect, grow exponentially to such a complex state that differentiating between cause and effect becomes too onerous.

The theory of reductionism asserts that complex data and phenomena can be explained by a process of reducing to simpler terms and analyzing the components independently to gain insight into the whole. Bettis and Prahalad (1995) and Dent (1999) argued the reductionist approach fails to effectively provide knowledge of the whole when studying organizations due to their complex nature. Viewing an organization as a CAS requires a holistic focus on the system in aggregate, not individual agents or pockets of agents. The performance of the whole cannot be enhanced by optimizing the performance of each individual agent nor should the problem with one agent be examined in isolation from the system (Luoma 2006). A wider context must always be at the forefront promoting examination of the unit in the broader perspective of agent relationships, dependencies, and downstream effects. Analysis of these connected relationships in a CAS offers a distinct opportunity to make the most of the agent diversity inherent in a system thus facilitating richer interpretations of the environment and fostering creative solutions.

Dimensionality

Dooley & Van de Ven (1999) defined the dimensionality of a CAS as the degrees of freedom that individual agents within the system have to enact behavior in a somewhat autonomous fashion. Controls such as rules and regulations, budgets, limits of authority, etc. constrain agent behavior and thus reduce dimensionality and change the complexity of the system's aggregate behavior (Stacey 1995; Thietart and Forgues 1995; Glass 1996). The system becomes predictable, stable, and less flexible. CAS researchers refer to these constraints as negative feedback in the sense the system works to maintain some stable condition where deviations lead to corrective action. When agents are allowed more autonomy to make decisions locally, outcomes then have the ability to emerge and cascade throughout the system possibly leading to the innovation and competitive advantage. This emergence reflects the concept of positive feedback where the system works to reinforce the phenomena increasing the overall effect. Increased dimensionality thrives on positive feedback. As an example, two scientists working together potentially can advance more rapidly than if in isolation due to the opportunity to leverage the unique perspectives, background, and knowledge each individual offers.

A fine balance must exist between negative and positive feedback to maintain system functionality. Pascale (1999) identified instances in the business world where either negative or positive feedback systems went awry. Both IT&T and Sunbeam implemented strict cost cutting measures that ultimately stifled imagination and creative energy leaving the companies in a state of stagnation and unable to cope with changes to the business environment. ValuJet, on the other hand, focused on expansive growth with little to no attention on appropriate controls – operational, safety, reliability, and service standards. The company achieved profitability faster than any other airline in history, apparently at the expense of implementing controls. The fatal crash on May 11, 1996 that killed all 110 people aboard culminated a chain of incidents and accidents that eventually led to grounding by the Federal Aviation Administration and a subsequent reverse merger with the significantly smaller Airways Corporation, parent of AirTran Airways. The successes of ValuJet amplified one another to the point where the company could not maintain an adequate supporting infrastructure, resulting in total system failure.

Environment

A complex system owes its existence to relationships with its environment, defined as anything outside of the system (Manson 2001). The system passes information and energy throughout itself and the resulting actions/interactions of agents subsequently produces an outflow to the environment that cycles back to the system; this process continues endlessly as long as the system exists. The environment, in relation to a CAS, depends entirely upon the scale of analysis chosen. For a CAS defined as the supply chain function of a manufacturing organization, internal agents may consist of the employees in the production planning, inventory management, and warehouse departments that interact with other potential internal CASs such as the purchasing and accounting departments and even executive management. Externally, meta-agents may include customers, suppliers, and transportation vendors. An expanded scale might consider the manufacturing organization, in aggregate, as the system which interacts with numerous other meta-agents in addition to the ones that interface with the supply function such as regulatory agencies, corporate shareholders, taxing authorities, etc. Regardless of scale chosen, (Choi et al.2001) characterize environments as dynamic and rugged.

<u>Dynamism</u>

The Merriam-Webster Online Dictionary defines dynamism as “a theory that all phenomena can be explained as manifestations of force” (Merriam-Webster 2008). Complex systems experience many sources of force, internally and externally. While a CAS attempts to emerge through agent interaction and proactively influence other neighboring systems, the external environment simultaneously exerts pressure on the CAS causing a reaction that, in turn, affects the environment. CAS theory posits that a system both reacts to and creates its environment through experiences of positive and negative feedback (Choi et al. 2001).

The constantly changing relationships among agents, between systems, and with the environment result in changes to the schema organizations incorporate into their day-to-day interpretations of reality and thus their behavior. The emergence of the Internet offers an excellent example of a dynamic change in the environment. The Internet delivered broad-based changes to the organization of economic activity so profound to warrant the title of a revolution; the declining cost of information led to increased business traffic, greater information access, personal autonomy in local decisions, and ultimately, greater dispersion of economic activity (Feldman 2002). A number of simultaneous developments resulted in positive feedback that reinforced and strengthened the Internet movement: expanding personal computer use, technological advances in hardware and software, increased awareness by users, improvements in telecommunications, falling technology prices, etc. (Luoma 2006).

As the Internet fever began to take hold, new competitors emerged to challenge traditional brick and mortar organizations. Barnes and Noble operated the largest chain of bookstores in the U.S. In 1997, the company surpassed the $2 billion revenue mark yet encountered a new competitive threat in Amazon.com, a two-year old online bookseller with 1997 revenues of $148 million, an increase of 840% over the previous year, and which subsequently reported 1998 revenues of $610 million. Barnes and Noble saw the writing on the wall: the Internet would upend the traditional bookselling business model. In response to the changing environment, Barnes and Noble launched an online platform to sell books and eventually developed an in-stock inventory of over 750,000 titles ready for immediate delivery and eight million new, out-of-print, and rare books – both of which the company claimed were the largest in the industry (Answers.com 2007). The experiences of Barnes and Noble and many others during the early years of the Internet demonstrate the interaction of numerous CASs and the broader effects of agent actions in a dynamic environment.

Rugged landscape

By nature, the eventual outcome of agent interaction is unknown and unpredictable. CAS researchers represent the potential states that a system can attain in a dynamic environment as a rugged landscape with many hills and valleys (Kauffman 1997; Ethiraj and Levinthal 2004) and is commonly referred to as a "fitness landscape." The highest point in the landscape symbolizes the optimal state of the system where the well-being of the system reaches its greatest level. However, many system components (agents) operate in a tightly, coupled manner each contributing to the overall direction of the system. The optimal state becomes difficult to locate as many local optima exist for the individual components; choices at the local level do not always result in performance gains at the organization level. Further exasperating the complexity of a CAS, environmental pressures force the landscape to change eliciting agents to exploit existing knowledge and explore new knowledge (March and Heath 1994) necessary to overcome the uncertainty imposed by the environment and ensure survivability (Choi et al.2001).

Choi et al. (2001) discussed the inter-dependencies of agents and the overall state of a CAS in the context of a supply chain network. The authors explained that incorporating modular design in the automotive supply chain process reduced the number of peaks in the rugged landscape creating a condition more conducive to overall system optimization. For example, as opposed to the manufacture of individual parts, the automotive industry reorganized the entire supply chain process to a point where first-tier suppliers produce entire modules or subsystems (e.g. complete engines, steering systems, etc.) minimizing the cost of coordination across the entire supply network.

Co-evolution

Co-evolution directly relates to the concept of connectivity in that multiple systems and/or sub-systems emerge *together* because "there is feedback among the systems in terms of competition or co-operation and utilization of the same limited resources" (Zimmerman et al.1998). Symbiotic relationships exist as different parties (agents and neighboring systems) depend upon and interact with each other. The environment imposes changes on its members who react thus changing themselves and consequently changing the environment. Therefore, co-evolution occurs when system members are forced to adapt continually to the changing context wrought by others' strategies in order to remain relatively fit (Van Valen 1973; Kim and Kaplan 2006).

In a business context, the increasing prevalence of partnerships and alliances in a traditionally competitive environment indicates a general shift of practice and strategy towards co-evolution (Luoma 2006). Many organizations seek to expand operations into foreign markets not through acquisitions and mergers but through mutual agreements in order to leverage the knowledge and resources of each party. Grant and Baden-Fuller (2004) presented a theory of strategic alliances that focuses on alliances as a strategic tool to access knowledge resources of other organizations rather than through acquisition of the organization possessing the knowledge. Alliances contribute to the efficiency in the application of knowledge by improving the integration of knowledge into the production of complex goods and services and increasing the efficiency of knowledge utilization. The efficiency advantages of alliances are enhanced when uncertainty exists in the environment.

Quasi-equilibrium and state change

Unlike chaos theory that focuses on the discovery of unpredictable behavior, complexity science and CAS theory strives to explain how order emerges from self-organizing agent interaction (Kauffman 1993; Holland 1995). Within the apparent randomness of a CAS, order can be unmasked to predict broad behavior in the short-term, not at the individual agent level but in the aggregate. Mainzer (1997) and McKelvey (2004) referred to complexity as an order-creation science.

Systems under complexity science can exist or vacillate between any of three states– stable, chaotic, and one in between (Lewin 1992). Many complexity researchers label the middle state as the “edge of chaos” (Lewin 1992; Kauffman 1995) where creativity, growth, and useful self-organization are at their optimal (Frederick 1998). A CAS maintains this quasi-equilibrium state, constantly balancing between complete order and incomplete disorder (Goldstein 1994). Highly ordered systems exhibit too much rigidity to effectively respond to environmental changes while highly chaotic systems cannot maintain any semblance of consistency and eventually collapse from excessive disruption. The “poised” systems that lie in the middle “may have special relevance to evolution because they seem to have the optimal capacity for evolving” (Kauffman 1991). These systems adhere to the principle of maximum entropy production where the system moves towards the brink of complete disorder (entropy) but never quite falls over the edge as new energy flows into the system forcing redirection back to a quasi-equilibrium state.

The Luoma (2006) discussion of complexity and management development presented the view that disequilibrium and disorder should not be seen as negative organizational attributes. Attempts to entirely eliminate disorder suppress a system's ability to self-organize (Stumpf 1995). Management should exert some control but allow an organization to exploit the innate ability to spontaneously develop behavior that most effectively moves the whole in a given direction. Weick (1979) supported this view by noting managers tend to get in the way of activities that have their own self-regulation, form, and self-correcting tendencies.

As another example of quasi-equilibrium and state change, an organization may reach a peak on the fitness landscape that represents a "Golden Age" of high prominence and profitability that will eventually fade away, forcing the organization to traverse down the peak into a "valley of darkness" before emerging on to another, potentially higher peak on the fitness landscape. Apple Inc. offers a prime instance of an organization negotiating the fitness landscape, teetering on the edge of chaos only to work its way to higher peak. Originally created as Apple Computer (a.k.a. "Apple") in 1976, a designer of personal computers, Apple entered a stage of its corporate life during the period of 1989-1991 earning record profits and obtaining a reputation as a quality manufacturer of desktop and portable computers. The trade magazine MacAddict referred to this time period as "the first golden age" of the Macintosh (iPodGames 2008). However, the good times could not and would not last. Microsoft's Windows operating system continued to develop and established a stranglehold on the personal computer market. Apple languished for nearly a decade until the return of Steve Jobs, one of the co-founders and generally regarded as the catalyst behind the early successes of Apple. Introduction of the iMac in 1998 and successful ventures into other consumer electronic products such as the iPod and iPhone, represent the beginning of what is now being called “The New Golden Age of Apple” (Brannan 2007) that will likely continue, for a time, before Apple must brave the valley once again.

Non-linear changes

The level of sensitive dependence on initial conditions delineates a CAS from a stable system (Briggs and Peat 1999; Phillips and Kim 1996). Generally, small changes in a stable system result in small effects, while large changes produce large effects. Changes in a CAS generate unpredictable effects; small changes can grow exponentially with each interaction through the system and large changes may languish or disintegrate altogether through agent inattention. Gibson (1997) and Wheatley (1996) advocated management application of small “nudges” to guide an event or process rather than dramatic actions intended to control. As in many instances in business, timing is everything. The right kind of nudge at the correct time can lead, through positive feedback, to major change (Nilson 1995).

Mason (2007) presented the first-mover advantage as an illustration of non-linear change in a business context. Sensitive dependence on initial conditions and positive feedback create a “flywheel affect” that reinforces early success, providing a significant advantage over the long term. A number of studies discounted first-mover advantage as a myth (Suarez and Lanzolla 2005; Pfeffer and Sutton 2006) yet others contended the opposite:

> “To gain advantage, first movers must capitalize on the opportunities that come with being a pioneer while at the same time manage the threats that arise. The bottom line: Being first in a market is only an advantage when you do something with it” (Finkelstein 2007).

Opinions supporting the validity of first-mover advantage epitomize core concepts of CAS theory. First-mover advantage occurs as a result of non-linear relationships and positive feedback yet the interaction between agents (pioneers, competitors, and the environment) results in unpredictable outcomes – whether or not a pioneer can maintain the advantage through proactive and reactive action.

Non-random future

Although the nature of a CAS prevents exact prediction of future actions and outcomes, distinct patterns of behavior exist underneath apparent randomness allowing examination and general predictive ability. Small changes may lead to drastically different future paths; however, the same characteristic pattern of behavior emerges despite the change (Choi et al.2001).

Recent work in financial economics highlighted patterns of non-random behavior that result in varied outcomes. Baker et al. (Baker et al.2002) attempted to solve the "dividend puzzle" by examining how managers determine dividend policy. Calling upon earlier work on habitual behavior (Waller Jr. 1989; Frankfurter and Lane 1984), the authors concluded that various market imperfections and frictions affect organizations differently; therefore, dividend policy differs from organization to organization and models should consider competing frictions on an organization-specific basis. Underlying this work, Waller (1989) suggested the concept of habit (non-reflective behavior) may be a useful tool for institutional policy analysis and can "be a fatal blow to work that is based on rational behavior" (Baker et al.2002). Habits reflect cultural and societal norms/standards that may contradict rational economic behavior. Further, Frankfurter and Lane (1984) asserted habitual behavior causes problems for models attempting to explain dividend policy assuming rational behavior and claimed socioeconomic consequences of modern corporate evolution best explain dividend behavior. This stream of research, although not explicitly stated, exhibits core CAS principles.

This section presented the elements of the CAS theoretical foundation presented by Choi et al. (2001) and provided examples from academic research, the business environment, and natural systems in order to explain the fundamentals underlying complex adaptive systems theory. Concluding this section, a quote from Steve Miller, a member of Royal Dutch/Shell’s committee of managing directors, epitomizes the impact of CAS theory on business strategy in today’s knowledge-based, information-oriented economy in contrast to the traditional management views of the past:

> “Today, if you’re going to have a successful company, you have to recognize that the top can’t possibly have all the answers. The leaders provide the vision and are the context setters. But the actual solutions about how best to meet the challenges of the moment, those thousands of strategic challenges encountered every day, have to be made by the people closest to the action.” (Pascale 1999)

The goal of this dissertation to develop an organizational theory with complexity theory at the core will be applied to the continuous assurance of corporate financial information utilizing computer simulation. As such, the use of CAS in both accounting and IS research warrants consideration. The remainder of the discussion on CAS examines related research in each the accounting and IS disciplines then discusses opportunities for future research.

Related Research in Accounting

Ballas and Theoharakis (2003) explored the diversity in accounting research by conducting a survey of accounting faculty on their perceptions of the most prominent journals that publish accounting-related research. An extensive review of the 58 journals listed in that study (see Appendix A) revealed only three articles that specifically discussed and incorporated complexity and CAS theory into the core of the study. Mouck (1998) and (2000) explored the challenges that chaos theory and complexity theory pose to the methodological views of capital markets research in accounting, the dominant paradigm in North America. The assumptions of neoclassical economics such as rational behavior, linearity, and predictability frankly fall short of accurately depicting reality and thus, leading economics researchers at the Santa Fe Institute have turned to CAS theory (Mouck 2000). More recently, Thrane (2007) examined the role and practice of accounting in dynamic and complex business networks, specifically how management accounting affects and effects change on complex inter-organizational systems. The author concluded management accounting in complex evolving inter-organizational systems acts as a source of instability rather than stability and as a source of emergent, unintended order rather than planned or institutional change. This view represents a stark contrast to the commonly-held belief that accounting rules and principles provide structure, consistency, and predictability.

The literature review also identified a few studies ancillary to CAS and/or published in journals other than the original 52 examined. Continuing the theme of unpredictability, Gouws and Lucow (2000) claimed traditional financial analysis approaches are no longer valid in a constantly changing business environment and presented a dynamic balance model to establish whether entities are able to adapt, survive, and prosper. Clarke (2005) drew a corollary between the key concepts of CAS theory and corporate governance and reviewed details of the Enron fraud under this lens. In the author's opinion, corporate governance "must no longer confine its analysis to the relationship between managers, boards, and shareholders" as the dynamic complexity of corporate governance in a connected world requires new, fresh theoretical perspectives. In the final related article identified during the literature review for this study, Painter-Morland (2006) analyzed the central assumptions of the current view of accountability in business ethics and offered a re-conceptualized version based on CAS theory. Under this approach, accountability can be viewed as a relational responsiveness towards stakeholders. The shared norms and values that organize and guide business behavior develop and emerge on a contingent basis as colleagues, clients, and competitors interact. As the author states, "the orderliness of business life is a reflection of the fluid internal logic of business as a system of dynamic functional relationships."

Although few in number, the aforementioned articles indicate CAS theory is slowly making inroads into accounting research and literature. Traditionally, the accounting discipline has borrowed theories from other areas such as management, economics, psychology, sociology, etc. Management science has been examining large-scale complex systems for over two decades as evidenced by early works based on complexity science published in *Management Science* (Florian et al.1980; Tilanus 1981; Bitran and Yanasse 1982) and the inclusion of CAS theory in the Astley and Van de Ven (1983) discussion of central perspectives and debates in organization theory. The movement has persevered (see 1999 special issue on complexity in *Organization Science*) and continues today (see 2007 special issue on complexity in *Management Science*). As par for the course, I suspect accounting researchers will begin to see the merits of examining accounting phenomena holistically as a part of and affected by dynamic, open systems that constantly ebb and flow as a result of the localized behavior of connected agents.

Related Research in IS

The IS discipline has embraced the key tenets of complexity theory to a far greater extent than accounting. As IS permeate all scientific disciplines, natural and social alike, IS researchers possess a diverse background and varied worldview. For many, complexity research is not a new approach but a staple in their research methodology repertoire. To determine the extent of usage of complexity and CAS in IS research, I first examined the publications from the 30 highest ranking journals for the IS research community as denoted by the Association for Information Systems (AIS) on their website (see Appendix B). Per the website, the association is "a professional organization whose purpose is to serve as the premier global organization for academics specializing in Information Systems" (AIS 2008) and sponsors the two most prominent IS conferences – The International Conference on Information Systems (ICIS) and The Americas Conference on Information Systems (AMCIS). As such, the AIS journal rankings hold sufficient credibility as a source of reference to identify leading journals for the discipline.

Review of the 30 journals identified 10 articles specifically incorporating complexity and CAS theory into the structure of the study. In this group of studies, two broad themes emerge linking two sets of articles with a remainder that addressed unique aspects of complexity and IS. First, four of the articles examined how complexity can aid in understanding the dynamic, unpredictable business environment organizations face today. In the earliest work found, Fulkerson (1997) discussed the change in the nature of the business environment from one where organizations could capture market share and achieve greater levels of profits by merely producing larger volumes of products for mass market consumption to an environment where success requires adopting methods to manage both anticipated and unanticipated change. Drawing upon core principles of complexity theory, the study described how technologies such as genetic algorithms and autonomous agents can enable mass customization strategy to respond to changes in the market. Later studies continued the call for organizations to alter their views on business and technology to think holistically, dynamically, and in a networked/connected manner (Atwater and Pittman 2006; Denning 2007) in order to survive and prosper in an ever-changing environment (Merali 2002). The second group of associated studies investigated enterprise system integration and evolution in general (Kishore et al.2006) in healthcare systems (Sutherland and van den Heuvel 2002; Tan et al.2005), and in the insurance industry (Sutherland and van den Heuvel 2002). As a whole, the research concluded the semi-autonomous, networked agent view of CAS can help organizations integrate enterprise systems in order to achieve the levels of agility and responsiveness necessary to compete in today's dynamic environment. Finally, the remaining articles relied on complexity and CAS to explore the problem of information overload in face-to-face electronic meetings using group support systems (Grisé and Gallupe 1999); demonstrate the socio-technical complexity of IS standardization (Hanseth et al. 2006); and analyze pricing decisions, piracy, and protective technologies in the software industry (Khouja et al. 2007).

Outside the group of journals above, one can also see the IS discipline starting to embrace complexity and CAS through publications and calls for research. In 2006, *Information Technology & People* dedicated a special issue to complexity research in IS with the introduction titled "Taking complexity seriously in IS research" written by Jacucci et al. (2006). The authors asserted that complexity is an important topic for IS research and practice for at least three reasons. First, the growing number of systems and inter-connectivity pose challenges to current software development methods and practices as they are no longer able to “scale to manage these increasingly complex, globally distributed systems at reasonable cost or project risk” (British Computer Society 2004). Second, changing market demands and increased workforce diversity force organizations to improve their ability to respond quickly and adapt to competitive markets. Third, the ever-flattening of the business world due to globalization results in an environment where local actions proliferate to the macro level with undeterminable and unexpected side effects. Courtney et al. (2008) concurred. In the inaugural issue of the *International Journal of Information Technologies and the Systems Approach* (IJITSA), the authors outlined their thoughts on how complexity can be used to inform IS research. They unequivocally stated the need for a complex view of organizations and their IS as well as noting the stage is set for such a paradigm shift:

> “… the science is there; the systems are there; the computational capacity is there. All that is lacking is the consciousness to apply them.”

Simulation and Agent-Based Modeling

As discussed in the previous sections of this chapter, certain phenomena in nature and society are complex, dynamic, and impossible to break down into deterministic cause and effect relationships; no definable end point or optimal solution exists. Gaining insight, understanding, and knowledge of these events or happenings requires robust tools and technologies. Analytical models fail to adequately account for the indirect effects of CAS agent interactions. Computer-based simulation, on the other hand, offers the capacity and power to mimic real-world system behavior and observe changes in system states at any time rather than merely predicting the output of a system based on a set of inputs (Siebers and Aickelin 2007). The purpose of simulation is to better understand the inner workings of a system and/or to predict performance. Siebers and Aickelin (2007) compared simulation to an artificial white-room that allows one to gain insight but also to test new theories and practices without disrupting the actual system's operation. Troitzsch (2000) stated that if the theory framed for a particular system holds and the theory has been adequately translated into a computer model, then the simulation can assist in determining 1) what kind of behavior a target system will display in the future and 2) which state the target system will reach in the future. Such predictions involve analyzing trends rather than generating precise and absolute predictions of system performance (Siebers and Aickelin 2007); Keen and Sol (2007) referred to this as "rehearsing the future." Simulation, therefore, should be viewed as a decision support tool that requires consideration of the context of the real system before moving forward to implement steps intended to alter the system's direction and influence future state changes.

Agent-based modeling (ABM) represents one type of simulation modeling and provides the capability to explore the non-linear, adaptive interactions inherent to a CAS (Siebers and Aickelin 2007; Srbljinovic and Skunca 2003). The researcher specifies the rules of behavior at the micro-level for the individual agents and the interactions between agents. Structures then emerge at the macro-level due to the actions of these agents and their interactions with each other and the environment. The consequences at the macro-level that result from ABM many times are not obvious or expected. This discovered knowledge allows the interested party to identify potential system states that may not have been considered otherwise, thus enhancing the effectiveness of the decision making process. Cederman (1997) noted the following as some advantages of ABM: 1) the possibility of modeling fluid or turbulent social conditions when modeled agents and their identities are not fixed or given, but susceptible to changes that may include birth or death of individual agents, as well as adaptation of their behavior; 2) the possibility of modeling boundedly rational agents, making decisions and acting in conditions of incomplete knowledge and information; and 3) the possibility of modeling processes out of equilibrium.

Complexity researchers began using ABM in earnest in the 1990's (Epstein and Axtell 1996) and the approach has become a well-established simulation modeling tool in academia (Siebers and Aickelin 2007). As an example, an entire specialty in economics called Agent-Based Computational Economics (ACE) developed from the CAS movement to computationally study economies modeled as evolving systems of autonomous interacting agents (Tesfatsion 2001). ACE attempts to understand why certain global regularities evolved and continue on in decentralized market economies despite the absence of a central controller (e.g. trade networks, currencies, and market protocols) and to examine the effects of alternative socio-economic structures on individual behavior and social welfare. ABM is also quickly becoming more commonplace in practice to solve real business problems such as examining customer behavior in a supermarket based on differing configurations of products in the store layout (Casti 1997) and stakeholder (investors, market makers, and issuers) reactions to proposed changes to the tick size on the NASDAQ stock exchange (Bonabeau 2002).

The underlying concepts of complexity science, CAS, and ABM hold true for ABM itself as a scientific tool. As current ABM researchers model complex phenomena, present their findings, and solve real world problems, more individuals will be convinced of the merits of ABM and begin to incorporate the approach in their own work which will increase exposure exponentially through a myriad of academic and professional networks. I fully expect ABM to continue a pattern of emergent behavior and growth for the foreseeable future.

Summary

In Chapter 2, I present the supporting base for this dissertation study. First, the discussion on continuous assurance and CA sets the stage for the practical setting of the study by detailing the origins and progression of the continuous audit movement. Next, the explanation of complexity science and complex adaptive systems provides the theoretical foundation upon which the study relies and examines related research in accounting and information systems. Finally, the section on simulation and agent-based modeling introduces the technical approach that will be applied. I utilize ABM in the validation of the theory developed in Chapter 3 and as an illustrative decision support tool for the auditor’s going concern opinion in Chapter 4.

CHAPTER THREE: A COMPLEX ADAPTIVE INQUIRING ORGANIZATION

Introduction

Today, organizations face a completely new business environment than generations past – one that is complex, service-oriented, connected, global, in a constant state of flux, and built on individual and organizational knowledge. The knowledge-based view of the firm (Kogut and Zander 1992) contends that a valuable, rare, and inimitable resource (i.e. knowledge) can contribute to the competitive advantage of the individual/organization possessing it and performance will be reflective of that knowledge (Rodgers et al. 2008). In this new knowledge-based economy, interactions with customers and clients impact long-term success more so than ever and survival depends on the ability to explore new knowledge and maintain existing knowledge (Hall and Paradice 2005). However, decision makers' traditional knowledge sources and endowments may not be sufficient (Rodgers et al. 2008) to address problems that are more socially-oriented and are thus considered semi- or unstructured. Commonly referred to as "wicked" problems (Rittel and Webber 1973) or ill-structured (Mason and Mitroff 1973), these decision-making scenarios are highly uncertain, difficult to define, inextricably connected to their environment, and possess irreversible solutions. Such environments not only require organizations to be able to make decisions effectively and rapidly, but also be able to create knowledge and learn (Hall and Paradice 2005).

Courtney (2001) called for a new decision-making paradigm for DSS to adequately address wicked problems in complex contexts. Drawing upon unbounded systems thinking (Mitroff and Linstone 1993), Singerian inquiring systems (Churchman 1971), and the notion of “inquiring organizations” (Courtney et al.1998) the author introduced an alternative approach to the technical perspective in DSS research. The new paradigm brings in the perspectives of many stakeholders in order to provide greater insight into the nature of the problem, relationships among the connected elements in the wicked system, possible solutions, and downstream effects of implementing various solutions. Knowledge from any discipline or profession may be included as needed to assist in understanding the problem. Courtney (2001) illustrated the proposed DSS paradigm with an analysis of the decision-making process for planning and constructing a city’s infrastructure. As noted, the process almost defies analysis with the countless internal departments and external parties involved, yet is ongoing and vital to every city in the world. These types of problems require and need DSS research that truly reflects the nature of the decision-making environment.

Recently, a stream of research has materialized examining information systems development (ISD) from a social constructivist lens where information systems and their requirements emerge from the interaction of multiple stakeholders’ views and knowledge is created (Carugati 2008; Richardson and Courtney 2004; Markus, Majchrzak, and Gasser 2002; Avison et al. 1998). Carugati (2008) extends previous ISD literature by integrating Churchman’s (1971) inquiring systems into a framework that maps various ISD activities on systems for the creation of knowledge. Applying the framework, the author examines ISD activity and knowledge creation at the micro-level where actors (i.e. agents) are involved in a continuous back-and-forth exercise between long-term planning (i.e. interacting with external networks) and day-to-day work activities (i.e. acting at the local level) concluding that adding an epistemological view of micro-level ISD activities allows for a better understanding of the situation, better prediction of success or failure, and ultimately better management of the ISD process.

This chapter answers Courtney’s (2001) call for DSS research that considers the complexity and connectedness of the real business environment that organizations face in a knowledge-based economy by examining the micro-level (local) activities of network agents. The next section briefly revisits the basic concepts of complex adaptive systems followed by a discussion of Churchman’s (1971) SIS that, together, provide the theoretical foundation for the notion of a Complex Adaptive Inquiring Organization that underlies the system design principles developed for knowledge management and decision support. I then conclude with a discussion of the theory’s implications for KMS and DSS research.

Complex Adaptive Systems

Life on planet Earth, in general, is becoming ever more connected and complex. For instance, Wal-Mart Stores, Fortune’s 2008 largest U.S. corporation, operates more than 4,000 facilities in the U.S. and over 2,800 in Argentina, Brazil, Canada, China, Costa Rica, El Salvador, Guatemala, Honduras, Japan, Mexico, Nicaragua, Puerto Rico, and the United Kingdom. Their famed global distribution network works with more than 61,000 suppliers in over 55 countries around the world through a global procurement office and the company demands that their business partners meet specific environmental, social, and quality standards. Wal-Mart’s corporate beliefs and values filter down throughout their massive business chain to the point of impacting the day-to-day lives of field workers picking cocoa beans in Nicaragua. Since 1998, Wal-Mart Centroamerica has supported the region’s social and economic growth by partnering with local farmers to learn new agriculture techniques through the Tierra Fertil program so they can produce high quality products for retail markets. Currently, the program helps 2,045 Costa Rican, 2,850 Nicaraguan, 155 Honduran, 16 Salvadoran, and 109 Guatemalan producers. The Wal-Mart global connection illustrates how advances in technology have and will continue to facilitate a flattening of the world, reducing time and space constraints. A true global community is emerging where the actions of individuals and organizations in one corner of the world affect many others residing in different location around the globe.

In order to make sense of such a connected world, researchers and organizational decision makers are increasingly turning towards complexity theory and complex adaptive systems theory. As discussed at great length in the previous chapter, the essence of CAS theory lies in the behavior of a dynamic network of many agents each following a simple set of rules to anticipate and respond to the actions of other agents and the environment. The actions of these agents at the micro level impact overall system performance at the macro level, but the non-linear relationships of agents to each other, with agents from other systems, and with the environment prevent long-term prediction of a system's state. However, distinct patterns of behavior exist underneath the apparent randomness that allow examination and general, short-term predictive ability. CAS theory offers the ability to view social phenomena in a more holistic manner that is more reflective of reality than traditional, linear models.

Churchman's Singerian Inquiring System

Organizations must learn, adapt, and manage knowledge in order to succeed in today's dynamic environment (Richardson et al.2001). To accomplish that, organizations must be inquisitive. Churchman (1971) defined inquiry as an activity that produces knowledge and stated one must have the capacity and ability to adjust behavior to changing circumstances in order to be considered knowledgeable. In his seminal text *The Design of Inquiring Systems*, Churchman examined the epistemologies of five schools of philosophy from a general systems theory perspective to determine their suitability as the foundation for the design of inquiring computer systems. These inquiring systems have been proposed as theoretical models for the creation of learning or inquiring organizations (Courtney et al.1998; Courtney 2001; Linden et al.2007). As such, Linden et al. (2007) in their review of Churchman's inquiring systems believed that the inquirers can form the basis for the design of organizational KMS and encouraged their use to further develop IS as a discipline. This dissertation study answers that call.

Churchman named one of the inquiring systems after the philosophical beliefs of his mentor, Edgar Singer. The Singerian Inquiring System is above all teleological, a grand teleology with an ethical base (Churchman 1971). The SIS aspires to reach a highly idealistic purpose, the creation of exoteric knowledge for everyone as opposed to scientific knowledge relevant to a smaller audience. The system seeks this knowledge in such a way as to take human and environmental considerations into account by choosing the right means for a broad spectrum of society. The SIS views the world as a holistic system in which everything is connected to everything else. Complex social and managerial problems must be analyzed as wholes (Mitroff and Linstone 1993) and may require knowledge from any domain, discipline, or profession to solve; reductionism inhibits the solution to such problems.

The SIS is based upon metrology, the science of measurement. Measuring in the system requires cooperation concerning the rules for measuring and agreement on units and standards in order to ensure the ability to replicate – the results agree within an acceptable level of refinement. The refinement process continues by partitioning the measurement process until the readings disagree (i.e. pushing back the decimal places), at which point progress can be made. Variables are “swept in” to explain the discrepancy. By bringing in information from any source to improve its image of the world including varied perspectives or worldviews of a problem, this process represents one of the most holistic aspects of the SIS (Richardson et al.2001). Mitroff and Linstone (1993) stated these perspectives bring to the forefront human beings collectively and individually in all their complexity. Because of the need for cooperation and a holistic view, the SIS has no primary authority built into the system. Every participant possesses authority, no single executive exists.

Refinement alone does not necessarily induce change and adaptation. The most dramatic changes occur when the need arises to revise assumptions. Churchman referred to this as the “heroic mood” necessary to slay the “status quo dragon.” To achieve such a state, the SIS uses a Hegelian dialectic process where the thesis and antithesis work simultaneously to determine the truth. The thesis defends the status quo of an existing paradigm while the antithesis proposes radical change questioning the quality of the status quo. Members of the community are both the observers of the debate as well as the participants. Ultimately, the community reaches a synthesis and the process of seeking new knowledge then continues on. Analyzing this process, Richardson et al. (2001) stated revolutionary change cannot be measured quantitatively and some of the most important measures in the SIS (exoteric knowledge, cooperation, diffusion of authority, and ethical purpose) may only be measurable qualitatively. Overall progress of an SIS should be measured by the extent to which the client, decision maker and designer become the same, that is, involves all of humanity, perhaps even those humans who are no longer living or are yet to be born (Linden et al.2007)

Linden et al. (2007) summarized the nine characteristics of an SIS which correspond to the nine conditions Churchman (1971) deemed necessary for something to be called a system:

1. The system has the purpose of creating knowledge, which means creating the capability of choosing the right means for one's desired ends.
2. The system's measure of performance is the "level" of scientific and educational excellence of all society.
3. The client is humankind, i.e., all human teleological beings.
4. The components of the system have traditionally been disciplines unless the goal is exoteric knowledge that is relevant to everyone in every society.
5. The system has a cooperative environment with fuzzy boundaries necessary for cooperation.
6. The decision makers are ideally everyone, the most important of which are the heroes.
7. The designers are ideally everyone. Progress can be measured in terms of the degree to which the client, decision maker, and designer are the same.
8. The designer's intention is to change the system so as to maximize its value to the client (everyone).
9. There is a built-in guarantor that gives a sense of optimism.

A smattering of knowledge management and decision support research has incorporated Churchman's inquiring systems into system design and decision-making frameworks. Hall et al. (2003) and Hall and Paradice (2005) described the architecture of a learning-oriented KMS developed through the integration of the inquiring systems with Simon's decision-making model.

Richardson, et al. (2006) presented design principles for KMS based on an integration of SIS with Habermas's theory of communicative action. From a decision-making standpoint, Hall and Davis (2007) extended Mitroff and Linstone's (1993) technical, organizational, and personal perspective (TOP) model and Courtney's (2001) DSS paradigm in the development of a value-based decision-making model that encourages multiple value-based perspectives and mediates value conflicts. As Linden et al. (2007) propose, these efforts show the beginnings or, more appropriately, a revitalization of Churchman's inquirers as kernel theories for the design of organizational KMS. The next section of this essay continues Churchman's legacy by integrating the core concepts of SIS and complex adaptive systems theory to present the design principles for a Complex Adaptive Inquiring Organization.

Design Principles for a Continuous Auditing System to Support a CAIO

As can be seen in the previous discussions, CAS theory and Churchman's (1971) SIS share many characteristics. This section delves further by comparing and contrasting the two types of systems (summarized in Table 1) then discusses how they can inform one another to offer a new, enlightened theoretical approach to the design of organizational KMS and DSS. The design principles developed and presented in Table 2 adhere to Churchman's (1971) conditions for a system and are intended to support decision-making in a complex, connected organization facing complex, wicked problems.

Table 1: Characteristics of the Complex Adaptive Inquiring Organization

Characteristic	**Complex Adaptive System**	**Singerian Inquiring System**	**Complex Adaptive Inquiring Organization**
Nature of system	Teleological	Teleological, ethical	Teleological, ethical
Purpose	To evolve and survive	To create exoteric knowledge	To evolve and survive by creating and sharing exoteric organizational knowledge
Measure of performance	Ability to adapt to dynamic environments (co-evolution)	Level of scientific and educational excellence of all society	Ability to adapt to dynamic environments by applying exoteric organizational knowledge
Client	Networked agents	All humankind	The organization and its stakeholders
Environment	Self-motivated, self-organized	Cooperative	Self-motivated and self-organized, but cooperative
Process & Control	Non-linear; no controller or executive; agents interact to spontaneously generate new internal structures and forms of behavior	Non-linear; no controller or executive; variables “swept-in” as necessary for revisions to adjust readings	Non-linear; no controller or executive; agents interact to spontaneously generate new internal structures and forms of behavior; variables “swept-in” as necessary for revisions to adjust readings
Decision makers	Internal agents; decisions based on local, simple rules	Everyone, most importantly “heroes”; focus on ethical behavior	Organizational members and stakeholders

Characteristic	**Complex Adaptive System**	**Singerian Inquiring System**	**Complex Adaptive Inquiring Organization**
Designers	Internal agents	Everyone	Organizational members and stakeholders
Designer intention	Maximize system value to agents	Maximize system value to everyone	Maximize system value to organizational members and stakeholders
Operating mode	Stable or chaotic	Normal or heroic	Normal or heroic
Nature of change	Emergence	Emergence	Emergence

Nature and purpose of the system

Churchman's (1971) first system requirement relates to the nature of a system. In order to be considered a system, something must be teleological (i.e. goal seeking) and consist of interrelated components that work together towards a common goal. Both a CAS and SIS exhibit teleological properties. A CAS seeks some overall goodness of fit for the networked system members (i.e. agents); survival is essential. An SIS strives to create exoteric knowledge for all humankind and implies the ability to know how to act in a specific situation, both from a procedural standpoint and an ethical manner. As Richardson et al. (2001) noted, ethical concerns are receiving increasing attention in information systems and knowledge management research. All parties involved in the design, development, and use of systems in a CAIO must be cognizant of and respond appropriately to the ethical issues facing an organization and its members. The first two system design principles are based on these presumptions:

Design Principle 1: The system must consist of knowledge-seeking, learning, cooperate agents who advance the relative position and fitness of the organization in an ethical manner.

Environment and Measures of Performance

The state of a CAS represents the compilation and interaction of system agents with one another and external environmental elements. Unlike in an SIS where system members are cooperative and wish to align their individual goals to complement each other, CAS agents are generally self-motivated and self-organized working from a simple set of local rules. A CAS succeeds merely by staying in existence – co-evolving with a dynamic environment by adapting to changes forced upon it by the environment and subsequently exacting change back on to the environment. The measure of performance for an SIS, on the other hand, is the level of scientific and educational excellence achieved for all society (Churchman 1971) which Richardson et al. (2001) aptly noted does not yet exist. The next system design principle originates from these concepts:

Design Principle 2: A system agent must be able to adapt to changes in the state of the external environment while, at the same time, increasing the overall knowledge level of an organization, its members, and connected systems.

Client, Designer, and Decision Maker

An SIS strives to integrate the perspectives of the system client, designer, and decision maker ultimately to a state where they become one: all humankind. Such a worldview implores ethical considerations in the design, development, and use of IS as the IS affects the level of knowledge for all humans. Although a lofty goal, this idealistic view defies practicality. To some extent, however, the like-mindedness of the three roles can be an achievable goal within a limited sphere of influence throughout a system. The bounds of the problem must be set to include only the most salient stakeholders (Mitchell et al.1997; Richardson et al.2001) which, for a CAS, includes internal agents and certain tightly-coupled environmental elements and connecting CASs.

Ignoring the connectedness of systems runs the risk of system rejection and/or deleterious, unintended effects. For example, the modernization efforts of the Greek social security organization IKA in the 1980's and 90's to develop and implement IS innovations failed miserably, not because of the technical capabilities of the systems (which were far superior) but rather due to the user community reflecting upon the way their actions impacted themselves and others in their social context (Avgerou and McGrath 2007). IKA employees reflected upon the ethical repercussions of their actions at both the local level and within the immediate sphere of networks and determined they could not, in good conscience, proceed using the systems. The users (i.e. the client), designers, and decision makers failed to work as one towards a common goal. The notions of a converged perspective, ethicality, and tightly-coupled systems lead to the following system design principle:

Design Principle 3: Design of the system must include input from and consideration of the goals for all salient stakeholders.

Operating Mode, Process and Control, and Nature of Change

As previously noted, both a CAS and SIS consist of many system agents interacting with one another and the state of these systems emerges over time through non-linear agent interaction and cannot be undone; no single controller or executive exists that determines overall system direction. The primary difference in system operations between CAS and SIS stems from agent intention. CAS agents focus on maximizing their local goodness of fit that, together, spontaneously (i.e. chaotically) generates new structures and forms of behavior. The system vacillates between a stable and chaotic state. SIS agents, on the other hand, work together towards common goals by "sweeping in" the perspectives of everyone to generate a consensus. The system resides in either a normal state or one referred to as "heroic" where the hero wanders around lost, not knowing what to do but somehow finally finds his way – in essence, chaotic. Churchman (1971) likens this state to a hero in mythology who goes on a quest but is constantly blown off course by storms or battles. Finally, the hero finds his way and successfully accomplishes the goal of saving society. The nature of system operations for CAS and SIS generate the final set of system design principles:

Design Principle 4: The system must allow agents to interact with one another.

Design Principle 5: The system must have the ability to support agents with many different perspectives.

Design Principle 6: The system must have the ability to project future system states in order to facilitate decision-making of wicked problems. The behavior of the CAIO emerges from the interaction of its many and varied agents.

Table 2: Design Principles for a Continuous Auditing System to Support a CAIO

1.	The system must consist of knowledge-seeking, learning, cooperative agents who advance the relative position and fitness of the organization in an ethical manner.
2.	A system agent must be able to adapt to changes in the state of the external environment while, at the same time, increasing the overall knowledge level of an organization, its members, and connected systems.
3.	Design of the system must include input from and consideration of the goals for all salient stakeholders.
4.	The system must allow agents to interact with one another.
5.	The system must have the ability to support agents with many different perspectives.
6.	The system must have the ability to project future system states in order to facilitate decision-making of complex problems. The behavior of the application system emerges from the interactions of its many and varied agents.

Discussion

The Complex Adaptive Inquiring Organization represents the next theoretical model of knowledge management systems for decision-making in learning organizations, an answer to Courtney's (2001) call for a new DSS paradigm to address wicked problems in complex contexts. The design principles presented in this chapter for CAIO systems are grounded in

Complex Adaptive Systems theory and Churchman’s (1971) Singerian Inquiring System to offer guidance on the approach to creating holistic, ethical KMS that support decision-making in a complex, connected environment. I apply these principles in the subsequent chapter to develop an illustrative simulation for the external auditor’s going concern opinion that can be potentially developed as a CA application and discuss current methods and procedures for evaluating the effectiveness and appropriateness of agent-based simulation models.

CHAPTER FOUR: AN INSTANTIATION OF THE CAIO THEORY

Introduction

The fourth chapter of this study presents the approach for an illustration of the Complex Adaptive Inquiring Organization Theory. I created an instantiation of the theory through an agent-based simulation model that assists with the auditor's going concern opinion for a low-fare airline, clearly a complex problem. The remainder of the chapter continues as follows. First, a brief discussion of the nature of design science and the techniques of simulation and agent-based modeling is warranted followed by a description of the research problem. The second section provides background information on the industry and company selected, details the design specifications for the model based on CAIO design principles, explains the overall modeling philosophy, and presents the types of agents and agent behavior. The third section explains the execution of the simulation model, analysis of prospective financial information, comparison of the ABM and prospective information to actual company results. The chapter concludes with a summary and lead-in to the final chapter of the study.

Design Science

The CAIO system design principles developed in this study fall under the design science paradigm. Design science research is intended to both further the academic field and be relevant to practitioners through the rigorous creation of IS artifacts in the form of theories, constructs, methods, frameworks, or instantiations (Parrish 2008) and is fundamentally a problem-solving paradigm (Hevner et al. 2004). Benbasat and Zmud (1999) argue that the relevance of IS research directly ties to the ability of practice to implement theories. This study adheres to the design science requirements of rigorousness and relevance by testing the CAIO organizational learning and knowledge management theory through the creation of an agent-based simulation model in the context of the auditor's going concern opinion. The simulation model represents an instantiation of the theory.

Simulation and Agent-Based Modeling

Organizational simulations depict the behaviors, processes, and outcomes that occur in real organizations and are essentially "automated theories" or sets of assumptions about organizational behavior that are acted out (Cameron and Whetten 1981). The model of a simulation describes the real system and is a theory of behavior representing the way in which some part of the real system works (Forrester 1994). Simulations allow one to draw conclusions and derive implications based on the potential state of an organization should the assumptions and simulation results actually transpire in real life. The overall purpose of simulation is to better understand the inner workings of a system, predict performance, and minimize risks (Wahlstrom 1994) – a general purpose of the auditor going-concern opinion.

Two fundamental types of models exist, deterministic and stochastic (North and Macal 2007). Deterministic models always produce the same output given the same input (i.e. linear relationships). Stochastic models, on the other hand, produce different output when repeatedly run with the same input due to agent and environmental behaviors that possess a range of possible values, means, variances, and other statistical measures (i.e. non-linear). Such models must be executed numerous times in order to produce valid general results. Agent-based modeling represents one type of stochastic simulation modeling that facilitates exploration of the non-linear, adaptive interactions inherent to a CAS. Agent rules of behavior define the actions of individual agents and the interactions between agents. Agent interaction leads to the emergence of structures at the macro-level for the system, any connected systems, and the environment.

The Research Problem

Accounting regulation mandates that auditors assess the ongoing viability of every client's business operations and report any substantial doubt about a company's ability to continue as a going concern for a reasonable period of time in the issued audit report that accompanies the financial statements. The Statement on Auditing Standards No. 59 *The Auditor's Consideration of An Entity's Ability to Continue As a Going Concern* (SAS 59) requires auditors to gain an understanding and assess existing conditions that affect an organization, including those of others in the industry and the economy in general. With the push by the SEC for continuous reporting, associated need for CA, and the quickness by which the state of an organization can change in today's environment, auditors will need to assess continuing operations on a more frequent basis than once a year.

Although not explicitly expected to predict future conditions or events, auditors historically have relied upon bankruptcy prediction models, most commonly the Altman (1968) Z-score model (Dugan and Zavgren 1988; Grice and Dugan 2001; Grice and Ingram 2001; McKee 2003). Bankruptcy prediction poses a challenge to auditors as the identification of cause and effect relationships between factors that may cause or be related to bankruptcy and the actual occurrence of bankruptcy can be difficult (McKee 2003), if not impossible in a complex and connected business environment. McKee (2003) examined 146 U.S. public companies that filed bankruptcy during 1991-1997. Of those, the auditors only reported a going concern problem in 54% of the cases. Further confounding the situation, the models auditors employ contain inherent faults and rely on restrictive assumptions such as linearity, normality, and independence among predictor variables (Zhang et al. 1999). Grice and Ingram (2001) examined the accuracy of the Altman Z-score and Grice and Dugan (2001) reviewed the same for the Zmijewski (1984) and Ohlson (1980) models. Both studies determined all three models failed to transcend generalizability to industries and time periods outside those of the original samples when manufacturing firms dominated the landscape. The environment most definitely has changed and now consists of knowledge-based organizations and economies that require more holistic research approaches.

Not all auditors employ statistical modeling techniques to the going concern assessment. Some perform analytical procedures such as historical trend analyses on operating losses, working capital deficiencies, negative operating cash flow, and adverse key financial ratios. They augment these financial reviews with examination of operational factors (e.g. labor work stoppages and dependence on the success of particular projects) and external circumstances (legal proceedings, changes in legislation, loss of a principal customer or supplier, or a natural disaster). Through discussions with partners from an international public accounting firm, I obtained a copy of the going concern guidance issued for auditors in their Central Florida practice that reflects a non-statistical modeling approach. See Appendix C for detailed language of the guidance but note that the removal of all references to the firm to ensure confidentiality. Each of the three partners elaborated on the guidance by stating that every client is unique and therefore factors may be weighted differently in their assessments from client to client. However, all three agreed the primary indicator of potential business distress relates to the ability to pay short-term debt so the level of working capital (current assets – current liabilities), operating losses, and cash from operations receive the most attention.

The next section applies the CAIO theory to develop an agent-based simulation model for the auditor's going concern opinion that takes into account external conditions and can be executed on a regular basis to obtain a more timely sense of the direction an organization may head (i.e. a CA application). The model is based on the operations and financial condition for Frontier Airlines, a low-fare airline that recently filed for bankruptcy protection.

Construction of the Model

The Company

Frontier Airlines, a low-cost and affordable airline established in 1994, operates primarily in a hub and spoke fashion connecting 49 U.S. cities coast to coast, eight cities in Mexico, and two in Canada through their hub at Denver International Airport (DIA). They currently are the second largest jet service carrier behind United Airlines. In January 2007, the Department of Transportation officially designated the company as a major carrier (at least $1 billion in annual revenue). During the years 2007 and 2006, Frontier increased its year-over-year capacity by 14.4% and 8.4%, respectively, and also increased the volume of passengers by 14.7% and 12.9%, respectively, outpacing their increase in capacity during both periods. This is no small feat given the recent turmoil in the airline industry due to rising fuel costs, tightening of access to credit, declining consumer demand, and bankruptcies and mergers of industry competitors. The company intends to continue its growth strategy by expanding to new markets and adding frequency to existing markets it believes are currently underserved. However, due to the company's lack of borrowing capacity under current lines of credit and lack of other borrowing facilities, Frontier must rely on existing cash and operating cash flows for current operations and future growth.

The nature of the airline industry (i.e. customer service orientation, responsibility to society, heavy regulation, and sensitivity to external forces) and business situation of Frontier offer an interesting subject area for complexity-oriented research that will be pursued through the design and execution of an agent-based simulation model. The design specifications, modeling approach, and details of the simulation model will now be presented.

Design Specifications

Hevner et al. (2004) asserted that IS design theory should guide both researchers and practitioners. These “kernel theories” are applied, tested, modified, and extended through experience, creativity, intuition, and problem solving capabilities of the researcher (Walls et al.1992; Markus et al.2002). As such, this study proceeds from the theory building stage to theory testing and refinement through construction of a simulation model, the IS artifact. The CAIO theory informs the design specifications for the model as summarized in Table 3.

Table 3: Design Specifications

	Design Principle	**Specification**
1.	The system must consist of knowledge-seeking, learning, cooperative agents who advance the relative position and fitness of the organization in an ethical manner.	The simulation will consist of consumers seeking services that meet their individual preferences and multiple airlines that attempt to gain market share through advertising towards those preferences. The users of the simulation (auditors and organizational management) work together to understand the potential impacts of interactions between the airline, consumers, and external elements on future states of the airline. The knowledge gained allows users to manage uncertainty and provide more accurate information about the ongoing viability of the organization – all in the best interests of organizational stakeholders.
2.	A system agent must be able to adapt to changes in the state of the external environment while, at the same time, increasing the overall knowledge level of an organization, its members, and connected systems.	Consumers' purchasing behavior will be dependent upon past purchasing behavior and past experiences. Airlines will adjust advertising based on consumer behavior as well as changes in general economic and other external factors. Both groups of agents will retain knowledge of past transactions in memory that will affect future behavior. The resulting number of passengers served on an annual basis as provided by the simulation will assist the auditors in their going concern opinion as well as offer insight into the inner workings of the economic system, creating knowledge for all salient stakeholders.
3.	Design of the system must include input from and consideration of the goals for all salient stakeholders.	Salient stakeholders in the airline include shareholders, organizational management, consumers, employees, trading partners, and auditors. Ongoing viability of the organization is critical to the future of all these groups. The simulation is designed to support the going concern assessment by the auditors in order to understand conditions that may have an imminent impact on the organization.

	Design Principle	Specification
4.	The system must allow agents to interact with one another.	Consumers, airlines, and external environmental elements interact in a mock economy.
5.	The system must have the ability to support agents with many different perspectives.	The individual agents in each group (consumer, airline, environmental) possess local decision-making rules and preferences.
6.	The system must have the ability to project future system states in order to facilitate decision-making of wicked problems. The behavior of the system emerges from the interactions of its many and varied agents.	The simulated economy will provide an estimate of the number of passengers the airline will serve over the course of one year. This information will be vital to the determination of whether the organization faces the risk of business failure.

Overall Modeling Philosophy

Realism versus Simplicity

The design and analysis of simulation models can range from the very specific to the very general and can be grounded in different combinations of theory and empirical data (Brenner and Werker 2007). Brenner and Werker (2007) presented an informative taxonomy of the various methods of building and analyzing simulation models as noted in Appendix D. Complexity researchers hotly debate and discuss the merits of these approaches, particularly on the dimension of realism (i.e. conventional, microsimulation, and history-friendly approaches) versus simplicity (Bayesian and abductive approaches). A number of researchers advocate the abductive approach that keeps the model as empirical as possible and as general as necessary yet allows the identification of underlying structural elements to explain observations and helps develop theory for the phenomena investigated; abduction enables connection of theory and data in a creative way (Sánchez 2006; Brenner and Werker 2007; Midgley et al.2007; North and Macal 2007). Sánchez (2006) stated:

“Only a replica of the original system, complete in every detail, would have the ability to answer any and every unanticipated question about the system. This is the very antithesis of modeling, since the purpose of modeling is to simplify and abstract to gain insights.”

The experiences of the Midley et al. (2007) supermarket ABM led the research team to believe the emphasis of simulation modeling should be on minimalism: “What are the one or two key aspects of consumer behavior that will explain 80% of the variance in purchases?” In their opinion, the overriding goal should be the development of the simplest model that captures a substantial part of the actual phenomena. This study adopts the minimalist ABM design approach put forth by these researchers and supported by many others.

Agent Design

The basic philosophy for agent design in the airline simulation model in this study consists of the notions of memory and decision rules, both necessary functionality for ABM. The consumer and airline agents possess the capacity to remember past experiences (i.e. knowledge management) and apply simple, local rules for considering and evaluating future opportunities. Consumers focus on personal consumption satisfaction while airlines concentrate on attracting the most passengers possible to increase operating profit and cash. Similar to the Midgley et al. (2007) supermarket retailer, airline agents exhibit larger memory capacity and more systematic decision-making capacities than consumer agents. Consumer decision timeframes vary from one week to a year based on the probability of travel derived from extant research of airline consumer traveling frequencies. After deciding to travel, consumers base purchase decisions on past purchasing behavior and related experiences. However, advertising by airlines may persuade some consumers to switch airlines (i.e. attract new customers) or solidify existing consumer loyalty. Airline agents conduct marketing decisions on a regular basis by analyzing current market share and environmental factors that affect the availability of capital.

Verification and Validation

Simulation models, in essence, act as a surrogate for experimentation with the actual system (existing or proposed) that may be too costly or impractical (Law 2005). Models that do not represent a “close” approximation to the actual system naturally raise concerns about the appropriateness of any conclusions derived from model results. Therefore, model verification and validation are essential parts of the model development process (North and Macal 2007). O’Keefe et al. (1987) define verification as “substantiating that a system correctly implements its specifications” and validation as “substantiating that the system performs with an acceptable level of accuracy”.

For this particular study, I elected to verify and validate the airline simulation model based on some of the techniques advocated by North and Macal (2007). In order to verify the design of my model agents and their interactions/relationships, I discussed my initial set of assumptions about the airline industry with the Director of Financial Analysis Operations at a national airline and revised the assumptions accordingly. Appendix E lists the assumptions included in the final model. The director attended ABM training at the Argonne National Laboratory and therefore possessed both specific domain expertise as well as an understanding of the nature of ABM and simulation modeling.

I programmed the ABM in the visual editor of the Recursive Porous Agent Simulation Toolkit, commonly referred to as Repast Symphony. The visual editor allows the modeler to build decision trees similar to flowcharting that automatically creates underlying Java program code. To validate the Repast programming, I performed three distinct steps. First, I conducted a structured code walk-through of the ABM programming that consisted of the designer (me) presenting the code to a “fresh pair of eyes” (an independent coder) and manually tracing through examples of execution sequences. I performed this process with one of the ABM instructors at the Argonne National Laboratory in order to gain the necessary expertise level to adequately review the model’s code. Second, I varied a number of the key parameters to extreme values in order to “stress test” the model ensuring that the model did not produce unattainable results even under extreme circumstances. Finally, I executed the simulation on previous years’ information for Frontier and compared the model’s results to that of the actual performance by the company. This case approach addresses both the goals of verification and validation. In combination, these steps provide comfort that the model works as designed and produces accurate results within an acceptable range of values. The following section describes each type of agent in more detail.

Types of Agents and Agent Behavior

Consumer Agent Properties

Harris and Uncles (2007) empirically examine a myriad of potential factors that affect airline consumer patronage – including behavioral, performance, promotional, and structural. The results indicate that past purchasing behavior (i.e. frequency of travel with a specific airline) and perception of airline performance (i.e. view of last experience) ranked highest in predictive ability for future purchases. Suzuki (2000) examines the relationship of airline on-time performance to market share noting that the level of satisfaction with the most recent experience can significantly affect future purchases, even for frequent flyers. Adhering to the minimalist abductive approach, the consumer agents in my model represent airline passengers that base their future purchases on past purchasing behavior and related experiences. For example, a consumer who has traveled Frontier Airlines seven times over the past three months is more likely to choose Frontier on the next purchase than a consumer who flew Frontier only once during that same timeframe. Furthermore, customers who encounter a low quality experience on their last Frontier flight will be more likely to switch airlines, particularly if the last prior experience with another airline was positive. Thus, frequent travel and high quality experiences act as anchors in consumer memory for future purchases. Consumer choice behavior in the model is therefore a function of four consumer agent properties that must be populated prior to executing the market simulation (i.e. Time 0):

1) Strength of Memory (1-9 rating)
2) Frequency of Travel (% chance to travel)
3) Last Airline Chosen (Frontier, United, Other, None)
4) Last Trip Experience (Good, Bad, None) for each airline (Frontier, United, and Other)

The incorporation of Strength of Memory (SOM) and Frequency of Travel was facilitated by access to data from a recent study on airline consumer travel frequency conducted by the Cornell University School of Hotel Administration. I classified the strength of memory for consumer agents (as a percentage of total agents) as a 1-9 rating where 1 represents consumers who traveled only once in the past three months and 9 for those that traveled 9 times or more over the same period. So, for example, in a model run consisting of 1,000 consumer agents, the strength of memory property for 717 agents contained a value of 1, 165 with a value of 2, and so forth. These ratings will affect consumer behavior as discussed in the next section. As for the frequency property, consumers in the simulation determine whether or not to travel each week in the 52 week simulation (note: a single ‘tick’ in the model represents time passage of one week). The chance for each consumer to travel is also derived from the data in the Cornell study. For instance, consumers with a Strength of Memory rating of 1 only traveled once during the past three months (i.e. 12 weeks); therefore, the chance to travel in any individual week is 8% (1/12). Table 4 presents the nine Strength of Memory ratings and the associated chance to travel each week.

Table 4: Consumer Travel Frequencies

Strength of Memory Rating	No. of Times Traveled Past 3 Months	Survey Responses	Proportion of Respondents	Proportion of Travelers	Chance to Travel Each Week
	None	61,207	67.39%	0%	0%
1	One time	21,238	23.38%	71.7%	8%
2	Two time	4,887	5.38%	16.5%	17%
3	Three times	1,688	1.86%	5.7%	25%
4	Four times	770	0.85%	2.6%	33%
5	Five times	385	0.42%	1.3%	42%
6	Six times	267	0.29%	0.9%	50%
7	Seven times	148	0.16%	0.5%	58%
8	Eight times	118	0.13%	0.4%	67%
9	Nine times	118	0.13%	0.4%	75%

To determine the value of the last airline chosen property for each consumer agent, the model first calculates a random number from 0-100 then a random number for new travelers (i.e. no last airline), Frontier, and United to “divvy up” the chances that the consumer agent will have flown a specific airline or not flown at all; the Other Airline category receives the remainder. In order to infuse some variability to avoid a deterministic model, the second calculation consists of a 10 point range around the 2006 market share percentage for the two airlines [obtained from the collection of online databases administered by the Bureau of Transportation Statistics (BTS) with a -2% adjustment to accommodate for the consideration of new travelers. For example, Frontier earned the business of 18% of the travelers during 2006 for routes the airline serviced (Bureau of

Transportation Statistics 2008); therefore, the range for the chance that a specific consumer will have flown Frontier on the last trip randomly falls between 11-21% (16% +5/-5). For new travelers, the random number calculation falls between 0-10%. As an illustration of the process, assume the following occurs for the first consumer in the model: Last Airline random number = 75, New Traveler = 5, Frontier = 15, and United Airline = 25. The if/then logic of the model to assign an airline (or none) as the last airline flown occurs in a sequence ending once a value has been assigned. The pseudo-code for this scenario would look like the following:

```
⇨ If Last Airline rand num < 5 Then Last Airline = 0 else
⇨ If Last Airline rand num < 20 Then Last Airline = 1 else
⇨ If Last Airline rand num < 45 Then Last Airline = 2 else
⇨ Last Airline = 3
⇨ End If
```

In this particular instance, the random number 75 causes an assignment of 3 for the last airline flown by this consumer agent.

Following assignment of the Last Airline property, the model determines whether the consumer agent experienced a good (value 0) or bad trip (value 1) by calculating a random number from 0-100 then comparing to the bad trip likelihood percentage (also a random number from 70-100). For instance, assume the model determines 75% of all travelers will experience a bad trip and then calculates a 60 as the random number used to determine the value for the Last Experience property. In this scenario, the consumer will have experienced a good trip with the last airline and thus, the value for Last Experience will be 0. Subsequently, the identical process occurs in order to set an experience value for the remaining two airlines. This holds true even if the consumer has not actually flown one or both of the airlines as the person may have predetermined opinions based on interaction with others in their personal network (i.e. friends, family, etc.). Appendix F displays the decision tree from the Repast visual editor for the coding that determines the initial Last Airline and Last Experience for each airline. Now, on to consumer agent behavior.

Consumer Agent Behavior

Once consumers (and other agent types: airlines, environmental factors) receive the property assignments discussed in the previous section, the model simulation can commence. At Time 1 and each subsequent “tick” representing the next week, the consumer “decides” whether or not to travel based on the probabilities assigned to each consumer’s Frequency of Travel property. Since new travelers have no previous flight experience they follow the same procedure for the assignment of Last Airline and Last Experience that existing travelers (i.e. they have flown before) encountered at Time 0; this occurs only when a new traveler chooses to travel for the first time. Existing travelers, after deciding to fly, the model reads whether Last Experience was good or bad. If good, then the traveler flies with the same airline and the Last Experience is updated for the current trip based on a random number compared against the bad trip likelihood percentage. On the other hand, if the Last Experience was bad, the model first identifies the Strength of Memory property value then continues on one of three distinct paths depending upon that value and will now be discussed.

I divided consumers into three groups based on the SOM: 1) 7, 8, 9 (very frequent travelers), 2) 4, 5, 6 (fly fairly often), and 3) 1, 2, 3 (fly less often). Consumers who fly very frequently typically are most likely to be enrolled in frequent flyer programs and have a stronger commitment to that specific airline; thus, they are not likely to switch airlines. The second group flies, on average, 1.3 – 2 times per month. Therefore, they too are very likely to be enrolled in frequent flyer programs and have a strong commitment to that specific airline but may be persuaded to switch airlines if the last trip was bad and/or they are exposed to airline promotional efforts (more detail provided in discussion of airline agents). For the final group, the model first looks at whether or not the Last Airline for the consumer advertised high during this week. If so, there is a greater likelihood that the consumer will have been exposed and persuaded to stay with the airline. If not, then the model examines the Last Experience for the other two airlines. If the consumer experienced positive trips for both other airlines, then the model randomly assigns one of the two as the airline the consumer now chooses to fly. If both past experiences on the other two airlines were not good, then the model will check to see if one was and the consumer will choose to fly that airline if true. If the consumer experienced bad trips for all three airlines, then the model will randomly choose to stay with the current airline or switch to one of the other two. The following discussion details airline agent properties and behavior.

Airline Agents

The Airport Council International ranked DIA as the fifth busiest airport in the U.S. and 10th in the world serving 47 million passengers annually through 29 airline carriers, as obtained from the 2006 report provided by the Denver International Business Center (City and County of Denver 2006). United dominates the regional market with a 56.4% share of 2006 passengers and Frontier, as a distant second, services 20.7%. Twenty-seven airlines attract the remaining 22.9% with no single carrier serving more than 4.1%. In this simulation, the airline agents consist of United, Frontier, and Others.

Due to the dependence on the DIA market as the sole hub and intense competition at that airport (and others) with United and other carriers, Frontier cannot raise prices to any significant degree to increase profit levels or offset unexpected and/or rising costs. Frontier management state in the 2006 annual report, "Business and leisure travelers continue to reevaluate their travel budgets and remain highly price sensitive" (Frontier Airlines Holdings 2006). For the four year period 2004 – 2007, the average fare ranged from a low of $102.31 in 2005 to a high of $103.54 in 2004, only a maximum swing of 1.2% over four years. Therefore, Frontier and other airline agents must entice consumers primarily through advertising efforts. The following excerpt from the 2006 annual report emphasizes the importance Frontier management places on advertising and brand awareness:

> "Our sales efforts target value conscious leisure and business travelers. Value conscious customers are price-sensitive; however, we believe their travel decisions are also balanced with other aspects of our product offering such as our frequent flyer program, non-stop service, advanced seat assignments, service level and live television entertainment. In the leisure market, we offer discounted fares marketed through the Internet, newspaper, radio and television advertising along with special promotions and travel packages. In May 2003, we launched a new brand strategy and advertising campaign designed to identify Frontier as "A Whole Different Animal" and to set us apart from our competition. The campaign includes television, print and radio components that began running in the Denver market and have since expanded to additional markets along our routes. We have gathered extensive customer and employee feedback that has allowed us to identify elements of service that are important to our

customers who have the potential to fly with us more often." (Frontier Airlines Holdings 2006)

In addition to the new branding campaign, Frontier also engaged in various sponsorship agreements including, but not limited to: the Colorado Avalanche hockey team, the Denver Nuggets basketball team, Colorado Rockies baseball team, etc. Advertising and promotions constituted 9% of total operating expenses in 2006. Clearly, advertising plays a large role in the success or failure of Frontier Airlines.

United Airlines entered bankruptcy proceedings on December 9, 2002 and eventually emerged on February 1, 2006. As expected, companies operating under the court's oversight must focus on managing costs. As a component of overall 2006 operating costs, advertising and promotion only comprised 6% (United Air Lines 2006). The report contained no specific income statement line item for these costs or any discussion of related expenses. Based on this lack of information, I assume United focuses their primary efforts on other areas and relies on their size, extensive flight availability, networks within the industry (e.g. partnership agreements with other airlines and travel agencies), and brand name to attract and retain customers. Therefore, the model places less of an emphasis on advertising for United.

As for the Other airlines category, the model contains no specific agent per se with. This category of airlines places a modicum of value on advertising levels in the consumer purchasing process and remains static. Only the Frontier and United agents evaluate environmental conditions (discussed later) and market share in order to adjust advertising levels.

The Frontier and United agents determine overall advertising levels initially at Time 0 then re-evaluate every two weeks (prior to the start of consumer purchases). Both agents first identify the environmental agent values: Fuel Cost (high/low), Regulation (high/low), and Credit Availability (high/low). As these conditions affect the availability to fund advertising efforts, high Fuel Cost, high Regulation, and low Credit Availability represents the worst case scenario. The agents next determine the advertising level to set based on the values of the environmental agents. Frontier relies heavily on advertising so the model allows them to set the property Advertising to high if two out of the three values are favorable. United, on the other hand, requires all three values to be favorable in order to set Advertising to high.

Starting at beginning of Week 12 of the simulation (i.e. end of the first quarter) and reoccurring every 12 weeks thereafter, the Frontier and United agents calculate the current market share to date in the simulation and compare to a predetermined minimum desired level (Frontier = 16%, United = 21%). I arbitrarily selected these amounts based on historic market share percentages. Should the current market share fall below the threshold, the airline agent re-evaluates the advertising level based on the environmental agent values at that time (same procedure as at Time 0). No action happens if the market share remains above threshold. The environmental factors affecting the airline advertising decisions will now be examined.

Environmental Factors

Several critical economic and environmental factors severely impact the ongoing operations and financial stability of Frontier as well as all carriers in the airline industry and therefore, will be included in the model. First, fuel costs represent the single largest individual operating expense item for Frontier (29.0% in 2007 and 27.9% in 2006) (Frontier Airlines

Holdings 2007, 2006). Over the four year period 2004 – 2007, fuel costs rose 103.8% and the trend is expected to continue with little ability to pass on to consumers. Second, subjection to heavy federal regulation has resulted in operating cost increases in the past and may do so in the future. For example, President Bush signed the Stabilization Act in to law in 2001 that federalizes most civil aviation security and requires the implementation of certain security measures by airlines and airports such as screening all passenger baggage by the Transportation Security Administration (TSA). Funding for these security activities comes from a $2.50 per enplanement ticket tax and the TSA can impose additional fees on air carriers as the agency deems necessary (Frontier Airlines Holdings 2006). In the "Risk Factors" section of their respective 2006 annual reports, both airlines acknowledge extensive government regulation could increase operating costs and affect the ability to conduct business. To the extent costs of measures such as the Stabilization Act cannot be passed on to consumers, airlines face significant financial challenges. Finally, availability of credit in the broader economic market or lack thereof poses a great risk to airlines in several respects. When credit is scarce and the economy suffers, consumers tend to tighten their purse strings and travel less. Creditors also become more stringent and wary of extending additional credit. As noted in their 2006 annual reports, both Frontier and United suffer from very little available credit and must rely on existing cash and generation of operating cash flows to support operating activities. Another potential consequence of tighter credit markets, commonly referred to as the "credit-card holdback", affects the airline industry specifically. Airlines contract with one or more (usually no more than three) credit card companies to process their customers' credit purchases. Typically, credit card processors immediately pass along the majority of proceeds to airlines with the remainder held back until the passenger completes the flight. However, processors have the right at any time to raise the amount held until completion of service should they feel at risk of an airline discontinuing service thus transferring responsibility to them for the money (Atlanta Journal-Constitution 2008). This sort of event could adversely affect the cash flow position of an airline, particularly those dependent on operating cash to survive.

Accordingly, the simulation model contains external agents that represent fuel costs, federal regulation, and credit availability. In actuality, the agents are proto-agents rather than full-fledge agents. The behavior of proto-agents affect other agents but do not act and react to changes to or made by other agents. The environmental agents in this study randomly start with either a high or low value then have the potential to adjust (randomly) on a predetermined schedule: Fuel Cost has a 60% chance of being set to high at Time 0 and every 4 weeks, Regulation 60% chance at Time 0 and every 12 weeks, and Credit Availability 50% at Time 0 and every 4 weeks.

Design Specifications and Model Functionality

Using the design principles derived from the CAIO theory presented in the previous chapter, I developed an agent-based model of the Frontier Airlines market as an illustration of the type of continuous auditing application an auditor can realistically develop as a tool in the analysis of a client's ongoing financial viability. I strived to maintain simplicity in the model by focusing on the factors that provide the greatest impact. The model contains agents representing consumers, airlines, and environmental factors that act and react to the behaviors of each other in a simulated economy.

Midgley et al. (2007) recommend 1) publishing detailed specifications and programming code, 2) enlisting the assistance of programming experts to inspect and correct code implementation of the specifications, 3) subjecting the model to destructive testing, and 4) empirically validating the model against real data. In this vein, Table 3 proposes a set of design specifications for the ABM and Table 5 compares the functionality built into the model to these specifications. Appendix G provides the Java coding for entire agent-based model. As previously noted, I enlisted the assistance of an ABM programming expert to evaluate the correctness of the coding as well as the implementation of design specifications and intended functionality. During initial model testing both I and the external expert executed the model numerous times to assess the robustness of the model to extreme parameter values. The next section of this chapter presents the results of the model simulation and subsequent financial statement analysis of projected 2007 operating results for Frontier Airlines to the actual 2007 results.

Table 5: Model Functionality

	Design Specification	Model Functionality
1.	The simulation will consist of consumers seeking services that meet their individual preferences and multiple airlines that attempt to gain market share through advertising towards those preferences. The users of the simulation (auditors and organizational management) work together to understand the potential impacts of interactions between the airline, consumers, and external elements on future states of the airline. The knowledge gained allows users to manage uncertainty and provide more accurate information about the ongoing viability of the organization – all in the best interests of organizational stakeholders.	The agent-based model consists of consumer agents that decide to travel (or not) on a weekly basis. Airline choices are dependent upon past experience with each of the airlines as well as exposure to advertising and promotion efforts by the airlines. Airline agents try to retain existing customers and gain additional market share through advertising. Users of the model can alter consumer preferences, advertising levels, and environmental factors in order to evaluate the effects of differing scenarios. The knowledge gained can assist in judging the ongoing viability of the organization.
2.	Consumers' purchasing behavior will be dependent upon past purchasing behavior and past experiences. Airlines will adjust advertising based on consumer behavior as well as changes in general economic and other external factors. Both groups of agents will retain knowledge of past transactions in memory that will affect future behavior. The resulting number of passengers served on an annual basis as provided by the simulation will assist the auditors in their going concern opinion as well as offer insight into the inner workings of the economic system, creating knowledge for all salient stakeholders.	The agent-based model consists of consumer agents that decide to travel (or not) on a weekly basis. Airline choices are dependent upon past experience with each of the airlines as well as exposure to advertising and promotion efforts by the airlines. After each trip, consumer agents update memory with the most recent airline choice and trip experience. Airline agents try to retain existing customers and gain additional market share through advertising provided the current state of environmental factors does not restrict cash available for advertising expenditures. The model tracks each consumer purchase over 52 'ticks' or weeks and provides market share data for each airline over the course of the simulated year. Auditors can detect a general trend in market share and use the data to calculate revenue for the prospective financial statements to be used in the going concern opinion.

	Design Specification	Model Functionality
3.	Salient stakeholders in the airline include shareholders, organizational management, consumers, employees, trading partners, and auditors. Ongoing viability of the organization is critical to the future of all these groups. The simulation is designed to support the going concern assessment by the auditors in order to understand conditions that may have an imminent impact on the organization.	Market share data calculated in the model simulation can be used to derive annual passenger revenue, the key driver to airline financial performance. Auditors can develop multiple prospective income statements using model data that depict the impact of different market conditions on the airline's financial performance. These prospective statements will help determine whether the airline's operating income is sufficient to maintain ongoing operations for the upcoming year given key market factors.
4.	Consumers, airlines, and external environmental elements interact in a mock economy.	The agent-based model consists of consumer agents; agents for Frontier Airlines and United Airlines; and agents that represent fuel costs, federal regulation, and credit availability. The agents interact in a 52 week simulation of the airline market that Frontier Airlines services.
5.	The individual agents in each group (consumer, airline, environmental) possess local decision-making rules and preferences.	Each consumer agent possesses memory that contains information about the last airline flown and the related experience (good/bad) as well as the last experience on the other two airlines, if the consumer had ever flown on them. The agent also has a strength of memory factor that dictates both how loyal the agent is (if at all) to the most recent airline as well as how susceptible the agent is to airline advertising. The Frontier and United airline agents evaluate current environmental market conditions to periodically set their levels of advertising. In addition, the agents regularly calculate their current market share and compare to a desired amount. If the airline's market share falls below this threshold, the airline examines current environmental factors to determine if they can raise advertising levels to raise market share. The three environmental factors randomly determine their levels on a scheduled basis.

	Design Specification	Model Functionality
6.	The simulated economy will provide an estimate of the number of passengers the airline will serve over the course of one year. This information will be vital to the determination of whether the organization faces the risk of business failure.	The model tracks each consumer purchase over 52 ‘ticks’ or weeks and provides market share data for each airline over the course of the simulated year. This data can be used to determine projected passenger revenue for the prospective income statement which will be used to evaluate the airline’s ongoing viability.

Simulation Results and Financial Statement Analysis

Test Design

The ABM represents a simulation of the 2007 airline market in which Frontier Airlines operates. The goal of the model entails generating a market share percentage for Frontier that can be incorporated into an overall financial analysis of the company. Determination of market share percentage helps derive passenger revenue (an airline’s primary source of income) that is included in the prospective income statement developed by the auditor to determine operating income and associated operating cash flow. As previously noted, auditors generally view operating income and operating cash flow as the foremost indicators of a company’s near-term financial viability.

As the auditor’s going concern opinion occurs once a year in conjunction with the annual audit report, the model simulates one year of Frontier’s airline market. Each ‘tick’ represents a single week (52 total ticks) where consumers decide to travel (or not) and airline agents monitor consumer and environmental agent activity to determine their levels of advertising expenditures. Each ‘run’ of an ABM simulation (i.e. 52 week simulation) can generate wildly different results from any other run as local agent interactions occur and unpredictably influence the collective state of the system. A modeler must therefore not rely on a single run but examine the general trend of multiple runs (North and Macal 2007). I chose to execute 30 runs in succession as a 'batch' and nine batches in total differing on number of total consumers (one thousand, ten thousand, and one hundred thousand) and states of the environmental factors (unconstrained, all set to bad, all set to good). Unfortunately, computing power of my laptop restricted analysis of a consumer market greater than 100,000. Auditors developing a similar ABM would have extensive resources to draw upon. Regarding the environmental factor settings, auditors typically wish to analyze multiple scenarios before issuing a going concern opinion. Forcing the factors to remain at certain levels facilitates examination of worst case and best case scenarios while allowing the factors to be 'free' provides a most likely scenario.

Simulation Results

Across the three consumer levels (i.e. one thousand, ten thousand, and one hundred thousand) with 30 runs each, Frontier's anticipated 2007 market share in the unconstrained model averaged 17.64% (range 15.74% - 19.33%) with United at 24.99% and Other at 57.37%. Consumers switched airlines 39.69% of the time. Frontier gained 29.38% of these 'switchers' and lost 28.95%, a slight net gain. Of the total Frontier trips, consumers switching to Frontier comprised 66.10%, repeat passengers 33.14% (i.e. non-switchers), and new flyers 0.74%.

Under the worst case scenario with both Fuel Cost and Regulation set at high and Credit Availability set at low, Frontier earned 16.03% (range 14.57% - 17.51%) of the market with United at 24.84% and Other at 59.13%. Consumers switched airlines 41.22% of the time. Frontier gained 27.36% of these 'switchers' and lost 27.76%, a slight net loss. Of the total Frontier trips, consumers switching to Frontier comprised 70.35%, repeat passengers 28.81%

(i.e. non-switchers), and new flyers 0.84%. Further analysis shows Frontier attracted 2.02% less switchers and retained 20.93% fewer repeat passengers in the ‘bad’ environment. To determine if the difference in Frontier’s market share is different between the unconstrained model and the bad model, I performed an independent-samples t-test comparison of mean values for the two sets (3 batches each with 30 records for a total n=90 per model). The test revealed the two sample means are statistically different (t = 14.782, p < .01) indicating the inability to advertise due to poor environmental factors severely affected Frontier’s ability to attract and retain consumers.

Under the best case scenario with both Fuel Cost and Regulation set at low and Credit Availability set at high, Frontier earned 21.25% (range 20.35% - 22.63%) of the market with United at 20.91% and Other at 57.84%. Consumers switched airlines 39.74% of the time. Frontier gained 32.43% of these ‘switchers’ and lost 30.81%, a weighty net gain. Of the total Frontier trips, consumers switching to Frontier comprised 60.66%, repeat passengers 38.68% (i.e. non-switchers), and new flyers 0.66%. Further analysis shows Frontier attracted 3.06% more switchers and retained 40.88% more repeat passengers in the ‘good’ environment. To determine if the difference in Frontier’s market share is different between the unconstrained model and the good model, I performed an independent-samples t-test comparison of mean values for the two sets (3 batches each with 30 records for a total n=90 per model). The test revealed the two sample means are statistically different (t = -35.388, p < .01) indicating the ability to advertise due to better environmental factors positively affected Frontier’s ability to attract and retain consumers.

The Bureau of Transportation Statistics reported that Frontier earned 17.46% of the market share for the routes the company serviced during 2007 with United at 32.20% and Other at 50.34% (Bureau of Transportation Statistics 2008). After three batches and 90 runs, the model estimated only 0.18% higher than Frontier's actual but underestimated United by 7.21% and overestimated Other by 7.03%. United's activity on these routes fluctuates significantly rising from a 23% market share in 2006 to 32% in 2007 then falling again to 25% in 2008 as the company acquires/sells gates and adds/subtracts flights. Given the focus of the model is to estimate Frontier's status in the marketplace, these results seem reasonable.

Table 6 highlights the results of the three models. Using the estimated market share from the unconstrained model, I will now create a prospective income statement to estimate operating income and operating cash flow in order to conduct a mock going concern analysis.

Table 6: Comparison of Model Simulation Results

	Unconstrained Model	**Bad Environment**	**Good Environment**
Frontier Market Share	17.64%	16.03%	21.25%
United Market Share	24.99%	24.84%	20.91%
Other Market Share	57.37%	59.13%	57.84%
Consumer Switching %	39.69%	41.22%	39.74%
Repeat Passengers (of total Frontier)	33.14%	28.81%	38.68%
Passengers Switching to Frontier (of total Frontier)	66.10%	70.35%	60.66%
New Flyers (of total Frontier)	0.76%	0.84%	0.66%
Change in Retention	N/A	-20.93%	+40.88%
Change in Attracting Switchers to Frontier	N/A	-2.02%	+3.06%

Financial Statement Analysis

As discussed previously, some auditors use statistical models (e.g. Altman Z-score) while others perform analytical procedures of key financial numbers to assess the ongoing viability of audit clients per SAS 59 requirements (i.e. the going concern opinion). Inherent problems arise with the structure and use of the more common statistical models hence the application of the alternative method that allows greater auditor judgment and reliance on expertise. Analytical procedures are comparisons of unaudited financial data with expected results (Glover et al.2000). These procedures occur primarily at the beginning of an audit to plan the nature, timing, and extent of testing (American Institute of Public Accountants 1998). Significant fluctuations between client data and auditor expectations signal increased risk of material error for a particular accounting area (e.g. valuation of accounts receivable and estimated bad debts). Auditors perform essentially the same process at the end of the audit to develop expectations of the financial statements for the upcoming year, also known as prospective financial statement forecasts. The prospective information (most notably the estimated operating income and operating cash flow) form the basis for the auditor’s going concern opinion. Drawing on results from the ABM developed in this study and historical trend analyses, the remainder of this chapter details the creation of 2007 prospective statements for Frontier Airlines and comparison to audited 2007 financial results.

The process to create prospective financial statements consists of first identifying the key account balances and financial statement line items that comprise the majority of each statement. Once determined, the auditor develops expectations of each item then assembles them together in the appropriate financial statement format. Expectations result from analysis of historical financial data, anticipated actions by management, industry factors that may affect the major financial statement line items. For this exercise, I chose to analyze the following income statement line items to project Frontier’s 2007 operating income and operating cash flow: Passenger Revenue, Cargo Revenue, Other Revenue, Fuel Costs, Promotion & Sales Expenses, and Other Expenses. Table 7 lists the line items and rationale for each estimate while Appendix H presents the prospective operating income statement and expected operating cash flow with historical actuals. Development of each expectation will now be explained.

Table 7: Prospective Operating Income and Cash Flow Items

Financial Line Item	**Rationale for Expectation**
Passenger Revenue	Combination of market share from ABM, industry outlook, Frontier expansion, historical revenue generated per passenger
Cargo Revenue	Percentage increase from two years prior to previous year carried forward
Other Revenue	Average three year change in year-to-year growth %
Fuel Costs	Average three year change in year-to-year growth % and adjustment for increase in use of free capacity
Promotion & Sales Expenses	Three year average percentage of revenue
Other Expenses	Previous year's expense structure carried forward
Operating Income	Projected revenues – projected expenses
Operating Cash Flow	Three year average percentage of operating income

Revenue Estimates

Passenger revenue constitutes the majority of overall Frontier revenue (e.g. 97% in 2006) and requires the estimation of two components: number of passengers and the revenue generated per passenger. To develop the estimate of 2007 passenger revenue, I first calculated a total market for routes Frontier services. BTS reported 40,784,999 passengers traveling the routes in 2006. In addition to this figure I analyzed potential adjustments related to industry projections and management's expansion efforts. In a 2007 economic outlook report sponsored by the Air Transport Association of America (ATA), the ATA Vice President and Chief Economist predicts a $4 billion industry net profit on operating revenues but also foresees a deceleration in passenger and cargo revenue growth due to declining global economic growth (Heimlich 2007).

The industry net profit is primarily a result of continued cost-cutting efforts. For the sake of conservatism, I included a 0% impact on the growth of passengers related to overall industry growth. Regarding management's expansion of the Frontier market, the company increased the number of passengers flown each of the last five years by an average of 24.42% which I continue forward as a proportionate increase to the total Frontier market. This process culminates in a projected market for Frontier of 50.75 million travelers.

After estimating a total market I determined a proportion to assign to Frontier. The unconstrained model of the ABM generated a 17.64% Frontier market share. Multiplying the simulated market share percentage times the projected total market results in an estimated 8.95 million passengers for Frontier. However, a discrepancy exists between the amount of passengers BTS reports for Frontier in 2006 and the amount reported in the financial statements. Of the 40.78 million passengers traveling Frontier routes in 2006, Frontier earned the business of 7.47 million (18.32%) according to BTS. The 2006 financial statements reported 8.68 million revenue passengers carried. The discrepancy lies in the various agreements with other airlines such as code-sharing and outsourcing of certain routes. BTS reports these passengers under the airline actually operating the flight, not the company whose financial statements include these passengers. To reflect the difference, I adjust the 8.95 million passengers estimated using BTS figures by +16% (the amount BTS differed from the financial statements in 2006) resulting in a projection of 10.38 million Frontier passengers.

To estimate the second component necessary to project passenger revenue, I calculated the revenue per passenger for the years 2003-2006. Using the average change from year-to-year for the period (-0.35%), I calculated $111.58 of revenue to be generated by each of the 10.38 million passengers. Projected passenger revenue for 2007 is $1.16 billion.

Cargo revenue represents a much smaller portion of total revenue, only 0.57% in 2006. Therefore, I merely carried forward the 15% increase from 2005 to 2006 to generate a projected $6.53 million. The other revenue category comprised 2.43% of total 2006 revenue. Applying a continuation of the three year average change in year-to-year growth of +31.12% to 2007 results in a projected $31.91 million of other revenue. Estimated operating revenues total $1.197 billion.

Cost and Expense Estimates

Auditors often estimate operating expenses in direct proportion to operating revenues. As $1 is earned a certain percentage of expenses are incurred. I prescribed to this approach to estimate Frontier operating expenses. As discussed in the construction of the ABM, fuel costs represent the single largest expense for Frontier Airlines, 27.93% in 2006. At the end of 2006 the average three year change in year-to-year growth of fuel costs relative to operating revenue was 29.73%. Frontier continually displays an increase of passengers over and beyond capacity meaning the company continues to be more efficient each year. The extra fuel cost of operating a flight with additional passengers in lieu of empty seats is negligible. Therefore, I adjusted the estimated fuel cost in relation to revenue by -0.25% to represent continued increased efficiency in 2007. As a percentage of revenue, I estimated fuel costs at $352.91 million. Similarly, I estimated promotion and sales expenses as $117.78 million based on a three year average relative to operating revenue (9.84%). For other expenses, I carried forward the cost structure (63.68% of revenue) to estimate $738.39 million. Estimated operating expenses total $1.209 billion.

Analysis of Projected Operating Income and Cash Flow

Operating income equals the net of operating revenues and operating expenses. The prospective statements for Frontier's 2007 operations shows a loss of $11.936 million (see Appendix H for additional detail). Although Frontier generated a net operating loss in two of the past three years since 2004, the company managed to generate positive operating cash flow each year with a factor of -205.15% of operating revenue. In fact, the company earned positive operating cash every year since 1999. Applying this percentage to the projected loss for 2007 provides estimated operating cash inflow of $24.49 million. Obviously, non-cash expenses drove operating income down to a loss level. For example, adding back the $28.37 million depreciation on fixed assets in 2006 (the vast majority being aircraft) would have resulted in a $20.48 million operating profit. As the auditor examining the 2007 prospective financial statements and issuing the going concern opinion, I would state Frontier faces very little risk of not being able pay short-term obligations the next year. However, the company cannot continue to experience net losses year after year before shareholders become impatient and creditors more leery.

So, how well did the prospective financial statements developed using the ABM simulation compare to Frontier's actual, audited financial data for 2007? As previously noted, the ABM derived a 17.64% market share for Frontier whereas the actual was 17.46%. Frontier reported 10.04 million passengers carried (3.33% less than prospective) and $1.17 billion revenue (2.19% less). However, the revenue generated per passenger of $112.71 exceeded that in the prospective financial statements by 1.01%. Furthermore, actual operating expenses were $1.18 billion (2.27% less) providing a higher actual operating income (a net loss of $9.83 million). Frontier appears to have further streamlined expenses to become even more efficient.

Frontier generated positive operating cash flow of $23.23 million which is 5.14% less than prospective and a higher percentage relative to operating revenues (-236.19%) meaning the company squeezed out more cash per revenue dollar than previous years. Based on actual operating income and the three year average factor of -205.15% included in the prospective financial statements, the expectation would be the generation of $20.18 million of free cash. On the key figures of operating income and operating cash flow, Frontier slightly outperformed the prospective financial statements. Overall, the ABM helped produce prospective financial statements fairly close to actual 2007 operations. Appendix I provides a comparison of prospective financial information to actuals.

Discussion and Summary

The fourth chapter of this study presented an instantiation of the Complex Adaptive Inquiring Organization Theory through an agent-based simulation model to assist with the auditor's going concern opinion for Frontier Airlines. The design of the model adhered to prescribed guidelines and best practices for design science, simulation, and agent-based modeling. The design specifications presented were based on CAIO design principles and acted as the guide for developing the types of agents, their behavior, and interaction with other agents. Multiple executions of the simulation model produced an expected market share for Frontier Airlines used to develop prospective financial information for 2007 necessary to issue a going concern opinion for the viability of Frontier during the next year. The ABM built from the underlying theories of complexity science and Singerian inquiring systems helped develop prospective financial information that estimated actual results fairly well and showed Frontier consistently produces positive operating cash in spite of continued net operating losses. The final chapter of this study summarizes the study as a whole and discusses the implications of continuous auditing in a complex business environment.

CHAPTER FIVE: CONCLUSION

Starting June 15, 2009, the mandatory XBRL reporting requirements for SEC registrant companies will begin the phase-in process based on company type and size. All registrants will be required to file interactive financial statements for fiscal year ends on or after June 15, 2011. All filings that contain financial information (e.g. annual 10-K, quarterly 10-Q, and 8-K revisions that can occur at any time) are subject to the XBRL requirements. The progression to more available and readable financial information helps fuel the fire for continuous auditing and assurance of said information. Continuous auditing applications and systems will need to be developed to meet the diverse needs of the assurers (public accounting firms), company management producing the financial statements, shareholders, consumers, regulatory authorities, financial market participants, creditors, etc. This network of stakeholders interacts with each other as well as members in other extended networks. Continuous auditing systems therefore must be designed to support constantly changing environments, generate new knowledge, and provide decision support in an increasingly complex and connected world.

Little research has focused on the design of continuous auditing systems and the interaction of the various stakeholders involved in and affected by the attestation process. This research study addressed the lack of an underlying system design theory and comprehensive view with the goal to determine how continuous auditing systems should be designed to produce knowledge that benefits auditors, clients, and society as a whole. To accomplish this, I first developed a comprehensive, system design theory called the Complex Adaptive Inquiring Organization Theory based on the fundamentals of complexity theory and Churchman's (1971) Singerian Inquiring Systems. Next, I established a set of associated design principles for continuous auditing systems. Applying these design principles, I then illustrated the general model by creating an agent-based simulation model of the Frontier Airlines market that includes agents representing Frontier, key competitors, consumers, and the general economic environment.

Using the Frontier agent-based model as a decision support system, I conducted a mock going concern analysis of the ongoing viability of Frontier by incorporating the airline's share of the market (per the ABM) as an input into the construction of 2007 prospective financial statements – much the same as the company's auditors perform each year. Comparison of the simulation results and prospective financial statements to actual operating results indicated the ABM developed from the CAIO theory performed well as a potential tool for continuous auditing of a company's financial health. The 17.64% market share generated by the unconstrained version of the ABM produced a projected 10.38 million Frontier passengers in 2007 and related passenger revenue of $1.15 billion. After developing estimates for other key line items of the income statement based on historical company trends, the prospective financial statements forecasted an $11.94 million operating loss and $24.49 million operating cash inflow. For 2007, Frontier earned 17.46% of the market, serviced 10.04 million passengers, earned $1.13 billion in passenger revenue, and reported a $9.83 million operating loss and $23.23 million operating cash inflow. With the assistance of the ABM based on the CAIO theory, the prospective financial statements offered a conservative estimate of the future financial performance of Frontier Airlines. An actual going concern opinion based on this model would have been slightly guarded and adhered to one of the core tenets accounting and auditing that of conservatism. An adverse going concern opinion can be a self-fulfilling prophecy that may ruin a company's future. Auditors must be extraordinarily cautious.

In practice, auditors can develop similar models to the ABM developed in this study based on their intimate knowledge of their clients and respective industries that can assist in the going concern opinion. Note, however, the going concern opinion represents the final stage of the audit. The opinion looks forward using the most recent audited financial statement information that depicts the current state of the company (i.e. the balance sheet) and the most recent performance (i.e. income statement). Should the financial information contain material errors or are fraudulent, the auditor's going concern opinion will be built on a faulty foundation and thus less likely to provide an accurate assessment of the future state of the client. The use of independent data such as oil prices, industry trends, etc. in an ABM can reduce the reliance on internal client information that may be biased or incorrect and help produce a higher quality opinion. Furthermore, as the movement of continuous reporting progresses the market will demand continuous assurance of reported financial information. Auditors can develop their models to "sweep in" current/updated information to provide more frequent and accurate opinions of the ongoing viability of their clients, create more elaborate models that build expectations for all the key financial statement line items rather than just the single one in illustrated in this study (i.e. passenger revenue), and ultimately meet the knowledge needs of associated stakeholders.

As continuous reporting becomes more commonplace academics and practitioners will need to address the functionality and ramifications of a continuous assurance environment in a business world that appears to be "flattening" and becoming more interconnected as evidenced by the Great Credit Crisis of 2008 currently affecting the entire global economic system. Systems will inevitably be created. How will they be designed? Will they serve needs of humanity, the ultimate stakeholder? Only time will tell. I humbly offer a starting point. For now, the complexity movement in the accounting and IS disciplines is under way and, as CASs infamously do, the movement will feed upon the energy of itself and the environment to evolve exponentially in a non-linear manner. The future is unknown, but I am confident an underlying pattern exists suggesting a bright future lies ahead for complexity research in accounting and IS, starting with continuous assurance.

APPENDIX A: ACCOUNTING JOURNALS

Acronym	**Journal Name**
AAAJ	*Accounting, Auditing, and Accountability Journal*
AAF	*Accounting and Finance*
AAR	*Australian Accounting Review*
ABA	*Abacus*
ABF	*Accounting, Business and Financial History*
ABR	*Accounting and Business Research*
AE	*Accounting Education*
AEJ	*Accounting Educators' Journal*
AEN	*Accounting Enquiries*
AFO	*Accounting Forum*
AHI	*Accounting Historians Journal*
AHJ	*Accounting Historians Journal*
AHO	*Accounting Horizons*
AIA	*Advances in Accounting*
AIN	*Advances in International Accounting*
AIS	*Advances in Accounting Information Systems*
AIT	*Advances in Taxation*
AMA	*Advances in Management Accounting*
AOS	*Accounting, Organizations and Society*
API	*Advances in Public Interest Accounting*
AUD	*Auditing: A Journal of Practice & Theory*
BAR	*British Accounting Review*
BRA	*Behavioral Research in Accounting*
CAR	*Contemporary Accounting Research*
CPA	*Critical Perspectives on Accounting*
EAR	*European Accounting Review*
ECA	*Economie Applique*
ESP	*Espace Europe*
FAM	*Financial Accountability and Management*
HBR	*Harvard Business Review*
IAE	*Issues in Accounting Education*
IAU	*International Journal of Auditing*
IJA	*International Journal of Accounting*
JAA	*Journal of Accounting, Auditing, and Finance*

JAC	*Journal of Accountancy*
JAE	*Journal of Accounting and Economics*
JAL	*Journal of Accounting Literature*
JAP	*Journal of Accounting and Public Policy*
JAR	*Journal of Accounting Research*
JATA	*Journal of American Taxation Association*
JBFA	*Journal of Business Finance and Accounting*
JCA	*Journal of Cost Analysis*
JCM	*Journal of Cost Management*
JED	*Journal of Accounting Education*
JIA	*Journal of International Accounting, Auditing, & Taxation*
JIF	*Journal of International Financial Management and Accounting*
JMA	*Journal of Management Accounting Research*
JTA	*Journal of Taxation*
MAR	*Management Accounting Research*
NTJ	*National Tax Journal*
RAE	*Research on Accounting Ethics*
RAR	*Research in Accounting Regulation*
RGN	*Research in Government & Non-Profit Accounting*
SBR	*Schmalenbach Business Review*
TAD	*Tax Adviser*
TAR	*The Accounting Review*
TLR	*Tax Law Review*

APPENDIX B: INFORMATION SYSTEMS JOURNALS

Acronym	**Journal Name**
ACMTDS	*ACM Transactions on Database Systems*
ACMTrans	*ACM Transactions (various)*
ACS	*ACM Computing Surveys*
AI	*Artificial Intelligence*
AIMag	*AI Magazine*
AMJ	*Academy of Management Journal*
CACM	*Communications of the ACM*
CAIS	*Communications of the AIS*
DSI	*Decision Sciences*
DSS	*Decision Support Systems*
EJIS	*European Journal of Information Systems*
HBR	*Harvard Business Review*
I&M	*Information & Management*
IEEESw	*IEEE Software*
IEEETC	*IEEE Transactions on Computers*
IEEETrans	*IEEE Transactions (various)*
IEEETSE	*IEEE Transactions on Software Engineering*
IEEETSMC	*IEEE Transactions on Systems, Man, and Cybernetics*
ISF	*Information Systems Frontiers*
ISR	*Information Systems Research*
JAIS	*Journal of the AIS*
JCOMP	*Journal on Computing*
JCSS	*Journal of Computer and System Sciences*
JMIS	*Journal of Management Information Systems*
JMS	*Journal of Management Science*
MISQ	*MIS Quarterly*
MS	*Management Science*
OS	*Organization Science*
SMR	*Sloan Management Review*

APPENDIX C: GOING CONCERN GUIDANCE

This memo is to document the process to evaluate an organizations ability to continue as a going concern and the key factors considered in such evaluation.

To assess whether there is substantial doubt about an entity's ability to continue as a going concern for a reasonable period of time the auditor first considers whether the results of the procedures performed throughout the engagement identify conditions or events that could indicate, in the aggregate, a potential going concern assessment. If the auditor believes there is substantial doubt about the entity's ability to continue as a going concern for a reasonable period of time he should obtain information about management's plans that are intended to mitigate the effect of such conditions or events and assess the likelihood that such plans can be effectively implemented.

Through the audit procedures performed the auditor should consider the analytical procedures, subsequent events, compliance with terms of debt and loan agreements, minutes of stockholder meetings, legal counsel's response regarding litigation, claims, and assessments, as well as confirmation with related and third parties of arrangements to provide or maintain financial support in order to appropriately determine if there is substantial doubt regarding the entity's ability to continue as a going concern for a reasonable period of time. The following are a list of conditions or events that should be considered from the review of the above sources:

- Negative trends – recurring operating losses, working capital deficiencies, negative cash flows from operating activities, adverse key financial ratios

- Other indicators of financial difficulties – default on loan or similar agreements, arrearages in dividends, denial of usual trade credit from suppliers, restructuring of debt, non compliance with statutory capital requirements, potential disposal of substantial assets, or the entity needs to seek new sources or methods of financing

- Internal matters – work stoppages or other labor difficulties, substantial dependence on the success of a particular project, need to significantly revise operations

- External matters – legal proceedings, changes in legislation that may adversely affect the entity, loss of a key franchise, license or patent, loss of a principal customer or supplier, or uninsured or under insured catastrophe or natural disaster

If it is believed that there is substantial doubt about the entity's ability to continue as a going concern for a reasonable period of time after considering the above, the auditor should obtain information about management's plans for dealing with the adverse effects of the conditions and events. The major considerations are the likelihood that such plans can be effectively implemented over a reasonable period of time to mitigate the adverse effects. In particular the auditor should consider the adequacy of support regarding the ability to obtain additional financing or the planned disposal of assets.

If the auditor concludes there is substantial doubt, he should consider the adequacy of disclosure about the entity's possible inability to continue as a going concern for a reasonable period of time and include an explanatory paragraph (following the opinion paragraph) in his audit report to reflect his conclusion. If the auditor concludes that substantial doubt does not exist, he should consider the need for disclosure.

APPENDIX D: A TAXONOMY OF INFERENCE

(Brenner and Werker 2007)

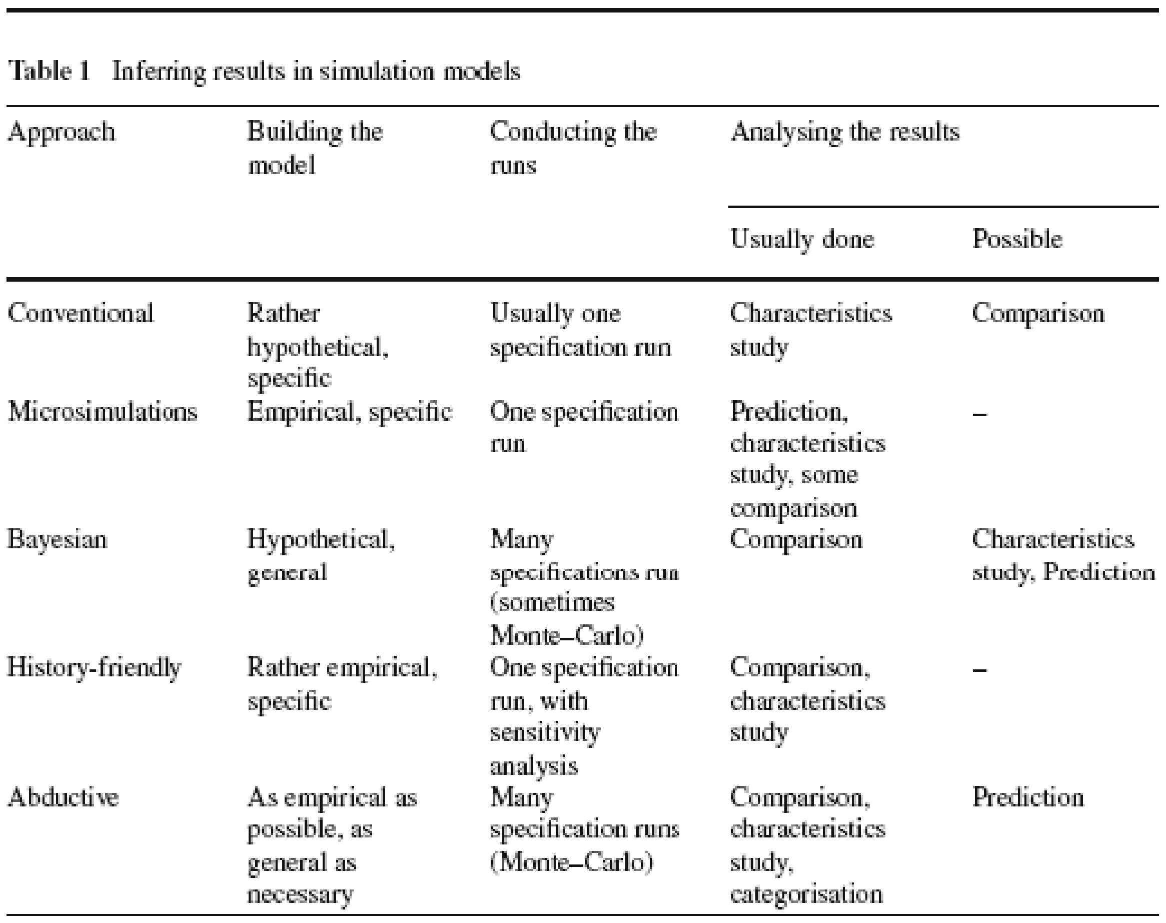


Table 1 Inferring results in simulation models

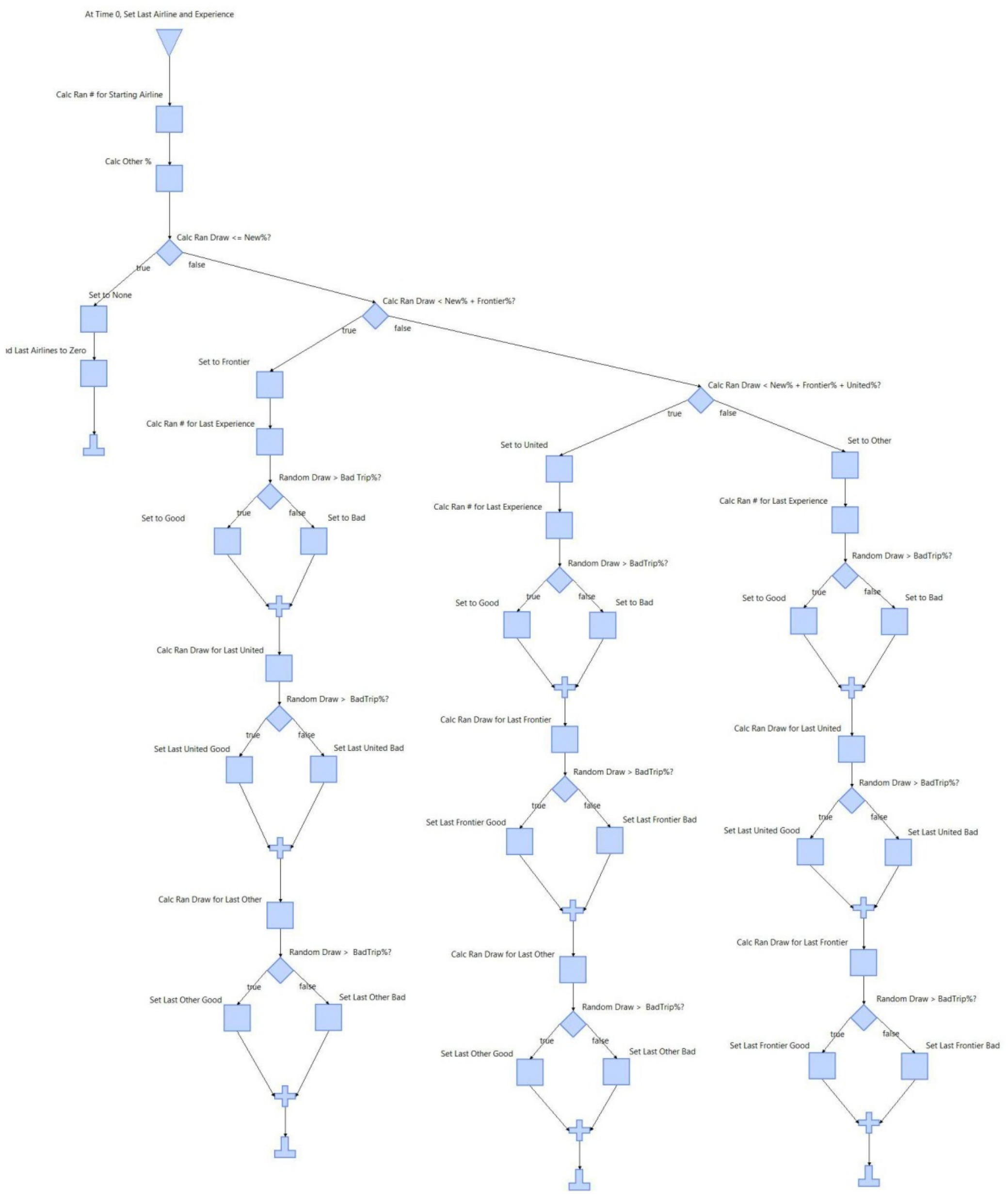
Approach	Building the model	Conducting the runs	Analysing the results	
			Usually done	Possible
Conventional	Rather hypothetical, specific	Usually one specification run	Characteristics study	Comparison
Microsimulations	Empirical, specific	One specification run	Prediction, characteristics study, some comparison	–
Bayesian	Hypothetical, general	Many specifications run (sometimes Monte–Carlo)	Comparison	Characteristics study, Prediction
History-friendly	Rather empirical, specific	One specification run, with sensitivity analysis	Comparison, characteristics study	–
Abductive	As empirical as possible, as general as necessary	Many specification runs (Monte–Carlo)	Comparison, characteristics study, categorisation	Prediction

APPENDIX E: LIST OF MODEL ASSUMPTIONS

1. Consumers travel only once per week and experiences (good or bad) reflect a single trip (one-way or round).
2. New travelers comprise 0-10% of all travelers.
3. A large majority of travelers encounter a bad experience (randomized range = 70-100%).
4. Once assigned a last airline and associated experience (good = 0 or bad = 1) at Time 0, travelers will be assigned an experience for the other airlines regardless if they have flown that airline. A value of 1 represents either an actual bad experience or no experience at all.
5. Travelers with a good last experience stay loyal to the airline for the next trip.
6. Less frequent travelers have a higher chance to switch airlines after experiencing a bad trip than travelers who travel more frequently.
7. Advertising levels for Frontier Airlines are more crucial and have a larger effect on consumers than that for United and the collective Other airlines group.
8. Airlines evaluate advertising levels for adjustment every two weeks and once a quarter (every 12 weeks) after identifying current market share.
9. Airlines review their current market share each quarter and adjust advertising levels based on a predetermined desired market share.
10. Due to inflation, fuel costs have a higher likelihood of rising and thus a greater chance to be set at high in the model.
11. Due to the volatility in the world today (i.e. terrorism), the cost to adhere to federal regulation has a higher likelihood of rising and thus a greater chance to be set at high in the model.

12. An even chance exists (50-50%) that credit availability will be lowered when creditors evaluate the creditworthiness of airlines.
13. Environmental agents potentially can change values on a scheduled basis: Fuel Cost every four weeks, Regulation every 12 weeks, and Credit Availability every four weeks.
14. A simulated market with 1,000; 10,000; and 100,000 consumer agents accurately represents the broader Frontier airline market (~$41 million passengers in 2006).

APPENDIX F: LAST AIRLINE AND LAST EXPERIENCE ASSIGNMENT


At Time 0, Set Last Airline and Experience
Calc Ran # for Starting Airline
Calc Other %
Calc Ran Draw <= New%?
true
false
Set to None
d Last Airlines to Zero
Calc Ran Draw < New% + Frontier%?
true
false
Set to Frontier
Calc Ran # for Last Experience
Random Draw > Bad Trip%?
true
false
Set to Good
Set to Bad
Calc Ran Draw for Last United
Random Draw > BadTrip%?
true
false
Set Last United Good
Set Last United Bad
Calc Ran Draw for Last Other
Random Draw > BadTrip%?
true
false
Set Last Other Good
Set Last Other Bad
Calc Ran Draw < New% + Frontier% + United%?
true
false
Set to United
Calc Ran # for Last Experience
Random Draw > BadTrip%?
true
false
Set to Good
Set to Bad
Calc Ran Draw for Last Frontier
Random Draw > BadTrip%?
true
false
Set Last Frontier Good
Set Last Frontier Bad
Calc Ran Draw for Last Other
Random Draw > BadTrip%?
true
false
Set Last Other Good
Set Last Other Bad
Set to Other
Calc Ran # for Last Experience
Random Draw > BadTrip%?
true
false
Set to Good
Set to Bad
Calc Ran Draw for Last United
Random Draw > BadTrip%?
true
false
Set Last United Good
Set Last United Bad
Calc Ran Draw for Last Frontier
Random Draw > BadTrip%?
true
false
Set Last Frontier Good
Set Last Frontier Bad

APPENDIX G: JAVA CODE

I. Model Initializer

```
/**
 *
 * This file was automatically generated by the Repast Simphony Agent Editor.
 * Please see http://repast.sourceforge.net/ for details.
 *
 */

/**
 *
 * Set the package name.
 *
 */
package frontierairlines

/**
 *
 * Import the needed packages.
 *
 */
import java.io.*
import java.math.*
import java.util.*
import javax.measure.unit.*
import org.jscience.mathematics.number.*
import org.jscience.mathematics.vector.*
import org.jscience.physics.amount.*
import repast.simphony.adaptation.neural.*
import repast.simphony.adaptation.regression.*
import repast.simphony.context.*
import repast.simphony.context.space.continuous.*
import repast.simphony.context.space.gis.*
import repast.simphony.context.space.graph.*
import repast.simphony.context.space.grid.*
import repast.simphony.engine.environment.*
import repast.simphony.engine.schedule.*
import repast.simphony.engine.watcher.*
import repast.simphony.groovy.math.*
import repast.simphony.integration.*
import repast.simphony.matlab.link.*
import repast.simphony.query.*
import repast.simphony.query.space.continuous.*
import repast.simphony.query.space.gis.*
import repast.simphony.query.space.graph.*
import repast.simphony.query.space.grid.*
import repast.simphony.query.space.projection.*
import repast.simphony.parameter.*
import repast.simphony.random.*
import repast.simphony.space.continuous.*
import repast.simphony.space.gis.*
import repast.simphony.space.graph.*
import repast.simphony.space.grid.*
import repast.simphony.space.projection.*
import repast.simphony.ui.probe.*
import repast.simphony.util.*
import simphony.util.messages.*
import static java.lang.Math.*
import static repast.simphony.essentials.RepastEssentials.*

/**
 *
 * This is an agent.
 *
 */
public class ModelInitializer  {

    /**
     *
     * This is an agent property.
     * @field TotalConsumers
     *
     */
    @Parameter (displayName = "Total # of Consumers", usageName = "TotalConsumers")
    public def getTotalConsumers() {
        return TotalConsumers
    }
    public void setTotalConsumers(def newValue) {
        TotalConsumers = newValue
    }
    public def TotalConsumers = 1000

    /**
     *
     * This is an agent property.
     * @field C1
     *
     */
    @Parameter (displayName = "Consumer 1's", usageName = "C1")
    public def getC1() {
        return C1
    }
    public void setC1(def newValue) {
        C1 = newValue
    }
    public def C1 = TotalConsumers * 0.645

    /**
     *
     * This is an agent property.
     * @field C2
     *
     */
    @Parameter (displayName = "Consumer 2's", usageName = "C2")
    public def getC2() {
        return C2
}
public void setC2(def newValue) {
   C2 = newValue
}
public def C2 = TotalConsumers * 0.149

/**
 *
 * This is an agent property.
 * @field C3
 *
 */
@Parameter (displayName = "Consumer 3's", usageName = "C3")
public def getC3() {
   return C3
}
public void setC3(def newValue) {
   C3 = newValue
}
public def C3 = TotalConsumers * 0.051

/**
 *
 * This is an agent property.
 * @field C4
 *
 */
@Parameter (displayName = "Consumer 4's", usageName = "C4")
public def getC4() {
   return C4
}
public void setC4(def newValue) {
   C4 = newValue
}
public def C4 = TotalConsumers * 0.023

/**
 *
 * This is an agent property.
 * @field C5
 *
 */
@Parameter (displayName = "Consumer 5's", usageName = "C5")
public def getC5() {
   return C5
}
public void setC5(def newValue) {
   C5 = newValue
}
public def C5 = TotalConsumers * 0.012

/**
 *
 * This is an agent property.
 * @field C6
 *
 */
@Parameter (displayName = "Consumer 6's", usageName = "C6")
public def getC6() {
   return C6
}
public void setC6(def newValue) {
   C6 = newValue
}
public def C6 = TotalConsumers * 0.008

/**
 *
 * This is an agent property.
 * @field C7
 *
 */
@Parameter (displayName = "Consumer 7's", usageName = "C7")
public def getC7() {
   return C7
}
public void setC7(def newValue) {
   C7 = newValue
}
public def C7 = TotalConsumers * 0.005

/**
 *
 * This is an agent property.
 * @field C8
 *
 */
@Parameter (displayName = "Consumer 8's", usageName = "C8")
public def getC8() {
   return C8
}
public void setC8(def newValue) {
   C8 = newValue
}
public def C8 = TotalConsumers * 0.004

/**
 *
 * This is an agent property.
 * @field C9
 *
 */
@Parameter (displayName = "Consumer 9's", usageName = "C9")
public def getC9() {
   return C9
}
```

```
public void setC9(def newValue) {
    C9 = newValue
}
public def C9 = TotalConsumers * 0.004

/**
 *
 * This is an agent property.
 * @field C0
 *
 */
@Parameter (displayName = "Consumer 0's", usageName = "C0")
public def getC0() {
    return C0
}
public void setC0(def newValue) {
    C0 = newValue
}
public def C0 = TotalConsumers * 0.10

/**
 *
 * This value is used to automatically generate agent identifiers.
 * @field serialVersionUID
 *
 */
private static final long serialVersionUID = 1L

/**
 *
 * This value is used to automatically generate agent identifiers.
 * @field agentIDCounter
 *
 */
protected static long agentIDCounter = 1

/**
 *
 * This value is the agent's identifier.
 * @field agentID
 *
 */
protected String agentID = "ModelInitializer " + (agentIDCounter++)

/**
 *
 * This is the user model builder
 * @method initializeModel
 *
 */
@ScheduledMethod(
    start = 0d,
    priority = 1.7976931348623157E308d,
    shuffle = true
)
public static def initializeModel() {

    // Define the return value variable.
    def returnValue

    // Note the simulation time.
    def time = GetTickCountInTimeUnits()


    // This is a loop.
    for (i in 1..7170) {

        // This is a task.
        Consumer C = new Consumer()
        AddAgentToContext("FrontierAirlines", C)
        // This is a task.
        C.Frequency = 0.08
        // This is a task.
        C.MemStrength = 1

    }


    // This is a loop.
    for (i in 1..1650) {

        // This is a task.
        Consumer C = new Consumer()
        AddAgentToContext("FrontierAirlines", C)
        // This is a task.
        C.Frequency = 0.17
        // This is a task.
        C.MemStrength = 2

    }


    // This is a loop.
    for (i in 1..570) {

        // This is a task.
        Consumer C = new Consumer()
        AddAgentToContext("FrontierAirlines", C)
        // This is a task.
        C.Frequency = 0.25
        // This is a task.
        C.MemStrength = 3

    }
```

```
// This is a loop.
for (i in 1..260) {

    // This is a task.
    Consumer C = new Consumer()
    AddAgentToContext("FrontierAirlines", C)
    // This is a task.
    C.Frequency = 0.33
    // This is a task.
    C.MemStrength = 4

}


// This is a loop.
for (i in 1..130) {

    // This is a task.
    Consumer C = new Consumer()
    AddAgentToContext("FrontierAirlines", C)
    // This is a task.
    C.Frequency = 0.42
    // This is a task.
    C.MemStrength = 5

}


// This is a loop.
for (i in 1..90) {

    // This is a task.
    Consumer C = new Consumer()
    AddAgentToContext("FrontierAirlines", C)
    // This is a task.
    C.Frequency = 0.50
    // This is a task.
    C.MemStrength = 6

}


// This is a loop.
for (i in 1..50) {

    // This is a task.
    Consumer C = new Consumer()
    AddAgentToContext("FrontierAirlines", C)
    // This is a task.
    C.Frequency = 0.58
    // This is a task.
    C.MemStrength = 7
    }


    // This is a loop.
    for (i in 1..40) {

        // This is a task.
        Consumer C = new Consumer()
        AddAgentToContext("FrontierAirlines", C)
        // This is a task.
        C.Frequency = 0.67
        // This is a task.
        C.MemStrength = 8

    }


    // This is a loop.
    for (i in 1..40) {

        // This is a task.
        Consumer C = new Consumer()
        AddAgentToContext("FrontierAirlines", C)
        // This is a task.
        C.Frequency = 0.75
        // This is a task.
        C.MemStrength = 9

    }

    // This is a task.
    Fuel F = new Fuel()
    AddAgentToContext("FrontierAirlines", F)
    // This is a task.
    Regulation R = new Regulation()
    AddAgentToContext("FrontierAirlines", R)
    // This is a task.
    Credit CR = new Credit()
    AddAgentToContext("FrontierAirlines", CR)
    // This is a task.
    United U = new United()
    AddAgentToContext("FrontierAirlines", U)
    // This is a task.
    Frontier FR = new Frontier()
    AddAgentToContext("FrontierAirlines", FR)
    println "Initializer Done"
    // This is a task.
    EndSimulationRunAt(52)
    Object agent = CreateAgent("FrontierAirlines", "frontierairlines.MKT")
    // Return the results.
    return returnValue

}
```

```
    /**
     *
     * This method provides a human-readable name for the agent.
     * @method toString
     *
     */
    @ProbeID()
    public String toString() {

        // Define the return value variable.
        def returnValue

        // Note the simulation time.
        def time = GetTickCountInTimeUnits()

        // Set the default agent identifier.
        returnValue = this.agentID
        // Return the results.
        return returnValue

    }
}
```

II. Consumer

```
/**
 *
 * This file was automatically generated by the Repast Simphony Agent Editor.
 * Please see http://repast.sourceforge.net/ for details.
 *
 */

/**
 *
 * Set the package name.
 *
 */
package frontierairlines

/**
 *
 * Import the needed packages.
 *
 */
import java.io.*
import java.math.*
import java.util.*
import javax.measure.unit.*
import org.jscience.mathematics.number.*
import org.jscience.mathematics.vector.*
import org.jscience.physics.amount.*
import repast.simphony.adaptation.neural.*
import repast.simphony.adaptation.regression.*
import repast.simphony.context.*
import repast.simphony.context.space.continuous.*
import repast.simphony.context.space.gis.*
import repast.simphony.context.space.graph.*
import repast.simphony.context.space.grid.*
import repast.simphony.engine.environment.*
import repast.simphony.engine.schedule.*
import repast.simphony.engine.watcher.*
import repast.simphony.groovy.math.*
import repast.simphony.integration.*
import repast.simphony.matlab.link.*
import repast.simphony.query.*
import repast.simphony.query.space.continuous.*
import repast.simphony.query.space.gis.*
import repast.simphony.query.space.graph.*
import repast.simphony.query.space.grid.*
import repast.simphony.query.space.projection.*
import repast.simphony.parameter.*
import repast.simphony.random.*
import repast.simphony.space.continuous.*
import repast.simphony.space.gis.*
import repast.simphony.space.graph.*
import repast.simphony.space.grid.*
import repast.simphony.space.projection.*
import repast.simphony.ui.probe.*
import repast.simphony.util.*
import simphony.util.messages.*
import static java.lang.Math.*
import static repast.simphony.essentials.RepastEssentials.*

/**
 *
 * This is an agent.
 *
 */
public class Consumer  {

    /**
     *
     * This is an agent property.
     * @field LastAirline
     *
     */
    @Parameter (displayName = "Last Airline Chosen", usageName = "LastAirline")
    public def getLastAirline() {
        return LastAirline
    }
    public void setLastAirline(def newValue) {
        LastAirline = newValue
    }
    public def LastAirline = 0
```

```
/**
 *
 * This is an agent property.
 * @field LastExperience
 *
 */
@Parameter (displayName = "Last Experience", usageName = "LastExperience")
public def getLastExperience() {
    return LastExperience
}
public void setLastExperience(def newValue) {
    LastExperience = newValue
}
public def LastExperience = 0

/**
 *
 * This is an agent property.
 * @field Frequency
 *
 */
@Parameter (displayName = "Frequency Factor", usageName = "Frequency")
public def getFrequency() {
    return Frequency
}
public void setFrequency(def newValue) {
    Frequency = newValue
}
public def Frequency = 0

/**
 *
 * This is an agent property.
 * @field MemStrength
 *
 */
@Parameter (displayName = "Strength of Memory", usageName = "MemStrength")
public def getMemStrength() {
    return MemStrength
}
public void setMemStrength(def newValue) {
    MemStrength = newValue
}
public def MemStrength = 0

/**
 *
 * This is an agent property.
 * @field LastUnited
 *
 */
@Parameter (displayName = "Last United Experience", usageName = "LastUnited")
public def getLastUnited() {
    return LastUnited
}
public void setLastUnited(def newValue) {
    LastUnited = newValue
}
public def LastUnited = 0

/**
 *
 * This is an agent property.
 * @field LastFrontier
 *
 */
@Parameter (displayName = "Last Frontier Experience", usageName = "LastFrontier")
public def getLastFrontier() {
    return LastFrontier
}
public void setLastFrontier(def newValue) {
    LastFrontier = newValue
}
public def LastFrontier = 0

/**
 *
 * This is an agent property.
 * @field LastOther
 *
 */
@Parameter (displayName = "Last Other Experience", usageName = "LastOther")
public def getLastOther() {
    return LastOther
}
public void setLastOther(def newValue) {
    LastOther = newValue
}
public def LastOther = 0

/**
 *
 * This is an agent property.
 * @field DrawToFly
 *
 */
@Parameter (displayName = "Random Draw To Fly", usageName = "DrawToFly")
public def getDrawToFly() {
    return DrawToFly
}
public void setDrawToFly(def newValue) {
    DrawToFly = newValue
}
public def DrawToFly = 0

/**
```

```
 *
 * This is an agent property.
 * @field ToFly
 *
 */
@Parameter (displayName = "To Fly", usageName = "ToFly")
public def getToFly() {
    return ToFly
}
public void setToFly(def newValue) {
    ToFly = newValue
}
public def ToFly = 0

/**
 *
 * This is an agent property.
 * @field DrawLastAirline
 *
 */
@Parameter (displayName = "Random Draw Last Airline", usageName = "DrawLastAirline")
public def getDrawLastAirline() {
    return DrawLastAirline
}
public void setDrawLastAirline(def newValue) {
    DrawLastAirline = newValue
}
public def DrawLastAirline = 0

/**
 *
 * This is an agent property.
 * @field DrawLastExperience
 *
 */
@Parameter (displayName = "Random Draw Last Experience", usageName = "DrawLastExperience")
public def getDrawLastExperience() {
    return DrawLastExperience
}
public void setDrawLastExperience(def newValue) {
    DrawLastExperience = newValue
}
public def DrawLastExperience = 0

/**
 *
 * This is an agent property.
 * @field DrawLastUnited
 *
 */
@Parameter (displayName = "Random Draw Last United", usageName = "DrawLastUnited")
public def getDrawLastUnited() {
    return DrawLastUnited
}
public void setDrawLastUnited(def newValue) {
    DrawLastUnited = newValue
}
public def DrawLastUnited = 0

/**
 *
 * This is an agent property.
 * @field DrawLastFrontier
 *
 */
@Parameter (displayName = "Random Draw Last Frontier", usageName = "DrawLastFrontier")
public def getDrawLastFrontier() {
    return DrawLastFrontier
}
public void setDrawLastFrontier(def newValue) {
    DrawLastFrontier = newValue
}
public def DrawLastFrontier = 0

/**
 *
 * This is an agent property.
 * @field DrawLastOther
 *
 */
@Parameter (displayName = "Random Draw Last Other", usageName = "DrawLastOther")
public def getDrawLastOther() {
    return DrawLastOther
}
public void setDrawLastOther(def newValue) {
    DrawLastOther = newValue
}
public def DrawLastOther = 0

/**
 *
 * This is an agent property.
 * @field DrawMedSwitch
 *
 */
@Parameter (displayName = "Med Memory Switch", usageName = "DrawMedSwitch")
public def getDrawMedSwitch() {
    return DrawMedSwitch
}
public void setDrawMedSwitch(def newValue) {
    DrawMedSwitch = newValue
}
public def DrawMedSwitch = 0

/**
 *
 * This is an agent property.
 * @field DrawOtherAd1
 *
 */
@Parameter (displayName = "Random Draw Other Ad #1", usageName = "DrawOtherAd1")
public def getDrawOtherAd1() {
   return DrawOtherAd1
}
public void setDrawOtherAd1(def newValue) {
   DrawOtherAd1 = newValue
}
public def DrawOtherAd1 = 0

/**
 *
 * This is an agent property.
 * @field DrawNewFly
 *
 */
@Parameter (displayName = "Random Draw New Flyer", usageName = "DrawNewFly")
public def getDrawNewFly() {
   return DrawNewFly
}
public void setDrawNewFly(def newValue) {
   DrawNewFly = newValue
}
public def DrawNewFly = 0

/**
 *
 * This is an agent property.
 * @field Switch
 *
 */
@Parameter (displayName = "Switch", usageName = "Switch")
public def getSwitch() {
   return Switch
}
public void setSwitch(def newValue) {
   Switch = newValue
}
public def Switch = 0

/**
 *
 * This is an agent property.
 * @field DrawAdHigh
 *
 */
@Parameter (displayName = "Random Draw High Ads", usageName = "DrawAdHigh")
public def getDrawAdHigh() {
   return DrawAdHigh
}
```

```
public void setDrawAdHigh(def newValue) {
    DrawAdHigh = newValue
}
public def DrawAdHigh = 0

/**
 *
 * This is an agent property.
 * @field DrawHighSwitch
 *
 */
@Parameter (displayName = "Random Draw High Mem Switches", usageName = "DrawHighSwitch")
public def getDrawHighSwitch() {
    return DrawHighSwitch
}
public void setDrawHighSwitch(def newValue) {
    DrawHighSwitch = newValue
}
public def DrawHighSwitch = 0

/**
 *
 * This is an agent property.
 * @field DrawOtherStart
 *
 */
@Parameter (displayName = "Random Draw Other Start", usageName = "DrawOtherStart")
public def getDrawOtherStart() {
    return DrawOtherStart
}
public void setDrawOtherStart(def newValue) {
    DrawOtherStart = newValue
}
public def DrawOtherStart = 0

/**
 *
 * This is an agent property.
 * @field AdPer
 *
 */
@Parameter (displayName = "Advertising Percentage", usageName = "AdPer")
public def getAdPer() {
    return AdPer
}
public void setAdPer(def newValue) {
    AdPer = newValue
}
public def AdPer = 0

/**
 *
 * This is an agent property.
 * @field DrawOtherAd2
 *
 */
@Parameter (displayName = "Random Draw Other Ad #2", usageName = "DrawOtherAd2")
public def getDrawOtherAd2() {
   return DrawOtherAd2
}
public void setDrawOtherAd2(def newValue) {
   DrawOtherAd2 = newValue
}
public def DrawOtherAd2 = 0

/**
 *
 * This is an agent property.
 * @field DrawNewStart
 *
 */
@Parameter (displayName = "Random Draw New Start", usageName = "DrawNewStart")
public def getDrawNewStart() {
   return DrawNewStart
}
public void setDrawNewStart(def newValue) {
   DrawNewStart = newValue
}
public def DrawNewStart = 0

/**
 *
 * This is an agent property.
 * @field DrawFrontierStart
 *
 */
@Parameter (displayName = "Random Draw Frontier Start", usageName = "DrawFrontierStart")
public def getDrawFrontierStart() {
   return DrawFrontierStart
}
public void setDrawFrontierStart(def newValue) {
   DrawFrontierStart = newValue
}
public def DrawFrontierStart = 0

/**
 *
 * This is an agent property.
 * @field DrawUnitedStart
 *
 */
@Parameter (displayName = "Random Draw United Start", usageName = "DrawUnitedStart")
public def getDrawUnitedStart() {
   return DrawUnitedStart
}
public void setDrawUnitedStart(def newValue) {
    DrawUnitedStart = newValue
}
public def DrawUnitedStart = 0

/**
 *
 * This is an agent property.
 * @field DrawLastGoodSwitch
 *
 */
@Parameter (displayName = "Random Draw Last Good Switches", usageName = "DrawLastGoodSwitch")
public def getDrawLastGoodSwitch() {
    return DrawLastGoodSwitch
}
public void setDrawLastGoodSwitch(def newValue) {
    DrawLastGoodSwitch = newValue
}
public def DrawLastGoodSwitch = 0

/**
 *
 * This is an agent property.
 * @field Switch2
 *
 */
@Parameter (displayName = "Switch2", usageName = "Switch2")
public def getSwitch2() {
    return Switch2
}
public void setSwitch2(def newValue) {
    Switch2 = newValue
}
public def Switch2 = 0

/**
 *
 * This value is used to automatically generate agent identifiers.
 * @field serialVersionUID
 *
 */
private static final long serialVersionUID = 1L

/**
 *
 * This value is used to automatically generate agent identifiers.
 * @field agentIDCounter
 *
 */
protected static long agentIDCounter = 1

/**
 *
 * This value is the agent's identifier.
```

```
     * @field agentID
     *
     */
    protected String agentID = "Consumer " + (agentIDCounter++)

    /**
     *
     * This is the step behavior.
     * @method step1
     *
     */
    @ScheduledMethod(
        start = 1d,
        interval = 1d,
        priority = -1.7976931348623157E308d,
        shuffle = true
    )
    public def step1() {

        // Define the return value variable.
        def returnValue

        // Note the simulation time.
        def time = GetTickCountInTimeUnits()


        // This is an agent decision.
        if (LastAirline == 0) {

            // This is a task.
            setDrawToFly(RandomDraw())

            // This is an agent decision.
            if (DrawToFly < Frequency) {

                // This is a task.
                setToFly(1)
                MKT.NewFlyer += 1
                // This is a task.
                setDrawNewFly(RandomDraw())

                // This is an agent decision.
                if (DrawNewFly <= (DrawFrontierStart + DrawUnitedStart) + ((DrawNewStart/3)*2) && DrawNewFly >
(DrawFrontierStart + (DrawNewStart /3))) {

                    // This is a task.
                    setLastAirline(2)
                    MKT.unitedMKT += 1
                    MKT.totalMKT += 1
                    setDrawLastExperience(RandomDraw())
                    MKT.NewFlyerUnited += 1

                    // This is an agent decision.
        if (DrawLastExperience > MKT.BadTrip) {

            // This is a task.
            setLastExperience(0)
            setLastUnited(0)

        } else  {

            // This is a task.
            setLastExperience(1)
            setLastUnited(1)

        }

    } else  {


        // This is an agent decision.
        if (DrawNewFly <= DrawFrontierStart + (DrawNewStart/3)) {

            // This is a task.
            setLastAirline(1)
            MKT.frontierMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.NewFlyerFrontier += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastFrontier(0)

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastFrontier(1)

            }

        } else  {

            // This is a task.
            setLastAirline(3)
            MKT.otherMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.NewFlyerOther += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {
                // This is a task.
                setLastExperience(0)
                setLastOther(0)

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastOther(1)

            }

        }

    }

  } else  {

      // This is a task.
      setToFly(0)

  }

} else  {

  // This is a task.
  setDrawToFly(RandomDraw())

  // This is an agent decision.
  if (DrawToFly < Frequency) {

      // This is a task.
      setToFly(1)
      MKT.PastFlyer += 1
      setDrawLastGoodSwitch(RandomDraw())

      // This is an agent decision.
      if (LastExperience == 0 && DrawLastGoodSwitch <= 0.75) {


          // This is an agent decision.
          if (LastAirline == 1) {

              // This is a task.
              setLastAirline(1)
              MKT.frontierMKT += 1
              MKT.totalMKT += 1
              setDrawLastExperience(RandomDraw())
              MKT.StayFrontier += 1

              // This is an agent decision.
              if (DrawLastExperience > MKT.BadTrip) {
```

```
        // This is a task.
        setLastExperience(0)
        setLastFrontier(0)

    } else  {

        // This is a task.
        setLastExperience(1)
        setLastFrontier(1)

    }

} else  {


    // This is an agent decision.
    if (LastAirline == 2) {

        // This is a task.
        setLastAirline(2)
        MKT.unitedMKT += 1
        MKT.totalMKT += 1
        setDrawLastExperience(RandomDraw())
        MKT.StayUnited += 1

        // This is an agent decision.
        if (DrawLastExperience > MKT.BadTrip) {

            // This is a task.
            setLastExperience(0)
            setLastUnited(0)

        } else  {

            // This is a task.
            setLastExperience(1)
            setLastUnited(1)

        }

    } else  {

        // This is a task.
        setLastAirline(3)
        MKT.otherMKT += 1
        MKT.totalMKT += 1
        setDrawLastExperience(RandomDraw())
        MKT.StayOther += 1

        // This is an agent decision.
        if (DrawLastExperience > MKT.BadTrip) {
                // This is a task.
                setLastExperience(0)
                setLastOther(0)

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastOther(1)

            }

        }

    }

} else  {

    // This is a task.
    setDrawHighSwitch(RandomHelper.nextIntFromTo(0,1))
    setDrawMedSwitch(RandomDraw())

    // This is an agent decision.
    if ((MemStrength >= 7) && (DrawHighSwitch = 0)) {


        // This is an agent decision.
        if (LastAirline == 1) {

            // This is a task.
            setLastAirline(1)
            MKT.frontierMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.StayFrontier += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastFrontier(0)

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastFrontier(1)

            }

        } else  {
```

```
        // This is an agent decision.
        if (LastAirline == 2) {

            // This is a task.
            setLastAirline(2)
            MKT.unitedMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.StayUnited += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastUnited(0)

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastUnited(1)

            }

        } else  {

            // This is a task.
            setLastAirline(3)
            MKT.otherMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.StayOther += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastOther(0)

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastOther(1)

            }

        }

    }
```

```
} else  {


    // This is an agent decision.
    if ((MemStrength >= 4) && (DrawMedSwitch > 0.75)) {


        // This is an agent decision.
        if (LastAirline == 1) {

            // This is a task.
            setLastAirline(1)
            MKT.frontierMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.StayFrontier += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastFrontier(0)

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastFrontier(1)

            }

        } else  {


            // This is an agent decision.
            if (LastAirline == 2) {

                // This is a task.
                setLastAirline(2)
                MKT.unitedMKT += 1
                MKT.totalMKT += 1
                setDrawLastExperience(RandomDraw())
                MKT.StayUnited += 1

                // This is an agent decision.
                if (DrawLastExperience > MKT.BadTrip) {

                    // This is a task.
                    setLastExperience(0)
                    setLastUnited(0)
            } else  {

                // This is a task.
                setLastExperience(1)
                setLastUnited(1)

            }

        } else  {

            // This is a task.
            setLastAirline(3)
            MKT.otherMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.StayOther += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastOther(0)

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastOther(1)

            }

        }

    }

} else  {


    // This is an agent decision.
    if (Frontier.FrontierAd == 1) {

        // This is a task.
        setAdPer(0.05)

    } else  {

        // This is a task.
        setAdPer(0.00)

    }

    // This is an agent decision.
```

```
if (LastAirline == 1) {

   // This is a task.
   setDrawAdHigh(RandomDraw())

   // This is an agent decision.
   if (Frontier.FrontierAd == 1 && DrawAdHigh <= 0.10) {

      // This is a task.
      setLastAirline(1)
      MKT.frontierMKT += 1
      MKT.totalMKT += 1
      setDrawLastExperience(RandomDraw())
      MKT.StayFrontier += 1

      // This is an agent decision.
      if (DrawLastExperience > MKT.BadTrip) {

         // This is a task.
         setLastExperience(0)
         setLastFrontier(0)

      } else  {

         // This is a task.
         setLastExperience(1)
         setLastFrontier(1)

      }

   } else  {


      // This is an agent decision.
      if (LastUnited == 0 && LastOther == 0) {

         // This is a task.
         setSwitch(RandomDraw())

         // This is an agent decision.
         if (Switch <= (DrawUnitedStart) / (DrawOtherStart + DrawUnitedStart)) {

            // This is a task.
            setLastAirline(2)
            MKT.unitedMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.SwitchToUnited += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

               // This is a task.
            setLastExperience(0)
            setLastUnited(0)
            MKT.SwitchFromFrontier += 1

        } else  {

            // This is a task.
            setLastExperience(1)
            setLastUnited(1)
            MKT.SwitchFromFrontier += 1

        }

    } else  {

        // This is a task.
        setLastAirline(3)
        MKT.otherMKT += 1
        MKT.totalMKT += 1
        setDrawLastExperience(RandomDraw())
        MKT.SwitchToOther += 1

        // This is an agent decision.
        if (DrawLastExperience > MKT.BadTrip) {

            // This is a task.
            setLastExperience(0)
            setLastOther(0)
            MKT.SwitchFromFrontier += 1

        } else  {

            // This is a task.
            setLastExperience(1)
            setLastOther(1)
            MKT.SwitchFromFrontier += 1

        }

    }

} else  {

    // This is a task.
    setSwitch2(RandomDraw())

    // This is an agent decision.
    if (LastUnited == 0 && Switch2 <= 0.75) {

        // This is a task.
        setLastAirline(2)
        MKT.unitedMKT += 1
        MKT.totalMKT += 1
    setDrawLastExperience(RandomDraw())
    MKT.SwitchToUnited += 1

    // This is an agent decision.
    if (DrawLastExperience > MKT.BadTrip) {

       // This is a task.
       setLastExperience(0)
       setLastUnited(0)
       MKT.SwitchFromFrontier += 1

    } else  {

       // This is a task.
       setLastExperience(1)
       setLastUnited(1)
       MKT.SwitchFromFrontier += 1

    }

 } else  {


    // This is an agent decision.
    if (LastOther == 0 && Switch2 <= 0.75) {

       // This is a task.
       setLastAirline(3)
       MKT.otherMKT += 1
       MKT.totalMKT += 1
       setDrawLastExperience(RandomDraw())
       MKT.SwitchToOther += 1

       // This is an agent decision.
       if (DrawLastExperience > MKT.BadTrip) {

          // This is a task.
          setLastExperience(0)
          setLastOther(0)
          MKT.SwitchFromFrontier += 1

       } else  {

          // This is a task.
          setLastExperience(1)
          setLastOther(1)
          MKT.SwitchFromFrontier += 1

       }

    } else  {
```

```
// This is an agent decision.
if (Switch2 <= DrawFrontierStart + (DrawNewStart/3) + AdPer) {

    // This is a task.
    setLastAirline(1)
    MKT.frontierMKT += 1
    MKT.totalMKT += 1
    setDrawLastExperience(RandomDraw())
    MKT.StayFrontier += 1

    // This is an agent decision.
    if (DrawLastExperience > MKT.BadTrip) {

        // This is a task.
        setLastExperience(0)
        setLastFrontier(0)

    } else  {

        // This is a task.
        setLastExperience(1)
        setLastFrontier(1)

    }

} else  {


    // This is an agent decision.
    if (Switch2 <= (DrawFrontierStart + DrawUnitedStart) + ((DrawNewStart/3)*2)) {

        // This is a task.
        setLastAirline(2)
        MKT.unitedMKT += 1
        MKT.totalMKT += 1
        setDrawLastExperience(RandomDraw())
        MKT.SwitchToUnited += 1

        // This is an agent decision.
        if (DrawLastExperience > MKT.BadTrip) {

            // This is a task.
            setLastExperience(0)
            setLastUnited(0)
            MKT.SwitchFromFrontier += 1

        } else  {

            // This is a task.
            setLastExperience(1)
            setLastUnited(1)
            MKT.SwitchFromFrontier += 1
```

```
                    }

                  } else  {

                    // This is a task.
                    setLastAirline(3)
                    MKT.otherMKT += 1
                    MKT.totalMKT += 1
                    setDrawLastExperience(RandomDraw())
                    MKT.SwitchToOther += 1

                    // This is an agent decision.
                    if (DrawLastExperience > MKT.BadTrip) {

                      // This is a task.
                      setLastExperience(0)
                      setLastOther(0)
                      MKT.SwitchFromFrontier += 1

                    } else  {

                      // This is a task.
                      setLastExperience(1)
                      setLastOther(1)
                      MKT.SwitchFromFrontier += 1

                    }

                  }

                }

              }

            }

          }

        }

      } else  {


        // This is an agent decision.
        if (LastAirline == 2) {

          // This is a task.
          setDrawAdHigh(RandomDraw())

          // This is an agent decision.
          if (United.UnitedAd == 1 || DrawAdHigh <= 0.25) {

            // This is a task.
        setLastAirline(2)
        MKT.unitedMKT += 1
        MKT.totalMKT += 1
        setDrawLastExperience(RandomDraw())
        MKT.StayUnited += 1

        // This is an agent decision.
        if (DrawLastExperience > MKT.BadTrip) {

            // This is a task.
            setLastExperience(0)
            setLastUnited(0)

        } else  {

            // This is a task.
            setLastExperience(1)
            setLastUnited(1)

        }

    } else  {


        // This is an agent decision.
        if (LastFrontier == 0 && LastOther == 0) {

            // This is a task.
            setSwitch(RandomDraw())

            // This is an agent decision.
            if (Switch <= (DrawFrontierStart + AdPer) / (DrawOtherStart + DrawFrontierStart)) {

                // This is a task.
                setLastAirline(1)
                MKT.frontierMKT += 1
                MKT.totalMKT += 1
                setDrawLastExperience(RandomDraw())
                MKT.SwitchToFrontier += 1

                // This is an agent decision.
                if (DrawLastExperience > MKT.BadTrip) {

                    // This is a task.
                    setLastExperience(0)
                    setLastFrontier(0)
                    MKT.SwitchFromUnited += 1

                } else  {

                    // This is a task.
                    setLastExperience(1)
                    setLastFrontier(1)
            MKT.SwitchFromUnited += 1

        }

    } else  {

        // This is a task.
        setLastAirline(3)
        MKT.otherMKT += 1
        MKT.totalMKT += 1
        setDrawLastExperience(RandomDraw())
        MKT.SwitchToOther += 1

        // This is an agent decision.
        if (DrawLastExperience > MKT.BadTrip) {

            // This is a task.
            setLastExperience(0)
            setLastOther(0)
            MKT.SwitchFromUnited += 1

        } else  {

            // This is a task.
            setLastExperience(1)
            setLastOther(1)
            MKT.SwitchFromUnited += 1

        }

    }

} else  {

    // This is a task.
    setSwitch2(RandomDraw())

    // This is an agent decision.
    if (LastFrontier == 0 && Switch2 <= 0.75) {

        // This is a task.
        setLastAirline(1)
        MKT.frontierMKT += 1
        MKT.totalMKT += 1
        setDrawLastExperience(RandomDraw())
        MKT.SwitchToFrontier += 1

        // This is an agent decision.
        if (DrawLastExperience > MKT.BadTrip) {

            // This is a task.
            setLastExperience(0)
            setLastFrontier(0)
        MKT.SwitchFromUnited += 1

    } else  {

        // This is a task.
        setLastExperience(1)
        setLastFrontier(1)
        MKT.SwitchFromUnited += 1

    }

} else  {


    // This is an agent decision.
    if (LastOther == 0 && Switch2 <= 0.75) {

        // This is a task.
        setLastAirline(3)
        MKT.otherMKT += 1
        MKT.totalMKT += 1
        setDrawLastExperience(RandomDraw())
        MKT.SwitchToOther += 1

        // This is an agent decision.
        if (DrawLastExperience > MKT.BadTrip) {

            // This is a task.
            setLastExperience(0)
            setLastOther(0)
            MKT.SwitchFromUnited += 1

        } else  {

            // This is a task.
            setLastExperience(1)
            setLastOther(1)
            MKT.SwitchFromUnited += 1

        }

    } else  {


        // This is an agent decision.
        if (Switch2 <= DrawUnitedStart + (DrawNewStart/3)) {

            // This is a task.
            setLastAirline(2)
            MKT.unitedMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.StayUnited += 1
```

```
                        // This is an agent decision.
                        if (DrawLastExperience > MKT.BadTrip) {

                            // This is a task.
                            setLastExperience(0)
                            setLastUnited(0)

                        } else  {

                            // This is a task.
                            setLastExperience(1)
                            setLastUnited(1)

                        }

                    } else  {


                        // This is an agent decision.
                        if (Switch2 <= (DrawFrontierStart + DrawUnitedStart + AdPer) +
((DrawNewStart/3)*2)) {

                            // This is a task.
                            setLastAirline(1)
                            MKT.frontierMKT += 1
                            MKT.totalMKT += 1
                            setDrawLastExperience(RandomDraw())
                            MKT.SwitchToFrontier += 1

                            // This is an agent decision.
                            if (DrawLastExperience > MKT.BadTrip) {

                                // This is a task.
                                setLastExperience(0)
                                setLastFrontier(0)
                                MKT.SwitchFromUnited += 1

                            } else  {

                                // This is a task.
                                setLastExperience(1)
                                setLastFrontier(1)
                                MKT.SwitchFromUnited += 1

                            }

                        } else  {

                            // This is a task.
                            setLastAirline(3)
                            MKT.otherMKT += 1
                            MKT.totalMKT += 1
                            setDrawLastExperience(RandomDraw())
                            MKT.SwitchToOther += 1

                            // This is an agent decision.
                            if (DrawLastExperience > MKT.BadTrip) {

                                // This is a task.
                                setLastExperience(0)
                                setLastFrontier(0)
                                MKT.SwitchFromUnited += 1

                            } else  {

                                // This is a task.
                                setLastExperience(1)
                                setLastFrontier(1)
                                MKT.SwitchFromUnited += 1

                            }

                        }

                    }

                }

            }

        }

    }

} else  {

    // This is a task.
    setDrawOtherAd1(RandomDraw(0.70,0.85))
    setDrawOtherAd2(RandomDraw())

    // This is an agent decision.
    if (DrawOtherAd2 >= DrawOtherAd1) {

        // This is a task.
        setLastAirline(3)
        MKT.otherMKT += 1
        MKT.totalMKT += 1
        setDrawLastExperience(RandomDraw())
        MKT.StayOther += 1

        // This is an agent decision.
        if (DrawLastExperience > MKT.BadTrip) {

            // This is a task.
            setLastExperience(0)
        setLastOther(0)

    } else  {

        // This is a task.
        setLastExperience(1)
        setLastOther(1)

    }

} else  {


    // This is an agent decision.
    if (LastFrontier == 0 && LastUnited == 0) {

        // This is a task.
        setSwitch(RandomDraw())

        // This is an agent decision.
        if (Switch <= (DrawFrontierStart + AdPer) / (DrawUnitedStart + DrawFrontierStart)) {

            // This is a task.
            setLastAirline(1)
            MKT.frontierMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.SwitchToFrontier += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastFrontier(0)
                MKT.SwitchFromOther += 1

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastFrontier(1)
                MKT.SwitchFromOther += 1

            }

        } else  {

            // This is a task.
            setLastAirline(2)
            MKT.unitedMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.SwitchToUnited += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastUnited(0)
                MKT.SwitchFromOther += 1

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastUnited(1)
                MKT.SwitchFromOther += 1

            }

        }

    } else  {

        // This is a task.
        setSwitch2(RandomDraw())

        // This is an agent decision.
        if (LastFrontier == 0 && Switch2 <= 0.75) {

            // This is a task.
            setLastAirline(1)
            MKT.frontierMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.SwitchToFrontier += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastFrontier(0)
                MKT.SwitchFromOther += 1

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastFrontier(1)
                MKT.SwitchFromOther += 1

            }
```

```
} else {


  // This is an agent decision.
  if (LastUnited == 0 && Switch2 <= 0.75) {

    // This is a task.
    setLastAirline(2)
    MKT.unitedMKT += 1
    MKT.totalMKT += 1
    setDrawLastExperience(RandomDraw())
    MKT.SwitchToUnited+= 1

    // This is an agent decision.
    if (DrawLastExperience > MKT.BadTrip) {

      // This is a task.
      setLastExperience(0)
      setLastUnited(0)
      MKT.SwitchFromOther += 1

    } else {

      // This is a task.
      setLastExperience(1)
      setLastUnited(1)
      MKT.SwitchFromOther += 1

    }

  } else {


    // This is an agent decision.
    if (Switch2 > (DrawFrontierStart + DrawUnitedStart) + ((DrawNewStart/3)*2)) {

      // This is a task.
      setLastAirline(3)
      MKT.otherMKT += 1
      MKT.totalMKT += 1
      setDrawLastExperience(RandomDraw())
      MKT.StayOther += 1

      // This is an agent decision.
      if (DrawLastExperience > MKT.BadTrip) {

        // This is a task.
        setLastExperience(0)
        setLastOther(0)

      } else {

        // This is a task.
```

```
            setLastExperience(1)
            setLastOther(1)

        }

    } else  {


        // This is an agent decision.
        if (Switch2 <= DrawFrontierStart + (DrawNewStart/3) + AdPer) {

            // This is a task.
            setLastAirline(1)
            MKT.frontierMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.SwitchToFrontier += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastFrontier(0)
                MKT.SwitchFromOther += 1

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastFrontier(1)
                MKT.SwitchFromOther += 1

            }

        } else  {

            // This is a task.
            setLastAirline(2)
            MKT.unitedMKT += 1
            MKT.totalMKT += 1
            setDrawLastExperience(RandomDraw())
            MKT.SwitchToUnited += 1

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastUnited(0)
                MKT.SwitchFromOther += 1

            } else  {
```

```
                                                            // This is a task.
                                                            setLastExperience(1)
                                                            setLastUnited(1)
                                                            MKT.SwitchFromOther += 1

                                                        }

                                                    }

                                                }

                                            }

                                        }

                                    }

                                }

                            }

                        }

                    }

                }

            }

        } else  {

            // This is a task.
            setToFly(0)

        }

    }
    // Return the results.
    return returnValue

}

/**
 *
 * This is the step behavior.
 * @method step0
 *
 */
@ScheduledMethod(
    start = 0d,
    priority = -1.7976931348623157E308d,
    shuffle = false
```

```
)
public def step0() {

    // Define the return value variable.
    def returnValue

    // Note the simulation time.
    def time = GetTickCountInTimeUnits()

    // This is a task.
    setDrawLastAirline(RandomDraw())
    setDrawNewStart(RandomDraw(0.00, 0.10))
    setDrawFrontierStart(RandomDraw(0.11, 0.21))
    setDrawUnitedStart(RandomDraw(0.16, 0.26))
    MKT.BadTrip = RandomDraw(0.70,1.0)
    // This is a task.
    setDrawOtherStart((1 - DrawNewStart - DrawFrontierStart - DrawUnitedStart))

    // This is an agent decision.
    if (DrawLastAirline <= DrawNewStart) {

        // This is a task.
        setLastAirline(0)
        MKT.StartNone += 1
        // This is a task.
        setLastExperience(0)
        setLastUnited(0)
        setLastFrontier(0)
        setLastOther(0)

    } else  {


        // This is an agent decision.
        if (DrawLastAirline <= (DrawNewStart + DrawFrontierStart)) {

            // This is a task.
            setLastAirline(1)
            MKT.StartFrontier += 1
            // This is a task.
            setDrawLastExperience(RandomDraw())

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastFrontier(0)

            } else  {

                // This is a task.
                setLastExperience(1)
                setLastFrontier(1)

            }
            // This is a task.
            setDrawLastUnited(RandomDraw())

            // This is an agent decision.
            if (DrawLastUnited > MKT.BadTrip) {

                // This is a task.
                setLastUnited(0)

            } else  {

                // This is a task.
                setLastUnited(1)

            }
            // This is a task.
            setDrawLastOther(RandomDraw())

            // This is an agent decision.
            if (DrawLastOther > MKT.BadTrip) {

                // This is a task.
                setLastOther(0)

            } else  {

                // This is a task.
                setLastOther(1)

            }

        } else  {


            // This is an agent decision.
            if (DrawLastAirline <= (DrawNewStart + DrawFrontierStart + DrawUnitedStart)) {

                // This is a task.
                setLastAirline(2)
                MKT.StartUnited += 1
                // This is a task.
                setDrawLastExperience(RandomDraw())

                // This is an agent decision.
                if (DrawLastExperience > MKT.BadTrip) {

                    // This is a task.
                    setLastExperience(0)
                    setLastUnited(0)
            } else  {

                // This is a task.
                setLastExperience(1)
                setLastUnited(1)

            }
            // This is a task.
            setDrawLastFrontier(RandomDraw())

            // This is an agent decision.
            if (DrawLastFrontier > MKT.BadTrip) {

                // This is a task.
                setLastFrontier(0)

            } else  {

                // This is a task.
                setLastFrontier(1)

            }
            // This is a task.
            setDrawLastOther(RandomDraw())

            // This is an agent decision.
            if (DrawLastFrontier > MKT.BadTrip) {

                // This is a task.
                setLastOther(0)

            } else  {

                // This is a task.
                setLastOther(1)

            }

        } else  {

            // This is a task.
            setLastAirline(3)
            MKT.StartOther += 1
            // This is a task.
            setDrawLastExperience(RandomDraw())

            // This is an agent decision.
            if (DrawLastExperience > MKT.BadTrip) {

                // This is a task.
                setLastExperience(0)
                setLastOther(0)
```

```
            } else  {

                // This is a task.
                setLastExperience(1)
                setLastOther(1)

            }
            // This is a task.
            setDrawLastUnited(RandomDraw())

            // This is an agent decision.
            if (DrawLastUnited > MKT.BadTrip) {

                // This is a task.
                setLastUnited(0)

            } else  {

                // This is a task.
                setLastUnited(1)

            }
            // This is a task.
            setDrawLastFrontier(RandomDraw())

            // This is an agent decision.
            if (DrawLastFrontier > MKT.BadTrip) {

                // This is a task.
                setLastFrontier(0)

            } else  {

                // This is a task.
                setLastFrontier(1)

            }

        }

    }

  }
  // Return the results.
  return returnValue

}

/**
 *
 * This method provides a human-readable name for the agent.
 * @method toString
 *
```

```
         */
        @ProbeID()
        public String toString() {

            // Define the return value variable.
            def returnValue

            // Note the simulation time.
            def time = GetTickCountInTimeUnits()

            // Set the default agent identifier.
            returnValue = this.agentID
            // Return the results.
            return returnValue

        }


    }
```

III. Frontier Airlines

```
    /**
     *
     * This file was automatically generated by the Repast Simphony Agent Editor.
     * Please see http://repast.sourceforge.net/ for details.
     *
     */

    /**
     *
     * Set the package name.
     *
     */
    package frontierairlines

    /**
     *
     * Import the needed packages.
     *
     */
    import java.io.*
    import java.math.*
    import java.util.*
    import javax.measure.unit.*
    import org.jscience.mathematics.number.*
    import org.jscience.mathematics.vector.*
    import org.jscience.physics.amount.*
    import repast.simphony.adaptation.neural.*
    import repast.simphony.adaptation.regression.*
    import repast.simphony.context.*
    import repast.simphony.context.space.continuous.*
    import repast.simphony.context.space.gis.*
```

```
import repast.simphony.context.space.graph.*
import repast.simphony.context.space.grid.*
import repast.simphony.engine.environment.*
import repast.simphony.engine.schedule.*
import repast.simphony.engine.watcher.*
import repast.simphony.groovy.math.*
import repast.simphony.integration.*
import repast.simphony.matlab.link.*
import repast.simphony.query.*
import repast.simphony.query.space.continuous.*
import repast.simphony.query.space.gis.*
import repast.simphony.query.space.graph.*
import repast.simphony.query.space.grid.*
import repast.simphony.query.space.projection.*
import repast.simphony.parameter.*
import repast.simphony.random.*
import repast.simphony.space.continuous.*
import repast.simphony.space.gis.*
import repast.simphony.space.graph.*
import repast.simphony.space.grid.*
import repast.simphony.space.projection.*
import repast.simphony.ui.probe.*
import repast.simphony.util.*
import simphony.util.messages.*
import static java.lang.Math.*
import static repast.simphony.essentials.RepastEssentials.*

/**
 *
 * This is an agent.
 *
 */
public class Frontier  {

    /**
     *
     * This is an agent property.
     * @field FrontierAd
     *
     */
    @Parameter (displayName = "Advertising", usageName = "FrontierAd")
    public static def getFrontierAd() {
        return FrontierAd
    }
    public static void setFrontierAd(def newValue) {
        FrontierAd = newValue
    }
    public static def FrontierAd = 0

    /**
     *
     * This is an agent property.
     * @field FrontierMarketShare
 *
 */
@Parameter (displayName = "Frontier Market Share", usageName = "FrontierMarketShare")
public def getFrontierMarketShare() {
    return FrontierMarketShare
}
public void setFrontierMarketShare(def newValue) {
    FrontierMarketShare = newValue
}
public def FrontierMarketShare = 0

/**
 *
 * This value is used to automatically generate agent identifiers.
 * @field serialVersionUID
 *
 */
private static final long serialVersionUID = 1L

/**
 *
 * This value is used to automatically generate agent identifiers.
 * @field agentIDCounter
 *
 */
protected static long agentIDCounter = 1

/**
 *
 * This value is the agent's identifier.
 * @field agentID
 *
 */
protected String agentID = "Frontier " + (agentIDCounter++)

/**
 *
 * This is the step behavior.
 * @method step0
 *
 */
@ScheduledMethod(
    start = 0d,
    interval = 2d,
    priority = 1d,
    shuffle = false
)
public def step0() {

    // Define the return value variable.
    def returnValue

    // Note the simulation time.
```

```
def time = GetTickCountInTimeUnits()


// This is an agent decision.
if (Fuel.fuel == 1) {


    // This is an agent decision.
    if (Regulation.regulation == 1) {


        // This is an agent decision.
        if (Credit.credit == 1) {

            // This is a task.
            setFrontierAd(0)

        } else  {

            // This is a task.
            setFrontierAd(0)

        }

    } else  {


        // This is an agent decision.
        if (Credit.credit == 1) {

            // This is a task.
            setFrontierAd(0)

        } else  {

            // This is a task.
            setFrontierAd(1)

        }

    }

} else  {


    // This is an agent decision.
    if (Regulation.regulation == 1) {


        // This is an agent decision.
        if (Credit.credit == 1) {

            // This is a task.
```

```
                setFrontierAd(0)

            } else  {

                // This is a task.
                setFrontierAd(1)

            }

        } else  {


            // This is an agent decision.
            if (Credit.credit == 1) {

                // This is a task.
                setFrontierAd(1)

            } else  {

                // This is a task.
                setFrontierAd(1)

            }

        }

    }
    // Return the results.
    return returnValue

}

/**
 *
 * This is the step behavior.
 * @method step1
 *
 */
@ScheduledMethod(
    start = 12d,
    interval = 12d,
    priority = 1d,
    shuffle = false
)
public def step1() {

    // Define the return value variable.
    def returnValue

    // Note the simulation time.
    def time = GetTickCountInTimeUnits()
```

```
// This is a task.
MKT.FrontierShare = MKT.frontierMKT / MKT.totalMKT

// This is an agent decision.
if (MKT.FrontierShare <= 0.16) {

    // This is a task.
    MKT.FrontierMktShrBad += 1

    // This is an agent decision.
    if (Fuel.fuel == 1) {


        // This is an agent decision.
        if (Regulation.regulation == 1) {


            // This is an agent decision.
            if (Credit.credit == 1) {

                // This is a task.
                setFrontierAd(0)

            } else  {

                // This is a task.
                setFrontierAd(0)

            }

        } else  {


            // This is an agent decision.
            if (Credit.credit == 1) {

                // This is a task.
                setFrontierAd(0)

            } else  {

                // This is a task.
                setFrontierAd(1)

            }

        }

    } else  {


        // This is an agent decision.
        if (Regulation.regulation == 1) {
                // This is an agent decision.
                if (Credit.credit == 1) {

                    // This is a task.
                    setFrontierAd(0)

                } else  {

                    // This is a task.
                    setFrontierAd(1)

                }

            } else  {


                // This is an agent decision.
                if (Credit.credit == 1) {

                    // This is a task.
                    setFrontierAd(1)

                } else  {

                    // This is a task.
                    setFrontierAd(1)

                }

            }

        }

    } else  {

        // This is a task.
        MKT.FrontierMktShrGood += 1

    }
    // Return the results.
    return returnValue

}

/**
 *
 * This is the step behavior.
 * @method step2
 *
 */
@ScheduledMethod(
        start = 1d,
        interval = 1d,
        priority = -1.7976931348623157E308d,
        shuffle = false
    )
    public def step2() {

        // Define the return value variable.
        def returnValue

        // Note the simulation time.
        def time = GetTickCountInTimeUnits()


        // This is an agent decision.
        if (FrontierAd == 1) {

            // This is a task.
            MKT.FrontierAdHigh += 1

        } else  {

            // This is a task.
            MKT.FrontierAdLow += 1

        }
        // Return the results.
        return returnValue

    }

    /**
     *
     * This method provides a human-readable name for the agent.
     * @method toString
     *
     */
    @ProbeID()
    public String toString() {

        // Define the return value variable.
        def returnValue

        // Note the simulation time.
        def time = GetTickCountInTimeUnits()

        // Set the default agent identifier.
        returnValue = this.agentID
        // Return the results.
        return returnValue

    }
}
```

IV. United Airlines

```
/**
 *
 * This file was automatically generated by the Repast Simphony Agent Editor.
 * Please see http://repast.sourceforge.net/ for details.
 *
 */

/**
 *
 * Set the package name.
 *
 */
package frontierairlines

/**
 *
 * Import the needed packages.
 *
 */
import java.io.*
import java.math.*
import java.util.*
import javax.measure.unit.*
import org.jscience.mathematics.number.*
import org.jscience.mathematics.vector.*
import org.jscience.physics.amount.*
import repast.simphony.adaptation.neural.*
import repast.simphony.adaptation.regression.*
import repast.simphony.context.*
import repast.simphony.context.space.continuous.*
import repast.simphony.context.space.gis.*
import repast.simphony.context.space.graph.*
import repast.simphony.context.space.grid.*
import repast.simphony.engine.environment.*
import repast.simphony.engine.schedule.*
import repast.simphony.engine.watcher.*
import repast.simphony.groovy.math.*
import repast.simphony.integration.*
import repast.simphony.matlab.link.*
import repast.simphony.query.*
import repast.simphony.query.space.continuous.*
import repast.simphony.query.space.gis.*
import repast.simphony.query.space.graph.*
import repast.simphony.query.space.grid.*
import repast.simphony.query.space.projection.*
import repast.simphony.parameter.*
import repast.simphony.random.*
import repast.simphony.space.continuous.*
import repast.simphony.space.gis.*
import repast.simphony.space.graph.*
import repast.simphony.space.grid.*
import repast.simphony.space.projection.*
import repast.simphony.ui.probe.*
import repast.simphony.util.*
import simphony.util.messages.*
import static java.lang.Math.*
import static repast.simphony.essentials.RepastEssentials.*

/**
 *
 * This is an agent.
 *
 */
public class United  {

    /**
     *
     * This is an agent property.
     * @field UnitedAd
     *
     */
    @Parameter (displayName = "Advertising", usageName = "UnitedAd")
    public static def getUnitedAd() {
        return UnitedAd
    }
    public static void setUnitedAd(def newValue) {
        UnitedAd = newValue
    }
    public static def UnitedAd = 0

    /**
     *
     * This is an agent property.
     * @field UnitedMarketShare
     *
     */
    @Parameter (displayName = "United Market Share", usageName = "UnitedMarketShare")
    public def getUnitedMarketShare() {
        return UnitedMarketShare
    }
    public void setUnitedMarketShare(def newValue) {
        UnitedMarketShare = newValue
    }
    public def UnitedMarketShare = 0

    /**
     *
     * This value is used to automatically generate agent identifiers.
     * @field serialVersionUID
     *
     */
    private static final long serialVersionUID = 1L

    /**
 *
 * This value is used to automatically generate agent identifiers.
 * @field agentIDCounter
 *
 */
protected static long agentIDCounter = 1

/**
 *
 * This value is the agent's identifier.
 * @field agentID
 *
 */
protected String agentID = "United " + (agentIDCounter++)

/**
 *
 * This is the step behavior.
 * @method step0
 *
 */
@ScheduledMethod(
    start = 0d,
    interval = 2d,
    priority = 1d,
    shuffle = false
)
public def step0() {

    // Define the return value variable.
    def returnValue

    // Note the simulation time.
    def time = GetTickCountInTimeUnits()


    // This is an agent decision.
    if (Fuel.fuel == 1) {


        // This is an agent decision.
        if (Regulation.regulation == 1) {


            // This is an agent decision.
            if (Credit.credit == 1) {

                // This is a task.
                setUnitedAd(0)

            } else  {

                // This is a task.
            setUnitedAd(0)

          }

        } else  {


          // This is an agent decision.
          if (Credit.credit == 1) {

            // This is a task.
            setUnitedAd(0)

          } else  {

            // This is a task.
            setUnitedAd(0)

          }

        }

      } else  {


        // This is an agent decision.
        if (Regulation.regulation == 1) {


          // This is an agent decision.
          if (Credit.credit == 1) {

            // This is a task.
            setUnitedAd(0)

          } else  {

            // This is a task.
            setUnitedAd(0)

          }

        } else  {


          // This is an agent decision.
          if (Credit.credit == 1) {

            // This is a task.
            setUnitedAd(0)

          } else  {
                // This is a task.
                setUnitedAd(1)

            }

        }

    }
    // Return the results.
    return returnValue

}

/**
 *
 * This is the step behavior.
 * @method step1
 *
 */
@ScheduledMethod(
    start = 12d,
    interval = 12d,
    priority = 1d,
    shuffle = false
)
public def step1() {

    // Define the return value variable.
    def returnValue

    // Note the simulation time.
    def time = GetTickCountInTimeUnits()

    // This is a task.
    MKT.UnitedShare = MKT.unitedMKT / MKT.totalMKT

    // This is an agent decision.
    if (MKT.UnitedShare <= 0.21) {

        // This is a task.
        MKT.UnitedMktShrBad += 1

        // This is an agent decision.
        if (Fuel.fuel == 1) {


            // This is an agent decision.
            if (Regulation.regulation == 1) {


                // This is an agent decision.
                if (Credit.credit == 1) {
```

```
            // This is a task.
            setUnitedAd(0)

        } else  {

            // This is a task.
            setUnitedAd(0)

        }

    } else  {


        // This is an agent decision.
        if (Credit.credit == 1) {

            // This is a task.
            setUnitedAd(0)

        } else  {

            // This is a task.
            setUnitedAd(0)

        }

    }

} else  {


    // This is an agent decision.
    if (Regulation.regulation == 1) {


        // This is an agent decision.
        if (Credit.credit == 1) {

            // This is a task.
            setUnitedAd(0)

        } else  {

            // This is a task.
            setUnitedAd(0)

        }

    } else  {


        // This is an agent decision.
        if (Credit.credit == 1) {
```

```
                    // This is a task.
                    setUnitedAd(0)

                } else  {

                    // This is a task.
                    setUnitedAd(1)

                }

            }

        }

    } else  {

        // This is a task.
        MKT.UnitedMktShrGood += 1

    }
    // Return the results.
    return returnValue

}

/**
 *
 * This is the step behavior.
 * @method step2
 *
 */
@ScheduledMethod(
    start = 1d,
    interval = 1d,
    priority = -1.7976931348623157E308d,
    shuffle = false
)
public def step2() {

    // Define the return value variable.
    def returnValue

    // Note the simulation time.
    def time = GetTickCountInTimeUnits()


    // This is an agent decision.
    if (UnitedAd == 1) {

        // This is a task.
        MKT.UnitedAdHigh += 1
        } else  {

            // This is a task.
            MKT.UnitedAdLow += 1

        }
        // Return the results.
        return returnValue

    }

    /**
     *
     * This method provides a human-readable name for the agent.
     * @method toString
     *
     */
    @ProbeID()
    public String toString() {

        // Define the return value variable.
        def returnValue

        // Note the simulation time.
        def time = GetTickCountInTimeUnits()

        // Set the default agent identifier.
        returnValue = this.agentID
        // Return the results.
        return returnValue

    }
}
```

V. Credit Level

```
/**
 *
 * This file was automatically generated by the Repast Simphony Agent Editor.
 * Please see http://repast.sourceforge.net/ for details.
 *
 */

/**
 *
 * Set the package name.
 *
 */
package frontierairlines

/**
 *
 * Import the needed packages.
 *
 */
import java.io.*
import java.math.*
import java.util.*
import javax.measure.unit.*
import org.jscience.mathematics.number.*
import org.jscience.mathematics.vector.*
import org.jscience.physics.amount.*
import repast.simphony.adaptation.neural.*
import repast.simphony.adaptation.regression.*
import repast.simphony.context.*
import repast.simphony.context.space.continuous.*
import repast.simphony.context.space.gis.*
import repast.simphony.context.space.graph.*
import repast.simphony.context.space.grid.*
import repast.simphony.engine.environment.*
import repast.simphony.engine.schedule.*
import repast.simphony.engine.watcher.*
import repast.simphony.groovy.math.*
import repast.simphony.integration.*
import repast.simphony.matlab.link.*
import repast.simphony.query.*
import repast.simphony.query.space.continuous.*
import repast.simphony.query.space.gis.*
import repast.simphony.query.space.graph.*
import repast.simphony.query.space.grid.*
import repast.simphony.query.space.projection.*
import repast.simphony.parameter.*
import repast.simphony.random.*
import repast.simphony.space.continuous.*
import repast.simphony.space.gis.*
import repast.simphony.space.graph.*
import repast.simphony.space.grid.*
import repast.simphony.space.projection.*
import repast.simphony.ui.probe.*
import repast.simphony.util.*
import simphony.util.messages.*
import static java.lang.Math.*
import static repast.simphony.essentials.RepastEssentials.*

/**
 *
 * This is an agent.
 *
 */
public class Credit  {

    /**
     *
     * This is an agent property.
     * @field Credit
     *
 */
@Parameter (displayName = "Credit Availability", usageName = "Credit")
public static def getCredit() {
    return Credit
}
public static void setCredit(def newValue) {
    Credit = newValue
}
public static def Credit = 0

/**
 *
 * This is an agent property.
 * @field DrawCredit
 *
 */
@Parameter (displayName = "Random Draw Credit", usageName = "DrawCredit")
public def getDrawCredit() {
    return DrawCredit
}
public void setDrawCredit(def newValue) {
    DrawCredit = newValue
}
public def DrawCredit = 0

/**
 *
 * This value is used to automatically generate agent identifiers.
 * @field serialVersionUID
 *
 */
private static final long serialVersionUID = 1L

/**
 *
 * This value is used to automatically generate agent identifiers.
 * @field agentIDCounter
 *
 */
protected static long agentIDCounter = 1

/**
 *
 * This value is the agent's identifier.
 * @field agentID
 *
 */
protected String agentID = "Credit " + (agentIDCounter++)

/**
 *
 * This is the step behavior.
 * @method step
 *
 */
@ScheduledMethod(
    start = 0d,
    interval = 4d,
    priority = 2d,
    shuffle = false
)
public def step() {

    // Define the return value variable.
    def returnValue

    // Note the simulation time.
    def time = GetTickCountInTimeUnits()

    // This is a task.
    setDrawCredit(RandomDraw())

    // This is an agent decision.
    if (DrawCredit > 0.50) {

        // This is a task.
        setCredit(1)

    } else  {

        // This is a task.
        setCredit(0)

    }
    // Return the results.
    return returnValue

}

/**
 *
 * This is the step behavior.
 * @method step1
 *
 */
@ScheduledMethod(
    start = 1d,
    interval = 1d,
    priority = -1.7976931348623157E308d,
    shuffle = false
)
public def step1() {

    // Define the return value variable.
    def returnValue
```

```
        // Note the simulation time.
        def time = GetTickCountInTimeUnits()


        // This is an agent decision.
        if (Credit == 1) {

            // This is a task.
            MKT.CreditHigh += 1

        } else  {

            // This is a task.
            MKT.CreditLow += 1

        }
        // Return the results.
        return returnValue

    }

    /**
     *
     * This method provides a human-readable name for the agent.
     * @method toString
     *
     */
    @ProbeID()
    public String toString() {

        // Define the return value variable.
        def returnValue

        // Note the simulation time.
        def time = GetTickCountInTimeUnits()

        // Set the default agent identifier.
        returnValue = this.agentID
        // Return the results.
        return returnValue

    }
}
```

VI. Fuel Costs

```
/**
 *
 * This file was automatically generated by the Repast Simphony Agent Editor.
 * Please see http://repast.sourceforge.net/ for details.
 *
 */
```

```
/**
 *
 * Set the package name.
 *
 */
package frontierairlines

/**
 *
 * Import the needed packages.
 *
 */
import java.io.*
import java.math.*
import java.util.*
import javax.measure.unit.*
import org.jscience.mathematics.number.*
import org.jscience.mathematics.vector.*
import org.jscience.physics.amount.*
import repast.simphony.adaptation.neural.*
import repast.simphony.adaptation.regression.*
import repast.simphony.context.*
import repast.simphony.context.space.continuous.*
import repast.simphony.context.space.gis.*
import repast.simphony.context.space.graph.*
import repast.simphony.context.space.grid.*
import repast.simphony.engine.environment.*
import repast.simphony.engine.schedule.*
import repast.simphony.engine.watcher.*
import repast.simphony.groovy.math.*
import repast.simphony.integration.*
import repast.simphony.matlab.link.*
import repast.simphony.query.*
import repast.simphony.query.space.continuous.*
import repast.simphony.query.space.gis.*
import repast.simphony.query.space.graph.*
import repast.simphony.query.space.grid.*
import repast.simphony.query.space.projection.*
import repast.simphony.parameter.*
import repast.simphony.random.*
import repast.simphony.space.continuous.*
import repast.simphony.space.gis.*
import repast.simphony.space.graph.*
import repast.simphony.space.grid.*
import repast.simphony.space.projection.*
import repast.simphony.ui.probe.*
import repast.simphony.util.*
import simphony.util.messages.*
import static java.lang.Math.*
import static repast.simphony.essentials.RepastEssentials.*

/**
 *
 * This is an agent.
 *
 */
public class Fuel  {

    /**
     *
     * This is an agent property.
     * @field Fuel
     *
     */
    @Parameter (displayName = "Fuel Cost", usageName = "Fuel")
    public static def getFuel() {
        return Fuel
    }
    public static void setFuel(def newValue) {
        Fuel = newValue
    }
    public static def Fuel = 0

    /**
     *
     * This is an agent property.
     * @field DrawFuelCost
     *
     */
    @Parameter (displayName = "Random Draw Fuel Cost", usageName = "DrawFuelCost")
    public def getDrawFuelCost() {
        return DrawFuelCost
    }
    public void setDrawFuelCost(def newValue) {
        DrawFuelCost = newValue
    }
    public def DrawFuelCost = 0

    /**
     *
     * This value is used to automatically generate agent identifiers.
     * @field serialVersionUID
     *
     */
    private static final long serialVersionUID = 1L

    /**
     *
     * This value is used to automatically generate agent identifiers.
     * @field agentIDCounter
     *
     */
    protected static long agentIDCounter = 1

    /**
     *
     * This value is the agent's identifier.
     * @field agentID
     *
     */
    protected String agentID = "Fuel " + (agentIDCounter++)

    /**
     *
     * This is the step behavior.
     * @method step0
     *
     */
    @ScheduledMethod(
        start = 0d,
        interval = 4d,
        priority = 2d,
        shuffle = false
    )
    public def step0() {

        // Define the return value variable.
        def returnValue

        // Note the simulation time.
        def time = GetTickCountInTimeUnits()

        // This is a task.
        setDrawFuelCost(RandomDraw())

        // This is an agent decision.
        if (DrawFuelCost > 0.40) {

            // This is a task.
            setFuel(1)

        } else  {

            // This is a task.
            setFuel(0)

        }
        // Return the results.
        return returnValue

    }

    /**
     *
     * This is the step behavior.
     * @method step1
     *
     */
    @ScheduledMethod(
            start = 1d,
            interval = 1d,
            priority = -1.7976931348623157E308d,
            shuffle = false
        )
        public def step1() {

            // Define the return value variable.
            def returnValue

            // Note the simulation time.
            def time = GetTickCountInTimeUnits()


            // This is an agent decision.
            if (Fuel == 1) {

                // This is a task.
                MKT.FuelHigh += 1

            } else  {

                // This is a task.
                MKT.FuelLow += 1

            }
            // Return the results.
            return returnValue

        }

        /**
         *
         * This method provides a human-readable name for the agent.
         * @method toString
         *
         */
        @ProbeID()
        public String toString() {

            // Define the return value variable.
            def returnValue

            // Note the simulation time.
            def time = GetTickCountInTimeUnits()

            // Set the default agent identifier.
            returnValue = this.agentID
            // Return the results.
            return returnValue

        }
    }
```

VII. Regulation Level

```
/**
 *
 * This file was automatically generated by the Repast Simphony Agent Editor.
 * Please see http://repast.sourceforge.net/ for details.
 *
 */

/**
 *
 * Set the package name.
 *
 */
package frontierairlines

/**
 *
 * Import the needed packages.
 *
 */
import java.io.*
import java.math.*
import java.util.*
import javax.measure.unit.*
import org.jscience.mathematics.number.*
import org.jscience.mathematics.vector.*
import org.jscience.physics.amount.*
import repast.simphony.adaptation.neural.*
import repast.simphony.adaptation.regression.*
import repast.simphony.context.*
import repast.simphony.context.space.continuous.*
import repast.simphony.context.space.gis.*
import repast.simphony.context.space.graph.*
import repast.simphony.context.space.grid.*
import repast.simphony.engine.environment.*
import repast.simphony.engine.schedule.*
import repast.simphony.engine.watcher.*
import repast.simphony.groovy.math.*
import repast.simphony.integration.*
import repast.simphony.matlab.link.*
import repast.simphony.query.*
import repast.simphony.query.space.continuous.*
import repast.simphony.query.space.gis.*
import repast.simphony.query.space.graph.*
import repast.simphony.query.space.grid.*
import repast.simphony.query.space.projection.*
import repast.simphony.parameter.*
import repast.simphony.random.*
import repast.simphony.space.continuous.*
import repast.simphony.space.gis.*
import repast.simphony.space.graph.*
import repast.simphony.space.grid.*
import repast.simphony.space.projection.*
import repast.simphony.ui.probe.*
import repast.simphony.util.*
import simphony.util.messages.*
import static java.lang.Math.*
import static repast.simphony.essentials.RepastEssentials.*

/**
 *
 * This is an agent.
 *
 */
public class Regulation  {

    /**
     *
     * This is an agent property.
     * @field Regulation
     *
     */
    @Parameter (displayName = "Federal Regulation", usageName = "Regulation")
    public static def getRegulation() {
        return Regulation
    }
    public static void setRegulation(def newValue) {
        Regulation = newValue
    }
    public static def Regulation = 0

    /**
     *
     * This is an agent property.
     * @field DrawRegulation
     *
     */
    @Parameter (displayName = "Random Draw Regulation", usageName = "DrawRegulation")
    public def getDrawRegulation() {
        return DrawRegulation
    }
    public void setDrawRegulation(def newValue) {
        DrawRegulation = newValue
    }
    public def DrawRegulation = 0

    /**
     *
     * This value is used to automatically generate agent identifiers.
     * @field serialVersionUID
     *
     */
    private static final long serialVersionUID = 1L

    /**
 *
 * This value is used to automatically generate agent identifiers.
 * @field agentIDCounter
 *
 */
protected static long agentIDCounter = 1

/**
 *
 * This value is the agent's identifier.
 * @field agentID
 *
 */
protected String agentID = "Regulation " + (agentIDCounter++)

/**
 *
 * This is the step behavior.
 * @method step
 *
 */
@ScheduledMethod(
    start = 0d,
    interval = 12d,
    priority = 2d,
    shuffle = false
)
public def step() {

    // Define the return value variable.
    def returnValue

    // Note the simulation time.
    def time = GetTickCountInTimeUnits()

    // This is a task.
    setDrawRegulation(RandomDraw())

    // This is an agent decision.
    if (DrawRegulation > 0.40) {

        // This is a task.
        setRegulation(1)

    } else  {

        // This is a task.
        setRegulation(0)

    }
    // Return the results.
    return returnValue
```

```
}

/**
 *
 * This is the step behavior.
 * @method step1
 *
 */
@ScheduledMethod(
    start = 1d,
    interval = 1d,
    priority = -1.7976931348623157E308d,
    shuffle = false
)
public def step1() {

    // Define the return value variable.
    def returnValue

    // Note the simulation time.
    def time = GetTickCountInTimeUnits()


    // This is an agent decision.
    if (Regulation == 1) {

        // This is a task.
        MKT.RegHigh += 1

    } else  {

        // This is a task.
        MKT.RegLow += 1

    }
    // Return the results.
    return returnValue

}

/**
 *
 * This method provides a human-readable name for the agent.
 * @method toString
 *
 */
@ProbeID()
public String toString() {

    // Define the return value variable.
    def returnValue

    // Note the simulation time.
```

```
        def time = GetTickCountInTimeUnits()

        // Set the default agent identifier.
        returnValue = this.agentID
        // Return the results.
        return returnValue

    }
}
```

VIII. MKT (used to perform market share calculations)

```
/**
 *
 * This file was automatically generated by the Repast Simphony Agent Editor.
 * Please see http://repast.sourceforge.net/ for details.
 *
 */

/**
 *
 * Set the package name.
 *
 */
package frontierairlines

/**
 *
 * Import the needed packages.
 *
 */
import java.io.*
import java.math.*
import java.util.*
import javax.measure.unit.*
import org.jscience.mathematics.number.*
import org.jscience.mathematics.vector.*
import org.jscience.physics.amount.*
import repast.simphony.adaptation.neural.*
import repast.simphony.adaptation.regression.*
import repast.simphony.context.*
import repast.simphony.context.space.continuous.*
import repast.simphony.context.space.gis.*
import repast.simphony.context.space.graph.*
import repast.simphony.context.space.grid.*
import repast.simphony.engine.environment.*
import repast.simphony.engine.schedule.*
import repast.simphony.engine.watcher.*
import repast.simphony.groovy.math.*
import repast.simphony.integration.*
import repast.simphony.matlab.link.*
import repast.simphony.query.*
import repast.simphony.query.space.continuous.*
import repast.simphony.query.space.gis.*
import repast.simphony.query.space.graph.*
import repast.simphony.query.space.grid.*
import repast.simphony.query.space.projection.*
import repast.simphony.parameter.*
import repast.simphony.random.*
import repast.simphony.space.continuous.*
import repast.simphony.space.gis.*
import repast.simphony.space.graph.*
import repast.simphony.space.grid.*
import repast.simphony.space.projection.*
import repast.simphony.ui.probe.*
import repast.simphony.util.*
import simphony.util.messages.*
import static java.lang.Math.*
import static repast.simphony.essentials.RepastEssentials.*

/**
 *
 * This is an agent.
 *
 */
public class MKT  {

    /**
     *
     * This is an agent property.
     * @field unitedMKT
     *
     */
    @Parameter (displayName = "United Market Tally", usageName = "unitedMKT")
    public static double getUnitedMKT() {
        return unitedMKT
    }
    public static void setUnitedMKT(double newValue) {
        unitedMKT = newValue
    }
    public static double unitedMKT = 0

    /**
     *
     * This is an agent property.
     * @field frontierMKT
     *
     */
    @Parameter (displayName = "Frontier Market Tally", usageName = "frontierMKT")
    public static double getFrontierMKT() {
        return frontierMKT
    }
    public static void setFrontierMKT(double newValue) {
        frontierMKT = newValue
    }
    public static double frontierMKT = 0
```

```
/**
 *
 * This is an agent property.
 * @field otherMKT
 *
 */
@Parameter (displayName = "Other Market Tally", usageName = "otherMKT")
public static double getOtherMKT() {
    return otherMKT
}
public static void setOtherMKT(double newValue) {
    otherMKT = newValue
}
public static double otherMKT = 0

/**
 *
 * This is an agent property.
 * @field totalMKT
 *
 */
@Parameter (displayName = "Total Market Tally", usageName = "totalMKT")
public static double getTotalMKT() {
    return totalMKT
}
public static void setTotalMKT(double newValue) {
    totalMKT = newValue
}
public static double totalMKT = 0

/**
 *
 * This is an agent property.
 * @field UnitedShare
 *
 */
@Parameter (displayName = "United Market Share", usageName = "UnitedShare")
public static double getUnitedShare() {
    return UnitedShare
}
public static void setUnitedShare(double newValue) {
    UnitedShare = newValue
}
public static double UnitedShare = 0

/**
 *
 * This is an agent property.
 * @field FrontierShare
 *
 */
@Parameter (displayName = "Frontier Market Share", usageName = "FrontierShare")
```

```
public static double getFrontierShare() {
    return FrontierShare
}
public static void setFrontierShare(double newValue) {
    FrontierShare = newValue
}
public static double FrontierShare = 0

/**
 *
 * This is an agent property.
 * @field OtherShare
 *
 */
@Parameter (displayName = "Other Market Share", usageName = "OtherShare")
public static double getOtherShare() {
    return OtherShare
}
public static void setOtherShare(double newValue) {
    OtherShare = newValue
}
public static double OtherShare = 0

/**
 *
 * This is an agent property.
 * @field NewFlyer
 *
 */
@Parameter (displayName = "New Flyers", usageName = "NewFlyer")
public static double getNewFlyer() {
    return NewFlyer
}
public static void setNewFlyer(double newValue) {
    NewFlyer = newValue
}
public static double NewFlyer = 0

/**
 *
 * This is an agent property.
 * @field NewFlyerUnited
 *
 */
@Parameter (displayName = "New Flyer United", usageName = "NewFlyerUnited")
public static double getNewFlyerUnited() {
    return NewFlyerUnited
}
public static void setNewFlyerUnited(double newValue) {
    NewFlyerUnited = newValue
}
public static double NewFlyerUnited = 0
```

```
/**
 *
 * This is an agent property.
 * @field NewFlyerFrontier
 *
 */
@Parameter (displayName = "New Flyer Frontier", usageName = "NewFlyerFrontier")
public static double getNewFlyerFrontier() {
   return NewFlyerFrontier
}
public static void setNewFlyerFrontier(double newValue) {
   NewFlyerFrontier = newValue
}
public static double NewFlyerFrontier = 0

/**
 *
 * This is an agent property.
 * @field NewFlyerOther
 *
 */
@Parameter (displayName = "New Flyer Other", usageName = "NewFlyerOther")
public static double getNewFlyerOther() {
   return NewFlyerOther
}
public static void setNewFlyerOther(double newValue) {
   NewFlyerOther = newValue
}
public static double NewFlyerOther = 0

/**
 *
 * This is an agent property.
 * @field PastFlyer
 *
 */
@Parameter (displayName = "Past Flyers", usageName = "PastFlyer")
public static double getPastFlyer() {
   return PastFlyer
}
public static void setPastFlyer(double newValue) {
   PastFlyer = newValue
}
public static double PastFlyer = 0

/**
 *
 * This is an agent property.
 * @field StartFrontier
 *
 */
@Parameter (displayName = "StartFrontier", usageName = "StartFrontier")
public static double getStartFrontier() {
    return StartFrontier
}
public static void setStartFrontier(double newValue) {
    StartFrontier = newValue
}
public static double StartFrontier = 0

/**
 *
 * This is an agent property.
 * @field StartUnited
 *
 */
@Parameter (displayName = "StartUnited", usageName = "StartUnited")
public static double getStartUnited() {
    return StartUnited
}
public static void setStartUnited(double newValue) {
    StartUnited = newValue
}
public static double StartUnited = 0

/**
 *
 * This is an agent property.
 * @field StartOther
 *
 */
@Parameter (displayName = "StartOther", usageName = "StartOther")
public static double getStartOther() {
    return StartOther
}
public static void setStartOther(double newValue) {
    StartOther = newValue
}
public static double StartOther = 0

/**
 *
 * This is an agent property.
 * @field StartNone
 *
 */
@Parameter (displayName = "StartNone", usageName = "StartNone")
public static double getStartNone() {
    return StartNone
}
public static void setStartNone(double newValue) {
    StartNone = newValue
}
public static double StartNone = 0

/**
```

```
 *
 * This is an agent property.
 * @field SwitchToOther
 *
 */
@Parameter (displayName = "Switch to Other", usageName = "SwitchToOther")
public static double getSwitchToOther() {
   return SwitchToOther
}
public static void setSwitchToOther(double newValue) {
   SwitchToOther = newValue
}
public static double SwitchToOther = 0

/**
 *
 * This is an agent property.
 * @field SwitchFromUnited
 *
 */
@Parameter (displayName = "Switch from United", usageName = "SwitchFromUnited")
public static double getSwitchFromUnited() {
   return SwitchFromUnited
}
public static void setSwitchFromUnited(double newValue) {
   SwitchFromUnited = newValue
}
public static double SwitchFromUnited = 0

/**
 *
 * This is an agent property.
 * @field SwitchToFrontier
 *
 */
@Parameter (displayName = "Switch to Frontier", usageName = "SwitchToFrontier")
public static double getSwitchToFrontier() {
   return SwitchToFrontier
}
public static void setSwitchToFrontier(double newValue) {
   SwitchToFrontier = newValue
}
public static double SwitchToFrontier = 0

/**
 *
 * This is an agent property.
 * @field SwitchToUnited
 *
 */
@Parameter (displayName = "Switch to United", usageName = "SwitchToUnited")
public static double getSwitchToUnited() {
   return SwitchToUnited
}
public static void setSwitchToUnited(double newValue) {
    SwitchToUnited = newValue
}
public static double SwitchToUnited = 0

/**
 *
 * This is an agent property.
 * @field SwitchFromOther
 *
 */
@Parameter (displayName = "Switch from Other", usageName = "SwitchFromOther")
public static double getSwitchFromOther() {
    return SwitchFromOther
}
public static void setSwitchFromOther(double newValue) {
    SwitchFromOther = newValue
}
public static double SwitchFromOther = 0

/**
 *
 * This is an agent property.
 * @field SwitchFromFrontier
 *
 */
@Parameter (displayName = "Switch from Frontier", usageName = "SwitchFromFrontier")
public static double getSwitchFromFrontier() {
    return SwitchFromFrontier
}
public static void setSwitchFromFrontier(double newValue) {
    SwitchFromFrontier = newValue
}
public static double SwitchFromFrontier = 0

/**
 *
 * This is an agent property.
 * @field StayFrontier
 *
 */
@Parameter (displayName = "Stay with Frontier", usageName = "StayFrontier")
public static double getStayFrontier() {
    return StayFrontier
}
public static void setStayFrontier(double newValue) {
    StayFrontier = newValue
}
public static double StayFrontier = 0

/**
 *
 * This is an agent property.
 * @field StayUnited
 *
 */
@Parameter (displayName = "Stay with United", usageName = "StayUnited")
public static double getStayUnited() {
    return StayUnited
}
public static void setStayUnited(double newValue) {
    StayUnited = newValue
}
public static double StayUnited = 0

/**
 *
 * This is an agent property.
 * @field StayOther
 *
 */
@Parameter (displayName = "Stay with Other", usageName = "StayOther")
public static double getStayOther() {
    return StayOther
}
public static void setStayOther(double newValue) {
    StayOther = newValue
}
public static double StayOther = 0

/**
 *
 * This is an agent property.
 * @field CreditLow
 *
 */
@Parameter (displayName = "Credit Low", usageName = "CreditLow")
public static double getCreditLow() {
    return CreditLow
}
public static void setCreditLow(double newValue) {
    CreditLow = newValue
}
public static double CreditLow = 0

/**
 *
 * This is an agent property.
 * @field CreditHigh
 *
 */
@Parameter (displayName = "Credit High", usageName = "CreditHigh")
public static double getCreditHigh() {
    return CreditHigh
}
```

```
public static void setCreditHigh(double newValue) {
   CreditHigh = newValue
}
public static double CreditHigh = 0

/**
 *
 * This is an agent property.
 * @field FuelHigh
 *
 */
@Parameter (displayName = "Fuel High", usageName = "FuelHigh")
public static double getFuelHigh() {
   return FuelHigh
}
public static void setFuelHigh(double newValue) {
   FuelHigh = newValue
}
public static double FuelHigh = 0

/**
 *
 * This is an agent property.
 * @field FuelLow
 *
 */
@Parameter (displayName = "Fuel Low", usageName = "FuelLow")
public static double getFuelLow() {
   return FuelLow
}
public static void setFuelLow(double newValue) {
   FuelLow = newValue
}
public static double FuelLow = 0

/**
 *
 * This is an agent property.
 * @field RegHigh
 *
 */
@Parameter (displayName = "Regulation High", usageName = "RegHigh")
public static double getRegHigh() {
   return RegHigh
}
public static void setRegHigh(double newValue) {
   RegHigh = newValue
}
public static double RegHigh = 0

/**
 *
 * This is an agent property.
 * @field RegLow
 *
 */
@Parameter (displayName = "Regulation Low", usageName = "RegLow")
public static double getRegLow() {
   return RegLow
}
public static void setRegLow(double newValue) {
   RegLow = newValue
}
public static double RegLow = 0

/**
 *
 * This is an agent property.
 * @field FrontierAdHigh
 *
 */
@Parameter (displayName = "Frontier Ad High", usageName = "FrontierAdHigh")
public static double getFrontierAdHigh() {
   return FrontierAdHigh
}
public static void setFrontierAdHigh(double newValue) {
   FrontierAdHigh = newValue
}
public static double FrontierAdHigh = 0

/**
 *
 * This is an agent property.
 * @field FrontierAdLow
 *
 */
@Parameter (displayName = "Frontier Ad Low", usageName = "FrontierAdLow")
public static double getFrontierAdLow() {
   return FrontierAdLow
}
public static void setFrontierAdLow(double newValue) {
   FrontierAdLow = newValue
}
public static double FrontierAdLow = 0

/**
 *
 * This is an agent property.
 * @field FrontierMktShrGood
 *
 */
@Parameter (displayName = "Frontier Market Share Good", usageName = "FrontierMktShrGood")
public static double getFrontierMktShrGood() {
   return FrontierMktShrGood
}
public static void setFrontierMktShrGood(double newValue) {
    FrontierMktShrGood = newValue
}
public static double FrontierMktShrGood = 0

/**
 *
 * This is an agent property.
 * @field FrontierMktShrBad
 *
 */
@Parameter (displayName = "Frontier Market Share Bad", usageName = "FrontierMktShrBad")
public static double getFrontierMktShrBad() {
    return FrontierMktShrBad
}
public static void setFrontierMktShrBad(double newValue) {
    FrontierMktShrBad = newValue
}
public static double FrontierMktShrBad = 0

/**
 *
 * This is an agent property.
 * @field UnitedAdHigh
 *
 */
@Parameter (displayName = "United Ad High", usageName = "UnitedAdHigh")
public static double getUnitedAdHigh() {
    return UnitedAdHigh
}
public static void setUnitedAdHigh(double newValue) {
    UnitedAdHigh = newValue
}
public static double UnitedAdHigh = 0

/**
 *
 * This is an agent property.
 * @field UnitedAdLow
 *
 */
@Parameter (displayName = "United Ad Low", usageName = "UnitedAdLow")
public static double getUnitedAdLow() {
    return UnitedAdLow
}
public static void setUnitedAdLow(double newValue) {
    UnitedAdLow = newValue
}
public static double UnitedAdLow = 0

/**
 *
 * This is an agent property.
 * @field UnitedMktShrGood
 *
 */
@Parameter (displayName = "United Market Share Good", usageName = "UnitedMktShrGood")
public static double getUnitedMktShrGood() {
    return UnitedMktShrGood
}
public static void setUnitedMktShrGood(double newValue) {
    UnitedMktShrGood = newValue
}
public static double UnitedMktShrGood = 0

/**
 *
 * This is an agent property.
 * @field UnitedMktShrBad
 *
 */
@Parameter (displayName = "United Market Share Bad", usageName = "UnitedMktShrBad")
public static double getUnitedMktShrBad() {
    return UnitedMktShrBad
}
public static void setUnitedMktShrBad(double newValue) {
    UnitedMktShrBad = newValue
}
public static double UnitedMktShrBad = 0

/**
 *
 * This is an agent property.
 * @field BadTrip
 *
 */
@Parameter (displayName = "Bad Trip %", usageName = "BadTrip")
public static def getBadTrip() {
    return BadTrip
}
public static void setBadTrip(def newValue) {
    BadTrip = newValue
}
public static def BadTrip = 0

/**
 *
 * This value is used to automatically generate agent identifiers.
 * @field serialVersionUID
 *
 */
private static final long serialVersionUID = 1L

/**
 *
 * This value is used to automatically generate agent identifiers.
 * @field agentIDCounter
```

```
     *
     */
    protected static long agentIDCounter = 1

    /**
     *
     * This value is the agent's identifier.
     * @field agentID
     *
     */
    protected String agentID = "MKT " + (agentIDCounter++)

    /**
     *
     * This method provides a human-readable name for the agent.
     * @method toString
     *
     */
    @ProbeID()
    public String toString() {

        // Define the return value variable.
        def returnValue

        // Note the simulation time.
        def time = GetTickCountInTimeUnits()

        // Set the default agent identifier.
        returnValue = this.agentID
        // Return the results.
        return returnValue

    }
}
```

APPENDIX H: PROSPECTIVE OPERATING INCOME STATEMENT AND CASH FLOW

Financial Statement Item (in $ thousands)	**2007 (Projected)**	**2006 (Actual)**	**2005 (Actual)**	**2004 (Actual)**	**2003 (Actual)**
Passengers	10,384	8,676	7,525	5,684	3,926
Revenues					
Passenger	1,158,697	971,507	816,091	626,581	460,188
Cargo	6,529	5,677	4,958	8,077	5,557
Other	31,913	24,338	16,536	9,021	4,191
Total Revenue	1,197,139	1,001,522	837,585	643,679	469,936
Expenses					
Fuel costs	352,905	281,906	185,821	108,862	86,063
Promotion & Sales	117,783	89,751	80,407	65,322	53,031
Other	738,387	637,762	597,804	442,012	361,632
Total Expenses	1,209,075	1,009,419	864,032	616,196	500,726
Operating Income	(11,936)	(7,897)	(26,447)	27,483	(30,790)
Operating Cash Flow	24,486	79,642	19,240	128,017	873

APPENDIX I: COMPARISON OF PROSPECTIVE FINANCIAL INFORMATION TO 2007 ACTUAL DATA

Financial Statement Item (in $ thousands)	2007 (Projected)	2007 (Actual)	Difference
Passengers	10,384	10,039	-3.33%
Revenues			
Passenger	1,158,697	1,131,466	
Cargo	6,529	6,880	
Other	31,913	32,603	
Total Revenue	1,197,139	1,170,949	-2.19%
Expenses			
Fuel costs	352,905	343,082	
Promotion & Sales	117,783	115,536	
Other	738,387	723,033	
Total Expenses	1,209,075	1,181,651	-2.27%
Operating Income	(11,936)	(9,834)	-21.37%
Operating Cash Flow	24,486	23,227	-5.14%

LIST OF REFERENCES

AIS. *Association for Information Systems: Purpose*. Association for Information Systems 2008 [cited. Available from http://home.aisnet.org/displaycommon.cfm?an=3.

Alles, M., G. Brennan, A. Kogan, and M. A. Vasarhelyi. 2006. Continuous monitoring of business process controls: A pilot implementation of a continuous auditing system at Siemens. *International Journal of Accounting Information Systems* 7 (2):137-161.

Alles, M. G., A. Kogan, and M. A. Vasarhelyi. 2002. Feasibility and Economics of Continuous Assurance. *Auditing*.

Altman, E. I. 1968. Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy. *Journal of Finance* 23 (4):589-609.

American Institute of Public Accountants. 1998. *Analytical Procedures, Statement on Auditing Standards No. 56*: New York, NY: AICPA.

———. *AICPA Code of Professional Conduct: Article II - The Public Interest*. AICPA 2008 [cited. Available from http://www.aicpa.org/about/code/et_section_53__article_ii_the_public_interest.html.

Anderson, P. 1999. Complexity theory and organization science. *Organization Science* 10 (3):216-232.

Andreoni, J., and J. H. Miller. 1995. Auctions with Artificial Adaptive Agents. *Games and Economic Behavior* 10 (1):39-64.

Answers.com. *Barnes and Noble*. Answers Corp. 2007 [cited. Available from http://www.answers.com/topic/barnes-noble-inc.

Asare, S. K., and A. M. Wright. 2004. The Effectiveness of Alternative Risk Assessment and Program Planning Tools in a Fraud Setting. *Contemporary Accounting Research* 21 (2):325-352.

Astley, W. G., and A. H. V. de Ven. 1983. Central Perspectives and Debates in Organization Theory. *Administrative Science Quarterly* 28 (2):245-273.

Atlanta Journal-Constitution. *Credit card processors can clip airlines' wings*. DallasNews.com 2008 [cited. Available from http://www.dallasnews.com/sharedcontent/dws/bus/industries/airlines/stories/DN-airlines_14bus.State.Edition1.96500a.html.

Atwater, J. B., and P. H. Pittman. 2006. Facilitating Systemic Thinking in Business Classes. *Decision Sciences The Journal of Innovative Education* 4 (2):273-292.

Avgerou, C., and K. McGrath. 2007. Power, rationality, and the art of living through socio-technical change. *MIS Quarterly* 31 (2):293-315.

Avison, D. E., A. T. Wood-Harper, R. T. Vidgen, and J. R. G. Wood. 1998. A further exploration into information systems development: the evolution of Multiview 2. *Information Technology & People* 11 (2):124-139.

Baker, H. K., G. E. Powell, and E. T. Veit. 2002. Revisiting the dividend puzzle: Do all of the pieces now fit? *Review of Financial Economics* 11 (4):241-261.

Ballas, A., and V. Theoharakis. 2003. Exploring Diversity in Accounting through Faculty Journal Perceptions. *Contemporary Accounting Research* 20 (4):619-644.

Bechtold, B. L. 1997. Chaos theory as a model for strategy development. *Empowerment in Organizations* 5 (4):193-201.

Benbasat, I., and R. W. Zmud. 1999. Empirical Research in Information Systems: The Practice of Relevance. *MIS Quarterly* 23 (1):3-16.

Benbya, H., and B. McKelvey. 2006. Toward a complexity theory of information systems development. *Information Technology and People* 19 (1):12-34.

Bettis, R. A., and C. K. Prahalad. 1995. The Dominant Logic: Retrospective and Extension. *Strategic Management Journal* 16 (1):5-14.

Bitran, G. R., and H. H. Yanasse. 1982. Computational Complexity of the Capacitated Lot Size Problem. *Management Science* 28 (10):1174-1186.

Bonabeau, E. 2002. Predicting the Unpredictable. *Harvard Business Review* 80 (3):109-116.

Brannan, B. *Thoughts On The New Golden Age Of Apple, The Mac*. PanGeo Media 2007 [cited. Available from http://mac360.com/index.php/mac360/comments/thoughts_on_the_new_golden_age_of_apple_the_mac/.

Brenner, T., and C. Werker. 2007. A Taxonomy of Inference in Simulation Models. *Computational Economics* 30 (3):227-244.

Briggs, J., and F. D. Peat. 1999. *Seven Life Lessons of Chaos: Timeless Wisdom from the Science of Change*: New York, NY: HarperCollins.

Bureau of Transportation Statistics. 2008. T-100 Domestic Market (All Carriers).

Burnes, B. 2005. Complexity theories and organizational change. *International Journal of Management Reviews* 7 (2):73-90.

Cameron, K. S., and D. A. Whetten. 1981. Perceptions of organizational effectiveness over organizational life cycles. *Administrative Science Quarterly* 26 (4):525-544.

Carugati, A. 2008. Information systems development activities and inquiring systems: an integrating framework. *European Journal of Information Systems* 17:143-155.

Casti, J. L. 1997. *Would-be Worlds: How Simulation is Changing the Frontiers of Science*: New York, NY: Wiley.

Cederman, L. E. 1997. *Emergent Actors in World Politics: How States and Nations Develop and Dissolve*: Princeton Univ Pr.

Chapman, G. P. 1985. The Epistemology of Complexity and Some Reflections on the Symposium: The Science and Praxis of Complexity, Montpellier, The United Nations University.

Choi, T. Y., K. J. Dooley, and M. Rungtusanatham. 2001. Supply networks and complex adaptive systems: control versus emergence. *Journal of Operations Management* 19 (3):351-366.

Churchman, C. W. 1971. *The design of inquiring systems: basic concepts of systems and organization*. New York, NY: Basic Books.

CICA/AICPA. 1999. *Continuous Auditing*. Canadian Institute of Chartered Accountants and American Institute of Certified Public Accountants. Research Report. Toronto, Canada: CICA.

Cilliers, P. 1998. *Complexity and Postmodernism: Understanding Complex Systems*: New York, NY: Routledge.

City and County of Denver, Colorado Municpal Airport System. *Annual Financial Report* 2006 [cited. Available from http://www.flydenver.com/diabiz/stats/financials/reports/2006_finrpt.pdf.

Clarke, T. 2005. Accounting for Enron: shareholder value and stakeholder interests. *Corporate Governance* 13 (5):598-612.

Coderre, D. 2006. A continuous view of accounts. *Internal Auditor* 63 (2):25-31.

Conner, D. R. 1998. *Leading at the Edge of Chaos: How to Create the Nimble Organization*: New York, NY: John Wiley and Sons.

Constanza, R., L. Wainger, C. Folke, and K. G. Mäler. 1993. Modeling Complex Ecological Economic Systems. *BioScience* 43 (8):545-555.

Corning, P. A. 1997. The Emergence of "Emergence": Now What? *EMERGENCE* 4 (3):54-71.

Courtney, J. F. 2001. Decision making and knowledge management in inquiring organizations: toward a new decision-making paradigm for DSS. *Decision Support Systems* 31 (1):17-38.

Courtney, J. F., D. T. Croasdell, and D. B. Paradice. 1998. Inquiring Organizations. *Australian Journal of Information Systems* 6 (1):3-15.

Cox, C. *Speech by SEC Chairman:*
'The SEC Agenda for 2008'. U.S. Securities and Exchange Commission 2008 [cited. Available from http://www.sec.gov/news/speech/2008/spch020808cc.htm.

Debreceny, R., G. L. Gray, W. L. Tham, K. Y. Goh, and P. L. Tang. 2003. The Development of Embedded Audit Modules to Support Continuous Monitoring in the Electronic Commerce Environment. *International Journal of Auditing* 7 (2):169-185.

Debreceny, R., G. Gray, J. Ng, K. Lee, and W. Yau. 2005. Embedded Audit Modules in Enterprise Resource Planning Systems: Implementation and Functionality. *Journal of Information Systems* 19 (2):7-27.

Denning, P. 2007. Computing is a Natural Science. *Communications of the ACM* 50 (7):13.

Dent, C. B. 1999. Complexity, the new World View. *EMERGENCE* 1 (3):5-20.

Dooley, K. J. 1997. A Complex Adaptive Systems Model of Organization Change. *Nonlinear Dynamics, Psychology, and Life Sciences* 1 (1):69-97.

Dooley, K. J., and A. H. Van de Ven. 1999. Explaining complex organizational dynamics. *Organization Science* 10 (3):358-372.

Dugan, M. T., and C. V. Zavgren. 1988. Bankruptcy Prediction Research: A Valuable Instructional Tool. *Issues in Accounting Education* 1:48-65.

Dyslin, A. *A hops crisis looms: Cost of beer could rise*. The Free Press 2007 [cited. Available from http://www.mankato-freepress.com/local/local_story_319225135.html.

Edmonds, B. 1999. What is Complexity?-The philosophy of complexity per se with application to some examples in evolution. In *The Evolution of Complexity*, edited by F. Heylighen and D. Aerts: Dordrecht: Kluwer.

Epstein, J. M., and R. Axtell. 1996. *Growing Artificial Societies: Social Science from the Bottom Up*: Washington, DC: Brookings Institution Press.

Ethiraj, S. K., and D. Levinthal. 2004. Modularity and Innovation in Complex Systems. *Management Science* 50 (2):159-173.

Feldman, M. P. 2002. The Internet revolution and the geography of innovation. *International Social Science Journal* 54 (171):47-56.

Finkelstein, S. 2007. First Mover Advantage for Internet Startups: Myth or Reality? In *2002 Handbook of Business Strategy* New York, NY: ED Media Group.

Fitzgerald, L. A., and F. M. van Eijnatten. 2002. Chaos speak: a glossary of chaordic terms and phrases. *Journal of Organizational Change Management* 15 (4):412-423.

Florian, M., J. K. Lenstra, and Ahgr Kan. 1980. Deterministic Production Planning: Algorithms and Complexity. *Management Science* 26 (7):669-679.

Forrest, E., and R. Mizerski. 1996. *Interactive Marketing: The Future Present*: American Marketing Association.

Forrester, J. W. 1994. System Dynamics, Systems Thinking, and Soft OR. *System Dynamics Review* 10 (2):245-256.

Frankfurter, G. M., and R. W. Lane. 1984. The rationality of dividends. *International Review of Financial Analysis* 1:115-130.

Frederick, W. C. 1998. Creatures, Corporations, Communities, Chaos, Complexity: A Naturological View of the Corporate Social Role. *Business & Society* 37 (4):358.

Frontier Airlines Holdings, Inc. 2006. Annual Report.

———. 2007. Annual Report.

Fulkerson, B. 1997. A response to dynamic change in the market place. *Decision Support Systems* 21 (3):199-214.

Fuller, T., and P. Moran. 2001. Small enterprises as complex adaptive systems: a methodological question? *Entrepreneurship & Regional Development* 13 (1):47-63.

Gault, S. B., and A. T. Jaccaci. 1996. Complexity meets periodicity. *The Learning Organization* 3 (2):33-9.

Gibson, R. 1997. *Rethinking the Future: Rethinking Business, Principles, Competition, Control & Complexity, Leadership, Markets and the World*: London, England: Nicholas Brealey.

Glass, N. 1996. Chaos, non-linear systems and day-to-day management. *European Management Journal* 14 (1):98-106.

Glover, S.M., J. Jiambalvo, and J. Kennedy. 2000. Analytical procedures and audit-planning decisions. *Auditing: A Journal of Practice & Theory* 19 (2):27-46.

Goldstein, J. 1994. *The Unshackled Organization: Facing the Challenge of Unpredictability Through Spontaneous Reorganization*: Portland, OR: Productivity Press.

Gouws, D. G., and P. Lucouw. 2000. A dynamic balance model for analysts and managers: School of Accounting Sciences, UP.

Grant, R. M., and C. Baden-Fuller. 2004. A Knowledge Accessing Theory of Strategic Alliances. *Journal of Management Studies* 41 (1):61-84.

Gray, G. L., and R. Debreceney. 2006. Continuous Assurance Using Text Mining. In *12th World Continuous Auditing Symposium*. Newark, N.J.

Grice, J. S., and M. T. Dugan. 2001. The Limitations of Bankruptcy Prediction Models: Some Cautions for the Researcher. *Review of Quantitative Finance and Accounting* 17 (2):151-166.

Grice, J. S., and R. W. Ingram. 2001. Tests of the generalizability of Altman's bankruptcy prediction model. *Journal of Business Research* 54 (1):53-61.

Grisé, M. L., and R. B. Gallupe. 1999. Information overload: addressing the productivity paradox in face-to-face electronic meetings. *Journal of Management Information Systems* 16 (3):157-185.

Groomer, S. M., and U. S. Murthy. 1989. Continuous Auditing of Database Applications: An Embedded Audit Module Approach. *Journal of Information Systems* 3 (2):53-69.

Haardoefer, R. 2005. Evolution of the Siemens Experience in its Effort to Test IT Controls on a Continuous Basis. In *10th World Continuous Auditing Symposium*. Newark, N.J.

Hackenbrack, K. 1993. The effect of experience with different sized clients on auditor evaluations of fraudulent financial reporting indicators. *Auditing: A Journal of Practice and Theory* 12 (1):99-110.

Haigh, C. 2002. Using chaos theory: the implications for nursing. *Journal of Advanced Nursing* 37 (5):462-469.

Hall, D. J., and R. A. Davis. 2007. Engaging multiple perspectives: A value-based decision-making model. *Decision Support Systems* 43 (4):1588-1604.

Hall, D. J., and D. Paradice. 2005. Philosophical foundations for a learning-oriented knowledge management system for decision support. *Decision Support Systems* 39 (3):445-461.

Hall, D. J., D. B. Paradice, and J. F. Courtney. 2003. Building a theoretical foundation for a learning-oriented knowledge management system. *Journal of Information Technology Theory and Application* 5 (2):63-84.

Hanseth, O., E. Jacucci, M. Grisot, and M. Aanestad. 2006. Reflexive standardization. Side-effects and complexity in standard-making. *MIS Quarterly* 30:563–581.

Harris, J., and M. Uncles. 2007. Modeling the Repatronage Behavior of Business Airline Travelers. *Journal of Service Research* 9 (4):297.

Heimlich, J. *2007 Outlook: "Reaching for the Skies?"* Air Transport Association of America, Inc. 2007 [cited 3/12/2009. Available from http://www.airlines.org/economics/review_and_outlook/ATA2007EconOutlookOpEd.htm.

Hevner, A. R., S. T. March, J. Park, and S. Ram. 2004. Design Science in Information Systems Research. *MIS Quarterly* 28 (1):75-105.

Hoffman, W. M., and M. Rowe. 2007. The Ethics Officer as Agent of the Board: Leveraging Ethical Governance Capability in the Post-Enron Corporation. *Business and Society Review* 112 (4):553-572.

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

Horgan, J. 1995. From complexity to perplexity. *Scientific American* 272 (6):104–109.

iPodGames. *History of Apple*. iDev Entertainment 2008 [cited. Available from http://www.ipodgames.com/history/apple.php5.

Johnston, R. B., and S. Gregor. 2000. A theory of industry-level activity for understanding the adoption of interorganizational systems. *European Journal of Information Systems* 9 (4):243-252.

Kauffman, S. A. 1991. Antichaos and adaptation. *Scientific American* 265 (2):64-70.

———. 1993. *The Origins of Order: Self-organization and Selection in Evolution (1993)*: New York, NY: Oxford University Press.

———. 1995. *At Home in the Universe: The Search for Laws of Self-organization and Complexity*: New York, NY: Oxford University Press.

———. 1997. At Home in the Universe. *Mathematical Social Sciences* 33 (1):94-95.

Keen, P. G. W., and H. G. Sol. 2007. Rehearsing the Future: Building Decision Agility through Decision Enhancement Services: press.

Kelly, S., and M. A. Allison. 1999. *The complexity advantage: how the science of complexity can help your business achieve peak performance*: New York, NY: McGraw-Hill.

Khouja, M., M. Hadzikadic, H. K. Rajagopalan, and L. S. Tsay. 2007. Application of complex adaptive systems to pricing of reproducible information goods. *Decision Support Systems*.

Kim, R. M., and S. M. Kaplan. 2006. Interpreting socio-technical co-evolution: Applying complex adaptive systems to IS engagement. *Information Technology and People* 19 (1):35.

Kishore, R., H. Zhang, and R. Ramesh. 2006. Enterprise integration using the agent paradigm: foundations of multi-agent-based integrative business information systems. *Decision Support Systems* 42 (1):48-78.

Kogut, B., and U. Zander. 1992. Knowledge of the firm, combinative capabilities, and the replication of technology. *Organization Science* 3 (3):383-397.

Kuhn, J. R., and S. G. Sutton. 2006. Learning from WorldCom: Implications for Fraud Detection Through Continuous Assurance. *Journal of Emerging Technologies in Accounting* 3 (1):61-80.

———. 2008. Commentary on "Embedded Audit Modules in Enterprise Resource Planning Systems: Implementation and Functionality": Working paper, University of Central Florida.

Laurent, G., and C. Koch. 1999. Complexity and the nervous system. *Science* 284:96–98.

Law, A. M. 2005. How to build valid and credible simulation models. *Proceedings of the 37th conference on Winter simulation*:58-66.

Lewin, R. 1992. *Complexity: Life at the edge of chaos. 307*. Chicago, IL: The University of Chicago Press.

———. 1993. Order for free. *New Scientist*:10-1.

Linden, L. P., Kuhn, J. R., Parrish, J., Richardson, S. M., Adams L. A., Elgarah W., Courtney J. F. 2007. Churchman's Inquiring Systems: Kernel Theories for Knowledge Management. *Communications of the Association for Information Systems* 20:836-871.

Lorenz, E. N. 1993. *The Essence of Chaos*: Seattle, WA: University of Washington Press.

Luoma, M. 2006. A Play of Four Arenas: How Complexity Can Serve Management Development. *Management Learning* 37 (1):101.

Mainzer, K. 1997. *Thinking in Complexity: The Complex Dynamics of Matter, Mind, and Mankind*: New York, NY: Springer.

Manson, S. M. 2001. Simplifying complexity: a review of complexity theory. *Geoforum* 32 (3):405-414.

———. 2003. Epistemological possibilities and imperatives of complexity research: a reply to Reitsma. *Geoforum* 34 (1):17-20.

March, J. G., and C. Heath. 1994. *A Primer on Decision Making: How Decisions Happen*: New York, NY: Free Press.

Markus, M. L., A. Majchrzak, and L. Gasser. 2002. A design theory for systems that support emergent knowledge processes. *MIS Quarterly* 26 (3):179-212.

Mason, R. B. 2007. The external environment's effect on management and strategy: A complexity theory approach. *Management Decision* 45 (1):10-28.

Mason, R. O., and I. I. Mitroff. 1973. A Program for Research on Management Information Systems. *Management Science* 19 (5):475-488.

Mathews, K. M., M. C. White, and R. G. Long. 1999. Why Study the Complexity Sciences in the Social Sciences? *Human Relations* 52 (4):439-462.

McKee, T. E. 2003. Rough sets bankruptcy prediction models versus auditor signalling rates. *Journal of Forecasting* 22 (8):569-586.

McKelvey, B. 2004. Toward a 0 thLaw of Thermodynamics: Order-Creation Complexity Dynamics from Physics and Biology to Bioeconomics. *Journal of Bioeconomics* 6 (1):65-96.

Merali, Y. 2002. The role of boundaries in knowledge processes. *European Journal of Information Systems* 11 (1):47-60.

Merriam-Webster. *Merriam-Webster's Online Dictionary*. Merriam-Webster, Inc. 2008 [cited. Available from http://www.merriam-webster.com/dictionary/complex.

Midgley, D., R. Marks, and D. Kunchamwar. 2007. Building and assurance of agent-based models: An example and challenge to the field. *Journal of Business Research* 60 (8):884-893.

Mikulecky, D. C. *Definition of Complexity (and linked pages)* 1999 [cited. Available from http://views.vcu.edu/~mikuleck/ONCOMPLEXITY.html.

Mitchell, R. K., B. R. Agle, and D. J. Wood. 1997. Toward a theory of stakeholder identification and salience: defining the principle of who and what really counts. *Academy of Management Review* 22 (4):853-886.

Mitroff, II, and H. A. Linstone. 1993. *The Unbounded Mind: Breaking the Chains of Traditional Business Thinking*. New York, NY: Oxford University Press.

Mouck, T. 1998. Capital markets research and real world complexity: the emerging challenge of chaos theory. *Accounting, Organizations and Society* 23 (2):189-215.

———. 2000. Beyond Panglossian theory: strategic capital investing in a complex adaptive world. *Accounting, Organizations and Society* 25 (3):261-283.

Nilson, T. H. 1995. *Chaos marketing: how to win in a turbulent world*: Maidenhead, Berks: McGraw-Hill.

North, M. J., and C. M. Macal. 2007. Managing Business Complexity: Discovering Strategic Solutions With Agent-Based Modeling And Simulation: Oxford, UK: Oxford University Press.

O'Keefe, R. M., O. Balci, and E. P. Smith. 1987. Validating expert system performance. *IEEE Expert* 2 (4):81-90.

Ohlson, J. A. 1980. Financial ratios and the probabilistic prediction of bankruptcy. *Journal of Accounting Research* 18 (1):109-131.

Painter-Morland, M. 2006. Redefining Accountability As Relational Responsiveness. *Journal of Business Ethics* 66 (1):89-98.

Parellada, R. J. F. 2002. Modeling of Social Organizations. *Emergence* 4 (1/2):131-46.

Parrish, Jr., J. L. 2008. Sensemaking and Inquiring Systems: Towards a Weickian Inquiring System, PhD diss., University of Central Florida.

Pascale, R. T. 1999. Surfing the edge of chaos. *Sloan Management Review* 40 (3):83-94.

Pfeffer, J., and R. I. Sutton. 2006. Three Myths of Management. *HBS Working Knowledge* March 27.

Phillips, F., and N. Kim. 1996. Implications of Chaos Research for New Product Forecasting. *Technological Forecasting and Social Change* 53 (3):239-261.

Pincus, K. V. 1989. The efficacy of a red flags questionnaire for assessing the possibility of fraud. *Accounting, Organizations and Society* 14 (1/2):153–163.

Reitsma, F. 2003. A response to simplifying complexity. *Geoforum* 34 (1):13-16.

Reynolds, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics* 21 (4):25-34.

Rezaee, Z., A. Sharbatoghlie, R. Elam, and P. L. McMickle. 2002. Continuous Auditing: Building Automated Auditing Capability. *Auditing*.

Ricart, G. 2007. Continuous Auditing and Reporting. The Role of Public Cryptography. In *14th World Continuous Auditing and Reporting Symposium*. Newark, N.J.

Richardson, K.A., and P. Cilliers. 2001. What is complexity science? A view from different directions. *Emergence* 3 (1):5-23.

Richardson, S. M., and J. F. Courtney. 2004. A Churchmanian theory of knowledge management system design. Paper read at Proceedings of the 37th Hawaii International Conference on System Sciences, January 5-8, at Big Island, HI: U.S.

Richardson, S. M., J. F. Courtney, and J. D. Haynes. 2006. Theoretical principles for knowledge management system design: Application to pediatric bipolar disorder. *Decision Support Systems* 42 (3):1321-1337.

Richardson, S. M., J. F. Courtney, and D. B. Paradice. 2001. An assessment of the Singerian approach to organizational learning: Cases from academia and the utility industry. *Information Systems Frontiers* 3 (1):49-62.

Rittel, H. W. J., and M. M. Webber. 1973. Dilemmas in a general theory of planning. *Policy Sciences* 4 (2):155-169.

Rodgers, W., L. Hedelin, T. Housel, and J. R. Kuhn. 2008. Exploratory and Exploitative Knowledge Learning by Investment Analysts. Working paper, University of California - Riverside.

Sánchez, P. J. 2006. As simple as possible, but no simpler: a gentle introduction to simulation modeling. *Proceedings of the 37th conference on Winter simulation*:2-10.

Schein, E. 1992. *Organizational Culture and Leadership*. San Francisco, CA: Jossey-Bass.

Siebers, P. O., and U. Aickelin. 2007. Introduction to Multi-Agent Simulation. In *Encyclopedia of Decision Making and Decision Support Technologies*: IDEAS Group (In press).

Simon, H. A. 1996. *The Sciences of the Artificial*: Cambridge, MA: MIT Press.

Srbljinovic, A., and O. Skunca. 2003. Agent Based Modelling and Simulation of Social Processes. *Interdisciplinary Description of Complex Systems* 1 (1-2):1-8.

Stacey, R. D. 1995. The Science of Complexity: An Alternative Perspective for Strategic Change Processes. *Strategic Management Journal* 16 (6):477-495.

Stumpf, S. A. 1995. Applying new science theories in leadership development activities. *Development* 14 (5):39-49.

Styhre, A. 2002. Non-linear change in organizations: organization change management informed by complexity theory. *Leadership & Organization Development Journal* 23 (6):343-51.

Suarez, F. F., and G. Lanzolla. 2005. The Half-Truth of First-Mover Advantage. *Harvard Business Review* 83 (4):121-129.

Sutherland, J., and W. J. van den Heuvel. 2002. Enterprise application integration and complex adaptive systems. *COMMUNICATIONS OF THE ACM* 45 (10):59-64.

Suzuki, Y. 2000. The relationship between on-time performance and airline market share: a new approach. *Transportation Research Part E* 36 (2):139-154.

Tan, J., H. J. Wen, and N. Awad. 2005. Health care and services delivery systems as complex adaptive systems. *COMMUNICATIONS OF THE ACM* 48 (5):36-44.

Thietart, R. A., and B. Forgues. 1995. Chaos Theory and Organization. *Organization Science* 6 (1):19-31.

Thrane, S. 2007. The complexity of management accounting change: Bifurcation and oscillation in schizophrenic inter-organisational systems. *Management Accounting Research* 18 (2):248-272.

Thrift, N. 1999. The Place of Complexity. *Theory, Culture & Society* 16 (3):31.

Tilanus, C. B. 1981. Management Science in the 1980s. *Management Science* 27 (9):1088-1090.

Troitzsch, K. G. *Approaching Agent-Based Simulation* 2000 [cited. Available from http://www.uni-koblenz.de/%7Emoeh/publik/ABM.pdf.

Uhl-Bien, M., R. Marion, and B. McKelvey. 2007. Complexity Leadership Theory: Shifting leadership from the industrial age to the knowledge era. *The Leadership Quarterly* 18 (4):298-318.

United Air Lines, Inc. 2006. Annual Report.

Van Valen, L. 1973. A new evolutionary law. *Evolutionary Theory* 1 (1):1-30.

Vardy, A. 1997. Algorithmic complexity in coding theory and the minimum distance problem. Paper read at Twenty-ninth Annual ACM Symposium on Theory of Computing.

Vasarhelyi, M. A. 2005. Would continuous audit have stopped the Enron mess? Working paper, Rutgers University.

Vasarhelyi, M. A., M. G. Alles, and A. Kogan. 2004. Principles of Analytic Monitoring for Continuous Assurance. *Journal of Emerging Technologies in Accounting* 1 (1):1-21.

Vasarhelyi, M. A., and F. B. Halper. 1991. The continuous audit of online systems. *Auditing: A Journal of Practice and Theory* 10 (1):110-125.

Wahlstrom, B. 1994. Models, modelling and modellers; an application to risk analysis. *European Journal of Operations Research* 75:477-487.

Waldrop, M. M. 1993. *Complexity: The Emerging Science at the Edge of Order and Chaos*: New York, NY: Simon & Schuster.

Waller Jr., W. T. 1989. The concept of habit in economic analysis. *Journal of Economic Issues* 22:113-126.

Walls, J. G., G. R. Widmeyer, and O. A. El Sawy. 1992. Building an Information System Design Theory for Vigilant EIS. *Information Systems Research* 3 (1):36-59.

Weick, K. E. 1979. *The social psychology of organizing*: Reading, MA: Addison-Wesley.

Wheatley, M. 1996. The unplanned organization: Learning from nature's emergent creativity. *Noetic Sciences Review*:20-21.

Wilkinson, I., and L. Young. 1998. On Competing: Firms, Relations and Networks. Paper read at Research Conference Proceedings: Relationship Marketing: Theory, Methods and Applications, at Goizueta Business School, Emory University.

Zhang, G., M. Y. Hu, B. E. Patuwo, and D. C. Indro. 1999. Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis, Europ. *J. Op. Research* 116:16.

Zimmerman, B., C. Lindberg, and P. Plsek. 1998. *Edgeware: Insights from Complexity Science for Health Care Leaders*: Irving, TX: VHA.

Zmijewski, M. E. 1984. Methodological issues related to the estimation of financial distress prediction models. *Journal of Accounting Research* 22 (1):59-82.

Zurek, W. H. 1989. Thermodynamic cost of computation, algorithmic complexity and the information metric. *Nature* 341:119-124.