2008

Inspired Design: Using Interdisciplinarity And Biomimicry For Software Innovation

Steven A. Korecki
Grand Valley State University

Inspired Design:
Using Interdisciplinarity and Biomimicry for Software Innovation

By
Steven A. Korecki
April, 2008

Inspired Design: Using Interdisciplinarity and Biomimicry for Software Innovation

By
Steven A. Korecki

A Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in

Computer Information Systems

at

Grand Valley State University

April, 2008

_____

Paul Jorgensen, Professor Date

_____

Greg Wolffe, Associate Professor Date

_____

Robert Adams, Associate Professor Date

GRAND VALLEY STATE UNIVERSITY

ABSTRACT

"INSPIRED DESIGN: USING
INTERDISCIPLINARITY AND BIOMIMICRY
FOR SOFTWARE INNOVATION"

By Steven A. Korecki

This thesis presents research and proposes a framework for increasing the breadth and depth of interdisciplinary knowledge in the field of computer science. The intent is to address an increasing problem of complexity in software and computing systems. The approach is to equip software developers and computer scientists with a contextual perspective and a set of strategies for injecting innovation and creativity into the solutions they design by leveraging knowledge and models outside the traditional realm of computer science. A review of current and historical forms of interdisciplinarity and biomimicry are presented to build that context. The strategies presented include interdisciplinary education, interdisciplinary collaboration, interdisciplinary tools, biomimetic design, and the creation of new pattern languages based on nature's design solutions. Each of these strategies stems from and leads to an open exchange of knowledge across disciplinary boundaries. When taken together, the knowledge and strategies presented here are intended to inspire and foster a paradigm that recognizes and harnesses the value of human and natural diversity as a source of innovation.

TABLE OF CONTENTS

LIST OF FIGURES

Number Page

LIST OF TABLES

Number Page

ACKNOWLEDGMENTS

Over the past four years, I've spent countless hours researching and writing this thesis. It has taken more time than all my other graduate classes combined and has truly been one of the most difficult things I've ever done. It is with this understanding that I wish to extend my great appreciation to the people who have made its completion possible.

First, I thank my committee. I thank Dr. Paul Jorgensen for his curiosity and open-mindedness which have enabled me to pursue my rather non-traditional thesis. His flexibility and support have enabled me to finally "get'er done". I thank Dr. Greg Wolffe who helped prepare me for a long-term commitment the very first time I met with him. His words have helped me to stay grounded when I could have flown off in several directions. I thank Dr. Robert Adams for taking a leap of faith and hitting the ground running. I thank Dr. Carl Erickson for his early contributions and feedback. I also thank Franco Lodato for giving me a new perspective on design and the exciting history and potential of bionics. I also appreciate the opportunities he gave me to network with other remarkable experts in design, technology, and biomimicry.

Along with Franco, I thank Catherine Bragdon for inviting me to participate in the BioDesign Team at Herman Miller. My involvement with this team has opened many doors and introduced me to people I never thought I'd have the opportunity to meet. First, Janine Benyus, founder of the Biomimicry Institute and the Biomimicry Guild. Her mission and talent for promoting nature as model, measure, and mentor are inspirational. I appreciate the encouragement, perspective, and sources she shared with me during her visits to HMI. Second, I thank Dr. Julian Vincent of the Univesity of Bath in the UK. I

met Dr. Vincent at a Biomimicry Database Workshop in Ontario in 2006. His insights, papers, and references introduced me to TRIZ. Although it seemed daunting at first, I now see it as a rich area of opportunity for many forms of development.

I thank my co-workers who have been excellent sounding boards. I thank Brian Geary for his interest and encouragement. I thank Jim Van Dragt for his good counsel and balanced perspective. I thank Ben Staples for his excellent insights on my subject matter and his willingness to really help me when I needed it most.

I thank Dr. Jonathan Engelsma of GVSU and Motorola for taking an interest in my research. His advice and encouragement helped me refocus my efforts after a time of doubt and frustration. I thank Dr. Zachary Huang of Michigan State University for his time and feedback on my multiagent simulation for modeling the division of labor in Honeybee colonies. My interactions with him really opened my eyes to the power of interdisciplinary collaboration and the deep knowledge that can be accessible through it.

Truly, I am most thankful for the support of my family. The prayers, words, and deeds from my parents W. James & Violet Korecki and my in-laws Michael & Terry Metzger have sustained me and my family on this long journey. I also thank my brothers, sisters, in-laws, and my Aunt Mary who have all been an encouragement. I particularly appreciate the motivational speeches from Jayne and Jim. Most of all I am forever grateful for the love and encouragement of my wife Stacey and our children Samantha, Alec, and Autumn. Stacey has been an inspiration to me and I simply could not have done this work without her. She has shown sacrifice and patience beyond anything I could have asked. I know that I worked compulsively, and yet she has encouraged me and loved me throughout. I am honored to have her as my wife and best friend.

Finally, it is my great joy to thank the Lord God Almighty for inspiring me and opening my eyes to His great creation. I am in awe of His endless creativity which has brought forth the diversity and unity of life and knowledge. Ultimately, it is through Him that all things will be revealed. It is my hope that this thesis in some way points to His great plan – for in Him all things are possible.

"Ask and it will be given to you; seek and you will find; knock and the door will be opened to you." – NIV Matthew 7:7

Chapter 1

1 INTRODUCTION

The field of computer science has experienced an expanding influence on other fields of scientific and academic study. General computing capabilities such as data processing, visualization, and communication as well as highly specialized software applications have lead to enormous breakthroughs in mathematics, natural science, social science, and more. These advancements have been made through unprecedented access to information that was not previously been within human reach. However, advancements in software have also introduced complexity issues that have the potential to affect nearly all sectors of society.

To solve these problems of complexity, computer scientists must consider alternative approaches to developing more robust and sustainable software. These alternative methods include interdisciplinary collaboration and exploratory search for better product and process models. This research asserts that the natural world is an invaluable source of models that can inspire and teach computer scientist new and better ways of designing and developing software.

There are already limited areas of interdisciplinary focus and bio-inspired design within computer science. However, this thesis hopes to encourage many more areas of innovation by providing a deeper understanding of interdisciplinarity and biomimicry. It also hopes to elevate early software related research on interdisciplinary innovation tools such as TRIZ.

1.1 Software Development Processes

Software development is a creative design process which captures human knowledge in a precise executable language. Like other design related fields, there are various approaches to this process. Some are highly predictive in nature, like the waterfall model of development. A waterfall process starts out with a long series of analysis and documentation steps which lead to an abstract design and finally an implementation. Once an implementation has been realized, a series of testing phases take place which mirror the analysis phases. An alternative approach to software development is based on iteration. Agile software development is an example which focuses on rapid development of executable code which is gradually evolved as new features are added and tested. Like other development processes, software development can be enhanced through interdisciplinary involvement and idealized design models.

1.2 A General Problem Solving Framework

One of the fundamentals of software development is a problem solving framework like the one shown in Figure 1. This simple framework illustrates an indirect problem solving process. Through analysis a problem is typically articulated into a requirements document which is an explicit representation of the real world problem. From this representation, a design can be modeled into a representation of a solution. This representation can then be implemented into source code as a software application that is intended to solve the original real world problem. If the implemented solution meets the original customer problem, then the process would be considered successful.

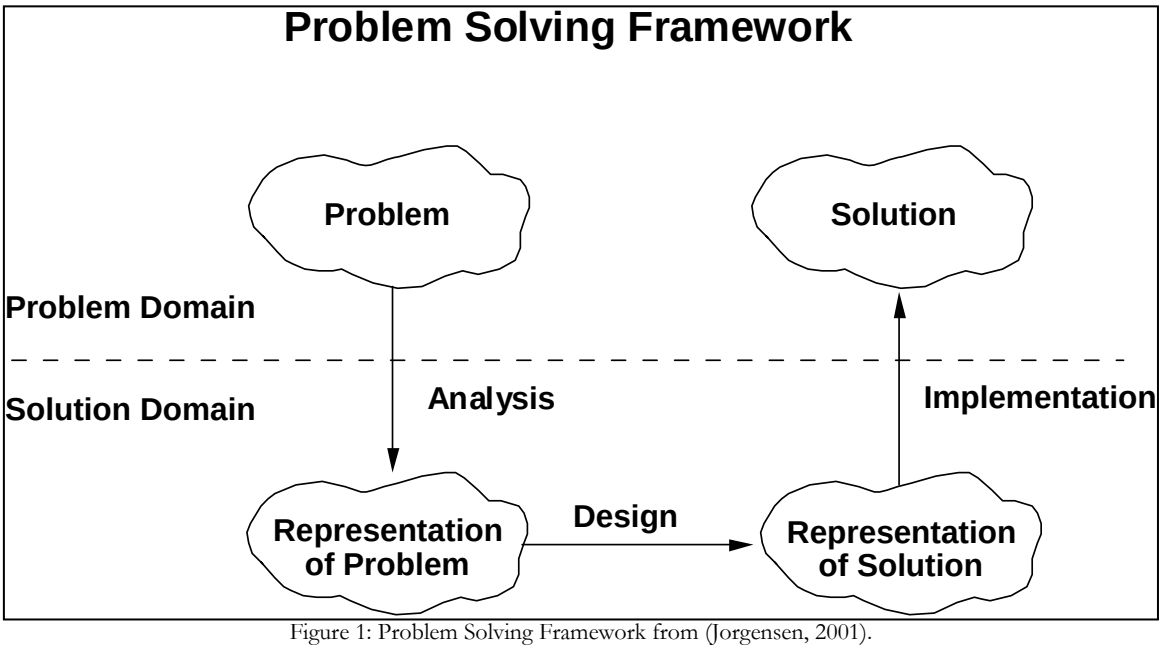

Figure 1: Problem Solving Framework from (Jorgensen, 2001).

As simple as this process appears, it holds great power – for abstraction can be a means of taking a problem or solution from one field and making it apply to another. That is, the abstraction layer becomes a common ground for experts in diverse fields to discuss in a common language the problems and solutions on which they work. Abstraction is a key enabler for interdisciplinary problem solving and innovation.

1.3 The Nature of Knowledge

Disciplinary knowledge is the result of hundreds of years of educational systems which have become increasingly specialized. Although specialization is a powerful tool which leverages the power of individuals in a society, it also introduces fragmentation and discontinuity in individual and collective understanding. It arguably undermines our ability to recognize the unity of knowledge and the natural world. However, there has been a tremendous amount of success in bridging the disciplinary silos which have been created. Crossdisciplinary, multidisciplinary, interdisciplinary, and transdisciplinary activities all

represent different ways of approaching topics more holistically. They can be implemented in educational programs, research activities, and problem solving.

1.4 The Knowledge of Nature

The natural world is the source and subject of nearly all human knowledge. Nature is the most complex system known to man and it contains countless "designs" in the form of the biological species and environmental phenomena that make our world unique. Biomimicry is a maturing science of studying the designs, processes, and phenomena in nature as a source of inspiration for human creations. It acknowledges nature as a model for us to imitate, a measure for us to evaluate our designs, and a mentor from which we can learn ((J. M. Benyus, 2002)).

1.5 Topic and Organization of this thesis

1.5.1 Objective

First, our intent is to develop a new vision for collaborative software development that recognizes and exploits knowledge and models outside the traditional realm of computer science. This vision or paradigm is developed through a survey of interdisciplinary literature and specific historical examples that illustrate the power of interdisciplinary knowledge.

Second, we propose a framework for pursuing this vision. This framework will identify specific strategies for increasing the breadth and depth of interdisciplinary knowledge and collaboration in the field of computer science to foster creativity and innovation in software development. Creativity and innovation are essential for computer scientists and developers to meet the demands of our increasingly technological societies. Without it, we will be paralyzed by the complexity of the systems we create. To better understand how such a framework could be implemented, we will explore

Interdisciplinarity and Biomimicry as models for innovation and problem solving. This framework is not intended to be a panacea. Rather, it will be a tool for software development teams to add to their arsenal.

1.5.2 Benefits

A shared vision and framework for interdisciplinary and collaborative software development may increase the level of creativity and innovation in software development teams. This should potentially elevate the quality of the product which would ultimately benefit its intended end users and the IT professionals that may support it. An interdisciplinary approach may also present opportunities to increase knowledge exchange among collaborators. This type of exchange can enrich all parties and allow for more synergy between disciplines. Participants may gain more opportunities to advance their respective fields by leveraging the knowledge, processes, and conventions learned during collaboration. These collaborations would also help in the development of cross-disciplinary social networks that can be drawn upon in the future.

Computer Science is a field that can benefit from the experience of other fields that may often have a longer history. Other disciplines, such as the classical sciences could benefit from an increased understanding of software technologies and the processes used to develop them.

1.5.3 Approach

First, we will develop a context for the challenges facing computer scientists, software developers, and members of a technological society. Second, we will explore the various forms of interdisciplinary activities which will serve as a model for computer scientists. Third, we will dive into nature inspired design and its methods as a source of innovation. Biomimetics or biomimicry are established approaches to leverage designs that

have been proven successful in nature. Fourth, we will review the current state of interdisciplinary and biomimetic software development. Fifth, we will present a framework for software innovation based on the material presented in the previous chapters. Sixth, we will present a case study of an interdisciplinary effort to develop a biomimetic software algorithm for task allocation based on the latest scientific research on honeybee specialization. Finally, we will conclude with a summary of the contributions.

1.5.4 Measurement

The ideas put forth in this thesis have emerged from studying and connecting diverse subject areas. A measure of success will be realized when the reader sees that seemingly unrelated topics can point to common patterns and simple truths. Common underlying principles can by found throughout the natural world, technology, and society. Through observation and conscious abstraction, software developers can use these principles and patterns as a source of inspiration.

Chapter 2

2 A TRANSCENDENT PROBLEM OF COMPLEXITY

A system is considered complicated if it can be described comprehensively as the sum of its parts, no matter how many there may be. Computing systems are certainly complicated; however, more than that they are also complex. A complex system is one that cannot be fully understood by analyzing the sum of is parts ((Reitsma, 2002)). So it is with computing systems that contain or interact with so many interconnected pieces that the sum becomes unpredictable. Industrialized societies are increasingly dependent on systems of systems whose emerging behavior cannot be fully understood.

2.1 Computing Induced Challenges

Computing systems and the data they contain are increasing in complexity at a tremendous rate. Rampant IT growth, nearly ubiquitous network connectivity, and emerging technologies have impacted the daily lives of nearly everyone in the industrialized world and beyond. The demands and expectations for IT systems are rising as they are integrated with one another and adopted into all facets of society. Ultimately, the complexity of these interconnected systems and the immense volume of data they contain will strain the limits of human capacity to manage and interact with them. This reality was acknowledged by IBM executive Paul Horn in 2001 when he stated:

"More than any other IT problem, this one—if it remains unsolved—will actually prevent us from moving to the next era of computing. The obstacle is complexity… Dealing with it is the single most important challenge facing the IT industry." – Paul Horn, IBM Senior Vice President and Director of Research (Ganek & Corbi, 2003)

These powerful words set the stage for this challenge and hint toward an increasingly technology dependent future. Robust, scalable, and networked software must be developed to manage and interact with the escalating complexity and ubiquity of computer-based technology. To understand the scope of this challenge, it is worth taking a closer look at the nature of issues such as rampant IT growth, nearly ubiquitous network connectivity, and emerging technologies.

2.1.1 Rampant IT Growth and Complexity

Software developers and application providers are being faced with an unprecedented number of choices as they design and enhance their products. New development paradigms have diverged from the traditional waterfall method to meet the rapidly changing demands of customers. Proprietary and open source platforms have forced many companies to scrutinize their short-term and long-term development strategies and tools. Source code is becoming more complex as developers choose to add new layers of abstraction and integration. They must also select from an abundance of platform options. For example, there are now more than 2500 high-level programming languages in use today. This is a surprisingly high number when one considers that Fortran introduced itself as the first high-level programming language in the mid-1950s ((Kinnersley, 2006)).

That averages out to be approximately 50 new languages per year. This diverse pallet of development tools has led to an even more diverse canvas of software applications that are produced by the IT industry to meet the needs of customers.

Commercial and institutional customers are now faced with an IT explosion. To meet their own business needs, they have turned to IT systems to maximize efficiency and effectiveness. They now demand more powerful, more scalable and more robust enterprise systems. The IT industry has responded with more interdependent and distributed architectures. Diverse layers of specialized software are integrated to develop highly complex computing ecosystems to meet the demands of businesses and organizations.

As the number of systems increases, so does the need for interoperability between them. Service-oriented architectures (SOA) leverage both proprietary and open standards to provide middleware integration that extends the life of legacy systems. The net effect is that server software platforms grow and change as new ones are introduced. The result of all this is an increase in the number of systems being managed and a continuous stream of maintenance.

The growth of the Internet has dramatically impacted IT providers, systems, and customers. Security to prevent and respond to hackers and malware are a drain on IT systems and personnel. Viruses and vulnerabilities are responsible for data, resource, and time loss. Personal, political, and institutional systems are under the constant threat of malicious attack. Some threats are indiscriminant viruses, worms, and Trojan horses while others are precisely orchestrated attacks by hackers. The Internet is a virtual battleground between the so called “white-hats” and “black-hats”.

2.1.2 Pervasive Software and Emergent Technology

New technologies are being developed each day that are forming a pervasive demand for software. Research organizations such as Gartner, Inc., The Institute for the Future (IFTF) and others monitor technology trends and make forecasts. They have identified many common threads that indicate an increasing dependence on computer-based technology as it becomes more and more ubiquitous in our daily lives. Gartner's annual "hype cycles" for emerging technologies (Fenn & al, 2006a) and human-computer interactions (Fenn & al, 2006b) describe technology maturity levels and forecasted adoption rates (See Figure 2). Many of the technologies identified hold great potential to blur the lines between the physical and the virtual worlds. Some examples include: Location-Aware technologies, mesh networks, speech recognition, RFID, IPv6, Virtual Reality, and Augmented Reality.

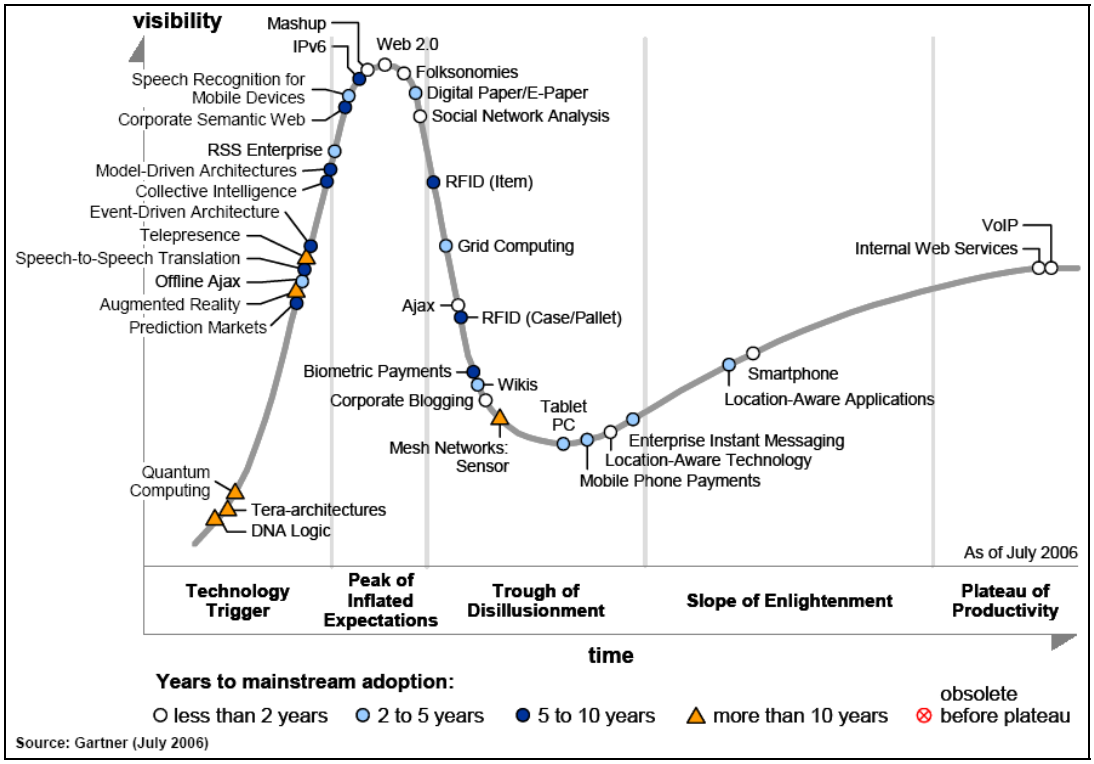

Figure 2: Gartner Hype Cycle for Emerging Technologies 2006 (Fenn & al, 2006a). Figure reprinted with permission from copyright owner.

The Institute for the Future tends to focus on macro trends that impact society. It is often the case that these trends are inseparable from technology. (IFTF, 2006) identifies three major waves of technology starting in the 1990s and continuing for nearly 50 years. These waves, shown in Figure 3 include communicating, sensing, and "sensemaking". The "Communicating" wave consists largely of the growth of the Internet. We now find ourselves in the midst of the "sensing" wave. This second wave describes the profound effect of sensing devices that bring information, awareness, and responsiveness to objects, places, and people. Examples of the technologies behind this wave include RFID, wireless sensor networks, MEMS, and power harvesting technologies. All of which are capable of feeding data streams into IT systems – which leads to the next wave. The "sensemaking" wave represents the ability to make sense out of the vast amounts of information being generated. This may be done with sophisticated mathematical models and simulations, sensory-rich user interfaces, and ubiquitous display technologies.

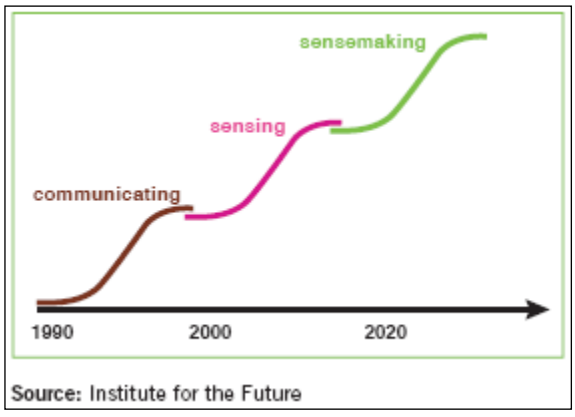

Figure 3: IFTF chart of major technology waves by decade (IFTF, 2006). Figure reprinted with permission from copyright owner.

The various technologies mentioned here are leading to a future where computing systems are integrated into commonplace objects and the background of the physical environment. Sometimes called ubiquitous computing, pervasive computing, or ambient intelligence – the goal is to develop "context-aware" environments that will not only perceive us, but enhance our perception as well. A new layer of digital information will overlay our world granting us a sort of "sixth sense," allowing us to see relevant contextual information as we go about our daily activities. No longer will we need to go to our technology, our technology will come to us. The concept of cyberspace may ultimately fade, as it becomes indistinguishable from the physical space.

The technologies described in this section seem to validate the potential for an increasingly technology enriched future. A future that is highly dependent on highly functional, adaptive, and robust software. Just as Microsoft Windows became a key to the adoption of the personal computer in the early 1990s, software will enable new technology platforms to emerge and be adopted forming an ever growing computing ecosystem.

2.2 Human Limitations and Information Overload

2.2.1 Volume of Information

In an attempt to estimate the total amount of information that is created each year, (Lyman & al, 2003) calculated the amount of new information that was created in 2002 on four types of storage media: print, film, magnetic, and optical. The key findings included:

1. *Print, film, magnetic, and optical storage media produced about 5 Exabytes of new information in 2002. Ninety-two percent of the new information was stored on magnetic media, mostly in hard disks.*

2. *The amount of new information stored on paper, film, magnetic, and optical media has about doubled in the last three years.*

3. *Information flows through electronic channels -- telephone, radio, TV, and the Internet – contained almost 18 Exabytes of new information in 2002, three and a half times more than is recorded in storage media. Ninety eight percent of this total is the information sent and received in telephone calls - including both voice and data on both fixed lines and wireless.*

(Lyman & al, 2003) goes on to explain that an Exabyte is 10^{18} bytes and is equivalent to half a million libraries containing nineteen million books each. Another estimated comparison is that five Exabytes of data is equivalent to "All words ever spoken by human beings." In other words, the amount of information created every year is beyond human comprehension.

Both (Raskino, 2005) and (ACM, 2002) surveyed IT professionals and management personnel about the biggest issues their organizations face. Respondents in both surveys indicated that information overload is a serious or potentially serious problem. (Leong & Basso, 2005) indicates that consumers and professionals alike are overwhelmed by the amount of communication they receive across numerous channels such as email, phone, IM, and fax. These problems may get worse as new technologies pile into our daily lives.

2.2.2 Dealing with Complexity in IT

Corporate and organizational IT departments are finding that managing increasingly complex computing systems is becoming too labor-intensive and prone to error. It is estimated that one-third to one-half of a company's total IT budget is spent preventing or recovering from crashes (Patterson et al., 2002). Additionally, the requirements of highly available systems are straining the people who administer them. The reality of human error increases the potential for costly outages that can impact a business. Research shows that approximately 40% of computer system outages are caused by operator error and the reason is not because operators are not proficient. Rather, the computing systems are too complex and difficult to understand. ((Ganek & Corbi, 2003) and sources)

2.3 Societal Issues

Societies are showing an increasing dependence on software technology. It permeates nearly all aspects of our lives. It is a central means of communication. It is a repository of human knowledge, and it is a tool that enables an information based economy. As its pervasiveness increases, there are also indications that there is an impending shortage of knowledge workers in the United States. There is a concern that a shortage of computer scientists and engineers could leave companies stranded with legacy systems that are extremely difficult and costly to maintain.

A digital divide is also emerging as industrialized nations forge forward into the information age and beyond. Developing countries on the other hand, lack the economic power to compete. They are still struggling to provide basic needs to populations in vast rural areas. "One Laptop per Child" is a non-profit organization attempting to narrow this digital divide. They are attempting to provide low cost computing systems (the $100 laptop) to children in developing countries to educate and equip them with the tools to join

and contribute the information age. It is unknown how diverse cultures will impact software as they embrace computing technology.

2.4 The Challenge

The challenges identified here can be summarized by saying that the problem of complexity in technology and specifically in computing software transcends the field of computer science. Software development is a process riddled with pitfalls that lead to software quality and complexity issues. These issues have physical, social, and philosophical impacts at individual, institutional, and societal levels. The challenge for computer scientists is to respond to this problem. Albert Einstein once said that "No problem can be solved from the same level of consciousness that created it." This statement is true for the problem of complexity in software. One can infer that to solve this problem, one must transcend the level of thinking that created it. One must be more innovative. To do this, we must seek alternative methods of developing, managing, and interacting with computing systems.

Chapter 3

3 A BRIEF REVIEW OF INTERDISCIPLINARY COLLABORATION

The many aspects of computing described in Chapter 2 make a compelling case that there are great challenges facing modern computer scientists. The primary challenge is complexity. The computing systems and the software they create are increasingly ubiquitous and critical with an impact on nearly all sectors of society. Accordingly, all sectors of society should be concerned with addressing that challenge. The challenge is far more than a technical one. There are personal, social, political, economic, scientific, and environmental aspects to the challenge.

Computer scientists are not equipped to address all aspects of technological complexity in isolation. This challenge requires attention and cooperation from all disciplinary fields. An inclusive and participative approach is necessary to innovate the future generations of software that will underpin technological societies. Cross-disciplinary, multidisciplinary, interdisciplinary, and transdisciplinary approaches can be used to organize this effort. They can be used independently and in conjunction to address problems which transcend the narrow scope of disciplinary boundaries. To better understand such approaches, this chapter will describe a brief history of disciplinary knowledge and the ways in which disciplinary boundaries can be crossed. It will also highlight some individuals who have had a great impact on interdisciplinary activities. As confirmed by (Klein, 2003) and (IFTF, 2006), crossing disciplinary boundaries will become imperative over the coming decades.

3.1 Segmentation of Modern Disciplines

In higher education, a discipline refers to a specific branch of scholarly knowledge. The word "branch" is instrumental in this definition in that it alludes to the historical lineage and segmentation of modern disciplines. Although early scientific and mathematical breakthroughs were made by people with broad expertise such as Archimedes, Leonardo da Vinci, Galileo Galilei, and Sir Isaac Newton – the last two hundred years have resulted in ever more fragmented silos of knowledge and discovery. The 19th century classical sciences of astronomy, physics, chemistry, geology, and medicine have evolved into thousands of specialized fields. According to (Klein, 2003), by 1990 approximately 8,000 scientific research topics could be identified and nearly 4000 differentiated disciplines. This trend is a function of the exponential increase in our collective human knowledge and individual human limitations. It is simply not possible to be educated in all identified forms of scholarly knowledge like the historical "Renaissance Men" of the 14th to 16th centuries.

3.2 Benefits of Specialization

This increasing segmentation has come about through necessity and has led to tremendous advances in human understanding. By organizing knowledge into specific disciplines, individuals with common interests have dedicated immeasurable time and effort into understanding the intricacies of the natural world. Like a laser, it has focused the attention of talented and passionate individuals. It has also focused funding from individuals, organizations, and governments. Specialization has allowed new generations of experts to become educated on specific fields, thus leveraging the knowledge of those who have gone before.

"If I have seen further, it is by standing on the shoulders of giants."

– Sir Isaac Newton 1642-1727

In a broad sense, specialization is a powerful strategy that has even been used in nature. There are countless examples in the natural world of how specialization can be used to achieve goals that surpass the capabilities of an individual. It is a principle pattern that has been proven successful (see sections 6.3.2.6 and 7.1).

3.3 Pitfalls of Specialization

Although specialization is born out of necessity and has great power, it is not without pitfalls. First, deeply segmented disciplines have the potential to create tunnel vision, narrow minded views, and a tendency to reinvent the wheel. Without a broader frame of reference, specialized disciplines color the lenses through which individuals perceive the world around them. They evoke a learned bias which causes a one-dimensional view of reality. Put simply, "if all you have is a hammer, then everything looks like a nail." The impact of this single-sided vision is an over simplification of reality ((Klein, 2003) & (Lattanzi, 1998)). Furthermore, it impairs one's ability to see the universality of nature that underlies diverging fields. Second, communication barriers emerge as each field develops and evolves its own technical language. Because of this, it is becoming increasingly difficult for individuals to participate and collaborate across disciplinary boundaries. This divergence of fields and language is sometimes likened to the "Tower of Babel" ((Nicolescu, 2002)). As the Biblical tale reveals, a lack of communication undermines our ability to realize potential. In this case, the potential to

realize the full impact of knowledge and advancements made in isolated fields across disciplinary boundaries.

3.4 Crossing Boundaries

Educational reform and scientific advancement over the last 60 years have precipitated a great deal of crossing of disciplinary boundaries. Recognizing that segmentation of disciplines and its limitations has led to various means of resolution. Although there is no consensus on terminology, there are several ways that the disciplines have been bridged. For the purposes of this research, these means will be described using the terms "Crossdisciplinary", "Multidisciplinary", "Interdisciplinarity", and "Transdisciplinarity". Each of these approaches can enrich a subject.

3.4.1 Crossdisciplinarity

According to (Seipel, 2004), a crossdisciplinary activity "views one discipline from the perspective of another, such as a Physics lab in which principles of physics are used to understand acoustics of music." In (Klein, 2003), Klein defines crossdisciplinary as "an adjective for any kind of crossing of disciplinary boundaries, sometimes formalized as "crossdisciplinarity" meaning axiomatic control from the viewpoint of one discipline, the solution of a problem, or creation of a new field". From these definitions, one can extract the concept of crossing disciplinary boundaries by using one discipline to explain another. For example, researching the ethics of engineering could be considered a crossdisciplinary activity.

3.4.2 Multidisciplinarity

Multidisciplinarity was described by Klein in (Klein, 2003) as "the juxtaposition of disciplines in an additive rather than integrative and interactive fashion, producing an encyclopedic alignment of multiple perspectives." Similarly, Nicolescu in (Nicolescu)

describes it as "studying a research topic not in only one discipline, but in several simultaneously." According to (Seipel, 2004), "multidisciplinary activity draws on the knowledge of several disciplines, each of which provides a different perspective on a problem or issue." The implication of these descriptions is that a multidisciplinary activity is performed by members of distinct disciplines without any attempt to integrate or assimilate the knowledge or activity from each one. The literature often describes this as an additive form of crossing disciplines, as each component discipline's contribution can stand alone. In other words, the whole is equal to the sum of its parts.

3.4.3 Interdisciplinarity

Interdisciplinarity is perhaps the fastest growing means of crossing disciplines. An increasing number of universities, research centers, and corporations are developing interdisciplinary programs as strategic initiatives. What distinguishes these programs from the crossdisciplinary and multidisciplinary programs is their focus on integration. It is a central focus of interdisciplinarity to develop people and knowledge that cross multiple disciplines. In this manner, the knowledge creation becomes synergistic.

Perhaps due to its widespread nature, interdisciplinarity has eluded a clear definition. Klein defines it in (Klein, 2003) as "a label for a variety of interactions that aim to integrate concepts, methods, data, or epistemology of multiple disciplines around a particular question, theme, problem, or idea.". An interdisciplinary analysis is described by Seipel in (Seipel, 2004) as "drawing on the specialized knowledge, concepts, or tools of academic disciplines and integrating these pieces to create new knowledge or deeper understanding". Furthermore, Seipel states that "Interdisciplinary analysis requires integration of knowledge from the disciplines being brought to bear on an issue. Disciplinary knowledge, concepts, tools, and rules of investigation are considered,

contrasted, and combined in such a way that the resulting understanding is greater than simply the sum of its disciplinary parts." In (Nicolescu), Nicolescu states that interdisciplinarity is concerned with the transfer of methods from one discipline to another.

Presumably, the inherent magnitude and diversity of interdisciplinary activities have provoked academics to create frameworks for characterizing it. Nissani articulates the "realms" to which the term interdisciplinarity is most commonly applied. The realms that have been identified help build an understanding of the nature of Interdisciplinarity. These realms can be found in Table 1.

Realm	Description
Interdisciplinary Knowledge	Involves familiarity with distinctive components of two or more disciplines.
Interdisciplinary Research	Combining distinctive components of two or more disciplines while searching or creating new knowledge, operational procedures, or artistic expressions.
Interdisciplinary Education	Merges distinctive components of two or more disciplines in a single program of instruction.
Interdisciplinary Theory	Interdisciplinary knowledge, research, or education as its main objects of study.

Table 1: The "Four Interdisciplinary Realms" according to (Nissani, 1995).

Nissani also proposed a set of criteria for ranking the richness of an Interdisciplinary effort. He identified four variables which can be used for this ranking. These variables measured: the number of disciplines involved, the "distance" between them, the novelty and creativity involved in combining the disciplines, and the degree of integration between the disciplines. Although there is some subjectivity in ranking an interdisciplinary pursuit by these variables, they do provide a deeper understanding of the nature of an effort.

Another attempt to rank aspects of interdisciplinarity was made by Nicolescu in (Nicolescu). In his research, Nicolescu postulates that there are degrees of interdisciplinarity (see Table 2). Like Nissani's variables, quantifying the degrees of interdisciplinarity would be somewhat subjective, but helpful in understanding the potential of an interdisciplinary effort.

Degree	Description
Degree of application	The transfer of a method from one field into an application for another field. For example, when the methods from nuclear physics were transferred to medicine it led to the appearance of new treatments for cancer.
Epistemological degree	The nature of the knowledge in one field being transferred to another field. For example, transferring methods of formal logic to the area of general law to provoke an analysis of the epistemology of law.
Degree of the generation of new disciplines	The emergence of new disciplines as a result of combining knowledge from existing disciplines. An example of this was when the methods of mathematics transferred to physics and the field of mathematical physics was formed.

Table 2: The three degrees of Interdisciplinarity according to (Nissani, 1995).

The definitions and perspectives identified here provide a starting point for understanding the approaches and potential for the interdisciplinarity. It is a higher level concept than the other means described thus far; however, it is not the most comprehensive means being studied.

3.4.4 Transdisciplinarity

Transdisciplinarity is a concept that acknowledges the benefits of specialization and provides a framework for overcoming its pitfalls to meet broad societal needs. Its most basic definition can be derived from the word itself. The prefix "trans" means "transcendent" or something that goes across, through, or beyond something. The root word "discipline", as stated, refers to a "specific branch of scholarly knowledge".

Therefore, transdisciplinarity is something that transcends specific branches of scholarly knowledge. This “something” includes problems, solutions, knowledge, and more.

Other definitions for transdisciplinarity prevail in the literature. In (Seipel, 2004), Siepel cites a definition from Stemper describing transdisciplinary analysis as “concerned with the unity of intellectual frameworks beyond the disciplinary perspectives." Furthermore, Siepel states that it may “deal with philosophical questions about the nature of reality and the nature of knowledge systems that transcend discplines.” In (Nicolescu), Nicolescu states that “transdisciplinarity concerns that which is at once between the disciplines, across the different disciplines, and beyond all discipline. Its goal is the understanding of the present world, of which one of the imperatives is the unity of knowledge.” Finally, Klein performed a survey of many of these definitions in (Klein, 2003). As a result, she, proposed a definition to be “a higher stage of interaction [than crossdisciplinarity, multidisciplinarity, and interdisciplinarity] that entails an overarching framework that organizes knowledge in a new way and, in a new discourse, cooperation of multiple sectors of society and stakeholders in addressing complex problems.” This definition attempts to incorporate the key aspects of transdisciplinarity that have evolved over the course of its history.

Klein’s research in (Klein, 2003) also provided a contextual history of Transdisciplinarity and its diverse origins and applications. These origins have been traced to many sources including a theory on knowledge production and a theory on an open structure of unity in complexity. It has been used as a label for comprehensive frameworks like general systems theory, a descriptor of fields like philosophy, a type of educational reform, a form of holistic team-based collaborations, and a new approach to problem

solving. Its formal origin is most prominently credited to the first international conference on interdisciplinarity which was sponsored by the Organization of Economic Cooperation and Development (OECD) and held September 7-12, 1970 in France. The purpose of the conference was to examine the role of "pluridisciplinarity" and "interdisciplinarity" in the modern university. This conference and its participants laid the ground work for decades of study on transdisciplinarity.

Although there are diverse threads of meaning for transdisciplinarity, (Klein, 2003) was able to extract a series of shifts that help articulate a set of common themes. These thematic shifts are reproduced in Table 3. The left side of the table represents a sort of "status quo" disciplinary perspective. The right side of the table seems to show a maturing view of reality that acknowledges its multidimensionality.

Shift From:	Shift To:
segmentation	boundary crossing and blurring
fragmentation	relationality
unity	integrative process
homogeneity	heterogeneity and hybridity
isolation	collaboration and cooperation
simplicity	complexity
linearity	non-linearity
universality	situated practices

Table 3: Thematic shifts of Transdisciplinarity drawn from the historical definitions and discourses as identified by (Klein, 2003).

The shifts identified in Table 3 show a transition from a narrow disciplinary perspective to a broader more inclusive view of reality. They acknowledge a reality that consists of complex social interactions and natural ecosystems. They also bring to light the epistemological implications of transdisciplinarity and bring into question our human ability to deal with transdisciplinarity. According to (Klein, 2003), the problems that are

considered to be transdisciplinary are categorized as such because they are of mega size, complexity, and elusiveness. An example of a transdisciplinary problem is that of environmental sustainability. It is mega size, in that it can have global implications. It is complex because it deals with the delicate balance of natural ecosystems, economics, public policy, chemistry, biology, and more. Furthermore, it is elusive because there are so many interconnected considerations that it is incredibly difficult to understand the meaning of parameters and the impact of actions.

3.5 Barriers to Crossing Disciplines

3.5.1 Knowledge and Human-Factors

This research has already alluded to the fact that crossing disciplinary boundaries can be a difficult task. There are a number of factors that have been identified:

1. Collective human knowledge is increasing at an exponential rate
2. Proliferation of disciplinary fields
3. Individual human limitations
4. Increasing specialization narrows individual viewpoints
5. Specialized technical language is a barrier to "outsiders"
6. Problems that cross disciplines are both complicated and complex

Additionally, there are many other human factors that can become a barrier to crossing disciplines. People are diverse and human relations issues can hinder interdisciplinary activities. People work differently, they are motivated differently, and they are all subject to different failure modes at times. Sometimes these statements can be

true of organizations as well. Industrial and organizational psychology is a disciplinary field that is dedicated to studying these issues.

3.5.2 Organization, Tradition, and Disposition

In his 1960 paper "How do we get there?" ((Steel, 1995)), Jack Steel (see 4.2.3 Jack Steele's "Bionics") identified some problems that can be encountered when crossing the disciplinary boundaries. He illustrates a hypothetical (and rather amusing) relationship between engineers, biologists, and mathematicians in the then emerging interdisciplinary field of bionics. He characterizes three main barriers which he refers to as organization, tradition, and disposition.

First, the problem of organization is the difficulty in finding the right relationship between the disciplines. Steel uses the metaphor of a box of electronic components and wires to represent the disciplines of mathematics, biology, and engineering as components of bionics. In this metaphor, throwing the components into a pile does not lead to anything special. However, putting them together in just the right relationship can form a radio. The same components can also be put into other configurations to create new devices that serve other functions. In this metaphor, the whole is greater than the sum of its parts. Similarly, combining the disciplinary fields of bionics into different configurations can lead to new and interesting developments. Furthermore, this metaphor can be extended to include a broader group of disciplinary experts who can be assembled into many configurations for interdisciplinary collaborations.

Second, tradition is a problem because people presume certain relationships between these disciplinary fields. Following the previous metaphor, one might see some components and automatically think of a radio. However, if that is all they think of, then they miss the potential for other configurations that lead to new functions. So it is with the

disciplinary fields. A traditional configuration of the disciplines in Steel's example would be that the engineer designs the equipment that the biologist uses to collect data for the mathematician to analyze. Although this is valuable, one may miss the opportunity to consider how other configurations could be used to produce very interesting developments in other fields such as engineering. The bottom line is that tradition does not promote the numerous potential disciplinary relationships that define can lead to new knowledge and problem solving.

Third, disposition is a problem when one considers the stereotypical characteristics of a pure specialist in each of the three fields. Steel conjectures that the biologist has a mind for observation and analysis rather than creativity, the engineer has a disdain for the messy multivariate complexity of nature, and the mathematician is satisfied with manipulating abstract symbols that have no link to reality. Although, this analysis can be taken as rather tongue-in-cheek, it does illustrate certain biases that may be present.

3.5.3 Educational barriers

In the case of interdisciplinary educational programs, other risks become evident. On one hand an interdisciplinary program at the university level may run a risk of falling between the cracks as each parent discipline focuses on its own goals((Steel, 1995)). On the other hand, multidisciplinary programs that fall under a disciplinary department may be influenced too much by its parent discipline and lose its symmetry ((Harkness, 2002)). Regardless, education is a primary factor for the success of interdisciplinary activities.

3.6 Strategies for Crossing Disciplines

Once the barriers to interdisciplinary activities have been identified, strategies can be taken to mitigate or overcome them. The solutions that Steel identified to help

overcome the barriers for bionics ((Steel, 1995)) as a discipline were “unceasing education and explanation”, gadgets, and simple solutions. These can be examined further.

3.6.1 Education

Almost by definition, education will aid in bridging the disciplines. There are volumes written on the development of multidisciplinary, interdisciplinary, and other boundary crossing programs at the university level. These programs can develop individuals with knowledge in more than one discipline who can pay the essential role of “translator”. In (Steel, 1995), Steel makes it a point to discuss the importance of this role, but also acknowledges there are challenges to achieving it. There can be a negative perception of being a “generalist” or so-called “jack-of-all-trades-master-of-none”.

3.6.2 Demonstration

Perhaps the most compelling method of justifying interdisciplinary efforts is to show success. Academic, commercial, and governmental communities are more likely to be interested in interdisciplinary activities if they see results even on a small scale. Starting out with small achievable goals can go a long way toward building credibility. This has already taken place to a great extent and gained tremendous momentum. One model of demonstration is the creation of centers of interdisciplinary research and applied knowledge. There are many examples of this at universities and institutions including the Santa Fe Institute in New Mexico and the International Center for Transdisciplinary Research (CIRET) in France.

3.7 Interdisciplinary Innovators and their Tools

There are a number of systems and tools that have been developed to assist with interdisciplinary problems, collaborations, and knowledge transfer. At a practical scale, modern Internet tools like the semantic web and text processing can assist in the transfer of

knowledge between the disciplines ((Kostoff, 2002)). Additionally, knowledge management and social networking tools enable the discovery and flow of information both explicit and tacit. On a larger scale, Norbert Wiener's Cybernetics has had a great impact on interdisciplinary research. It has provided a common framework and transdisciplinary language for the study of all types of systems whether biological, technical, organizational, and political ((Francois, 1999)). A description of cybernetics and its history is beyond the scope of this research, but it contains powerful tool that can be leveraged going forward. Norbert Wiener himself is an excellent example of the power of interdisciplinary knowledge. Another individual who is less widely known in the western world, but also made a great contribution to interdisciplinary knowledge transfer is Genrich Altshuller – the father of TRIZ.

3.7.1 Genrich Altshuller's TRIZ

TRIZ is a Russian acronym for "Teorija Reshenija Izobretatel'skih Zadach" which loosely translates to "The Theory of Inventive Problem Solving". It was first developed in the 1940s and 1950s by Genrich Altshuller, an inventor whose bold tenacity for innovation was not always well received by the communist Russian establishment. A brief but fascinating biography of Altshuller is available from Lerner in (Lerner, 1991). Some highlights from this work are described here.

Altshuller received his first patent for an underwater diving apparatus in 9th grade. He continued to invent and eventually began work in a Russian patent office reviewing patents. This inspired him to do more than work on his own inventions. Rather, he wanted to help others learn how to invent. It was this desire that shaped the course of his life and the development of TRIZ. In December of 1948 Altshuller acted on this desire and wrote a seemingly brash letter to his country's leader, Joseph Stalin. The letter stated that there was

"chaos and ignorance in the USSR's approach to innovation and inventing" and that he had developed a theory that could help any engineer to invent. This letter changed the course of Altshuller's life as it eventually led to his arrest, imprisonment, and a captivating series of both tragic and fortunate events. Throughout all of which, he continued to innovate and develop his theory of inventive problem solving which ultimately enabled him to survive the Russian gulags (forced labor camps). While in these camps, he met many imprisoned scientists, lawyers, and architects who befriended him and taught him their fields as a distraction from prison life. Eventually, Stalin passed away and Altshuller was freed. In 1956, he published his first paper titled "Psychology of Inventive Creativity". He continued his work studying world wide patents and concluded that invention derives from problem analysis that reveals contradictions. Specifically, he determined that there were about 1500 contradictions that could be easily overcome by applying some common principles. After many years of diligence, his ideas started to receive acceptance. In 1969 he published a book called *Algorithm of Inventing* which described 40 Principles and the first algorithm to solve complex inventive problems. His ideas continued to develop and gain acceptance and in 1989 he formed the Russian TRIZ Association which has continued to grow even since his death in 1998. TRIZ has become highly regarded for its success in transferring inventions and solutions from one field to another

According to (Julian Vincent & Mann, 2002), TRIZ represents the largest study of human creativity ever conducted – encompassing 1500 person years of effort over the course of 50 years. Nearly three million successful international patents were searched and ranked by inventiveness in an attempt to develop a system to classify all known solutions in terms of function. This research inspired Altshuller to state three basic principles of TRIZ.

- Problems and solutions are repeated across industries and sciences.
- Patterns of technical evolution are repeated across industries and sciences.
- Innovations used scientific effects outside the field where they were developed.

Altshuller's research also observed that the most inventive solutions resolved a conflict between competing parameters known as "technical contradictions". A technical contradiction is a pair of conflicting parameters of a system. For example, strength-versus-weight is a classic contradiction in structures. The contradiction is that one must be traded for the other. For one to achieve high strength, it is often the case that additional weight is required. Conversely, for an object to be light weight, it is often the case that strength must be sacrificed. However, there are many instances where inventions have achieved both high strength and light weight. When one faces this type of problem, TRIZ tools can be used to facilitate a resolution to the problem.

3.7.2 Basic Concepts of TRIZ

TRIZ is a very structured and systematic approach to innovation. At a high level, it follows the general problem solving framework (Figure 1) that was described in Section 1.2. However, what makes TRIZ unique is the effort that went into its development and the extensive set of tools that were created for it. Figure 4 is an illustration of a classic TRIZ process from (Changquing, Zezheng, & Fei, 2005).

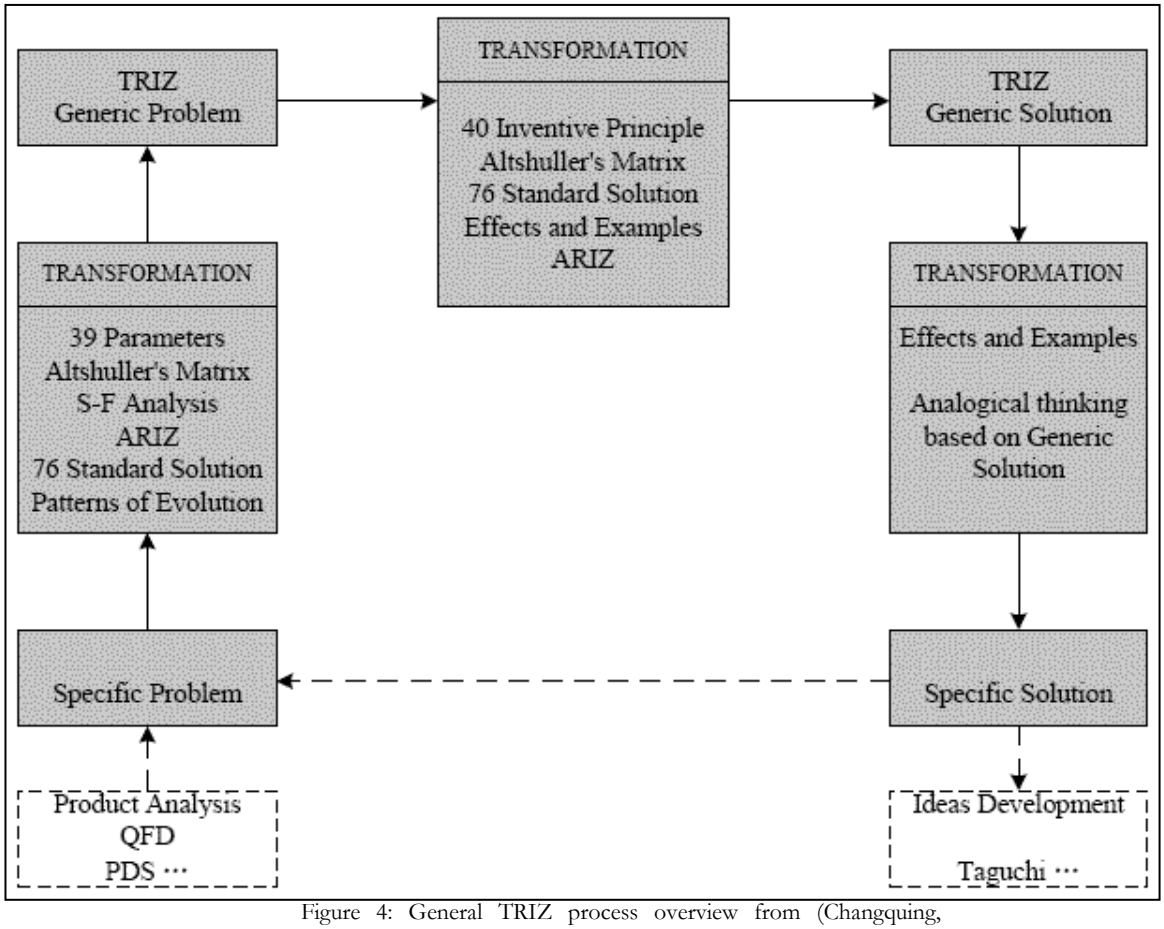

Figure 4: General TRIZ process overview from (Changquing, Zezheng, & Fei, 2005).

TRIZ is a collection of tools and techniques that facilitate the creation of a functional problem definition which can then be correlated with a set of known innovative solutions that have solved similar functional problems. This is done by mapping specific problems to generic problems which can be cross-referenced to their generic solutions using specific TRIZ tools. Some of the fundamental tools of TRIZ were summarized in an illustration by (Loebmann, 2002) which is shown in Figure 5.

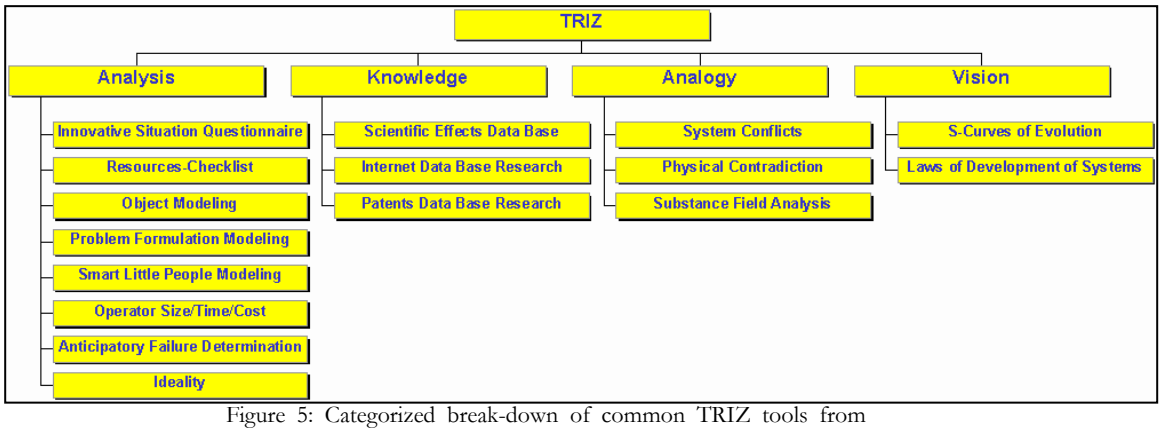

Figure 5: Categorized break-down of common TRIZ tools from (Loebmann, 2002).

The tools shown are organized by their function. The analysis tools are used to assist in problem definition. The knowledge tools are large databases of specific solutions to specific problems. The analogy tools are used to abstract problems so that they can be mapped to abstract solutions. The vision tools are used to consider the potential of a specific solution. Some of these tools will be described briefly.

Two of the most common TRIZ tools for performing this cross-reference are the "Contradiction Matrix" and the "40 Inventive Principles". An excerpt of a Contradiction Matrix is shown in Figure 6. A contradiction is defined as a pair of opposing parameters and an inventive solution is one that resolves that contradiction. The row & column headings contain common contradiction parameters that tend to result in a design trade-off. Using these headings, one can locate cells in the body of the table containing references to the most inventive generic solutions. The numbers in these cells correspond to entries in the list of "40 Inventive Principles". This list of only 40 inventive principles was the result of a comprehensive analysis that categorized and ranked the inventiveness of over 3,000,000 international patents. In TRIZ terms, this large pool of patents can be considered a "function" database which catalogs the various means of achieving a particular function.

	Worsening Feature → / Improving Feature ↓	Volume of moving object	Speed	Force (Intensity)	Stress or pressure	Shape	Reliability	Object-generated harmful factors	Ease of operation	Ease of repair	Device complexity	Difficulty of detecting and measuring
		7	9	10	11	12	27	31	33	34	36	37
9	Speed	7, 29, 34	+	13, 28, 15, 19	6, 18, 38, 40	35, 15, 18, 34	11, 35, 27, 28	2, 24, 35, 21	32, 28, 13, 12	34, 2, 28, 27	10, 28, 4, 34	3, 34, 27, 16
10	Force (Intensity)	15, 9, 12, 37	13, 28, 15, 12	+	18, 21, 11	10, 35, 40, 34	3, 35, 13, 21	13, 3, 36, 24	1, 28, 3, 25	15, 1, 11	26, 35, 10, 18	36, 37, 10, 19
11	Stress or pressure	6, 35, 10	6, 35, 36	36, 35, 21	+	35, 4, 15, 10	10, 13, 19, 35	2, 33, 27, 18	11	2	19, 1, 35	2, 36, 37
12	Shape	14, 4, 15, 22	35, 15, 34, 18	35, 10, 37, 40	34, 15, 10, 14	+	10, 40, 16	35, 1	32, 15, 26	2, 13, 1	16, 29, 1, 28	15, 13, 39
15	Duration of action of moving object	10, 2, 19, 30	3, 35, 5	19, 2, 16	19, 3, 27	14, 26, 28, 25	11, 2, 13	21, 39, 16, 22	12, 27	29, 10, 27	10, 4, 29, 15	19, 29, 39, 35
33	Ease of operation	1, 16, 35, 15	18, 13, 34	28, 13 35	2, 32, 12	15, 34, 29, 28	17, 27, 8, 40		+	12, 26, 1, 32	32, 26, 12, 17	

Figure 6: A portion of the TRIZ Contradiction Matrix from (Domb, 1997).

A simple example of a classic TRIZ process using the Contradiction Matrix and Inventive Principles was described in (Salamatov, 2005). This example has been reproduced in Table 4 for convenience. One can start to appreciate the thoroughness of this structure and its potential for "inventive problem solving". It minimizes the trial-and-error approach to design in favor of a guided thought process. Although this structure points an innovator in the right direction, it does not promise to make innovation easy. There is still a great deal of critical thinking required to move from a generic solution to a specific one.

Step	Description	Example
1	Select a product you want to improve.	"I want to improve a coffee cup".
2	If product consists of many parts, try to separate and focus on a specific part which causes a problem.	"A cup does not keep coffee warm for a long time. Therefore, we are interested in a new design of a coffee cup, and not in improving the coffee beans or maker."
3	Identify a parameter you want to improve.	"I want to keep coffee in the cup warm as long as possible. In other words, to make the temperature of the coffee as stable as long as possible."
4	Propose any method which will improve your technical parameter.	"I can keep coffee warm, for instance, by placing the cup on an electric heater."
5	Think of why you can not reach the desired improvement in a straightforward way by using the method proposed.	"In the case of using the heater, more electric energy will be consumed."
6	Formulate a contradiction in the following form: "I want to improve the parameter X. I can do it by doing (put what you can do) but the parameter Y gets worse."	"I want to improve the stability of the temperature of the coffee by providing external heating, but in this case energy consumption grows."
7	Use Altshuller's [Contradiction] matrix.	To improve: temperature. Gets worse: energy waste.
8	Find a cell in the matrix which is the intersection of vertical column and horizontal row for respectively the parameters you selected.	Improve: 17 – Temperature Worsening: 22 – Loss of energy
9	Use the list of inventive principles.	21 – Skipping 17 – Another dimension 35 – Parameter changes 38 – Strong oxidants
10	Interpret the recommended inventive principles in terms of your product.	
11	If no solution can be found, change the parameter that gets worse and return to step 5.	
12	If no solution can be found, redefine the parameter that you want to improve and return to step 3.	
13	If the Altshuller's Matrix does not help after several attempts, use Inventive Standards, Pointer to Physical effects, or ARIZ.	

Table 4: Steps and example of a classic TRIZ process using the Contradiction Matrix and Inventive Principles (Reproduced from (Salamatov, 2005)).

Some of the other notable tools are Substance-Field (S-F) Analysis and the Law of Increasing Ideality. S-F Analysis is a modeling technique to represent any technical system with a minimal number of elements. Specifically, it expresses a system in terms of one object (S1) acting (F) on another object (S2). The law of increasing ideality (also called the law of technical evolution) states that all successful innovations evolve in a direction of increasing ideality, where ideality is defined as more benefits, less cost, and less harm. There are a multitude of other tools and procedures that facilitate a TRIZ process. A full description of TRIZ and its tools is beyond the scope of this research, but additional information can be found in the bibliography.

Chapter 4

4 A BRIEF REVIEW OF NATURE INSPIRED DESIGN

Software and other forms of design depend on innovation to meet growing and changing demands. Incremental improvements are the basis for most software products today as they evolve from version to version. New features are introduced and existing features are improved upon. Breakthrough innovation, however, can be elusive. It is rare and often based on esoteric individuals and teams who serendipitously arrive at an innovation. Breakthrough innovations are almost, by definition, difficult to repeat. This, however, has not stopped individuals and institutions from striving to create them. To do this, some have turned to the natural world as a source of innovation. Nature inspired design in its various forms is a growing source of innovation for many fields of design including computer science. This chapter will provide a general overview to nature inspired design with its various forms and methodologies.

4.1 Nature Inspired Design

The natural world is the material world and its phenomena. It is also commonly referred to as the cosmos, the universe, nature, or the world (Dictionary, 2004). One can say with assurance that the natural world is by far the most complex system known to man. Its scale and diversity are beyond human measure at both the micro and macro levels. All areas of science are focused on studying the intricacies of the natural world, and harnessing it where possible. In spite of its innate complexity, there is an intrinsic order and balance to it. The natural world is flexible and diverse. It is dynamic and self-regulating. It adheres to a set of natural laws that govern the existence and behavior of everything in it. Whether one attributes the creation of the natural world to design or phenomena, it is undeniable that

it is the ultimate picture of systemic beauty and is the standard by which everything else is measured.

A paradox of nature is that in its complexity is an innate simplicity. Although nature is vast and intricate, it is not superfluous. There is intent to every design in nature. Everything has a purpose and is uniquely designed to fulfill that purpose. In (Bernsen, 2004), Bernsen describes how simplicity is the guiding principle of designs in nature. It furthermore states that nature "achieves simplicity in a demanding way by always acknowledging the complexity of the purpose at hand, whether in one single organism or in the interplay between a multitude of living species in a habitat." Highly regarded scientists have recognized the simplicity of nature's designs and its intuitive value.

Nature always tends to act in the simplest way.

– John Bernoulli (1696)

Everything should be made as simple as possible, but not simpler.

– Albert Einstein (1879-1955)

Creative people have taken design inspiration from the natural world for centuries, if not millennia. Those that were able to recognize or capture the simplicity, efficiency, functionality, and beauty of nature's designs in their own creations have taken their place in recorded history.

If one way be better than another, that you may be sure is Nature's way.

- Aristotle, Politics (350 B.C.)

Human subtlety will never devise an invention more beautiful, more simple or more direct than does Nature, because in her inventions, nothing is lacking and nothing is superfluous.

- Leonardo da Vinci (1452-1519)

This fascination and appreciation for the elegance of nature's designs have not diminished over time. Modern innovators still seek inspiration and insight from the natural world.

No matter what product you are designing, nature is always the best database. There is more in the world to be discovered than there is to be invented.

- Franco Lodato (2004)

These principles of nature inspired design have been and will continue to be used by artists, architects, designers, and engineers to drive innovation in many domains. Industrial design, material science, control systems, manufacturing are just a few popular

examples of fields that have benefited from mimicking nature. Computer Science can be included in this list, and is one focus of this research.

4.2 A Recent History of Nature Inspired Design

Although it may not be possible to determine the true origins of nature inspired design, there are converging paths in recent history. As scientific knowledge of nature increased in the early to mid-20th century, examples of nature inspired designs began to appear at the forefront of scientific research. Experts of varied learning and interdisciplinary efforts drove much advancement. By the mid-20th century, terms like "biomimetics" and "bionics" emerged in the United States as descriptors for nature inspired design. These and other terms like "biomimicry", "biologically inspired design", and "bioinspired design" are now largely considered to be synonymous. However, it is valuable to understand the major roots of these terms and the people who developed them. We intend to show that the advancements made in nature inspired innovation came forth through interdisciplinary applications, knowledge, and collaboration. To begin, we will describe an example of nature inspired design that predates the terms described above.

4.2.1 Warren McCulloch and Walter Pitts' Artificial Neural Networks

Artificial neural networks are self learning systems that are modeled after the interconnected system of cells called "neurons" in the human brain. Their intent is to imitate the brain's ability to "learn" from trial and error by recognizing relationships and patterns. In other words, their purpose is to enable an artificial system to learn from experience, much like humans do. This capability allows systems to be developed that can adapt to solve new problems without explicit coding by a trained developer. This powerful paradigm has had significant success in various application areas including time series

prediction, decision making, pattern recognition, and data mining to name a few. However, it was not these applications that drove the invention of artificial neural networks.

The earliest work on artificial neural networks took place in 1943 by a neurophysiologist named Warren McCulloch and a mathematician named Walter Pitts (McNeil, 1992). Together they modeled a simple neural network using electrical circuits and wrote a landmark paper called "A Logical Calculus Immanent in Nervous Activity". The intent of their work was as much to develop an understanding of human thought as it was to develop an "experimental epistemology." The logical calculus they proposed attempted to provide a rigorous and materialistic description of neural activity.

Because of this work, Warren McCulloch is now recognized as a pioneer in cybernetics, neurology, and the development of the computer. McCulloch's educational background is described by (APS, 2000). He completed his bachelor's degree in philosophy and psychology at Yale in 1921 and his masters in psychology from Columbia University in 1923. He went on to receive his MD in 1927 from the College of Physicians and Surgeons in New York to further develop his understanding of the nervous system.

Little biographical information is known about Walter Pitts. He was only 20 years old with no formal college degree when he published the seminal paper with McCulloch. In spite of this, he had already worked with prominent scientists in logic and mathematical biology. He was known for his aptitude in logic and mathematics and went on to contribute to early conferences on cybernetics in the 1940s and 1950s. (Easterling, 2001)

Together, McCulloch and Pitts made a powerful impact on the field of computer science without ever intending to do so. Over 60 years ago, their attempts to better

understand human thought launched an entire new field of study leading to innovations that are still being developed today.

4.2.2 Otto Schmitt's "Biomimetics"

Otto Schmitt, a man of varied learning and a talent for connections, is credited with coining the term "biomimetics". Born in St. Louis, Missouri in 1913, Schmitt grew up in a well educated family and was able to pursue interests in electrical engineering, biology, physics, and mathematics. Schmitt's formal training was as a scientist and he spent much of his career during the 1950s, 60s, and 70s as a faculty member at the University of Minnesota. While there, he was uniquely appointed to a joint "biophysics" program between the biology and physics departments. By 1957, he had conceived what would later be known as "biomimetics". He believed that fundamental biological phenomena can be understood in relatively simple physical and chemical terms once the painstaking effort has been made to study them adequately by quantitative biophysical methods ((Harkness, 2002)). This concept defined much of his career even from his early years. His doctoral research included the development of a device that mimicked the electrical action of a nerve. Later in his career, he and his students would continue this pursuit and perform extensive research into biological nervous systems including the extraction and testing of actual nerve fibers from squid, lobster, and other specimens.

Schmitt spent a great deal of his career contributing to the field of "biophysics". In one of his papers called "The Emerging Science of Biophysics", he described the topic as follows:

"Biophysics is not so much a subject matter as it is a point of view. It is an approach to problems of biological science utilizing the theory and technology of the physical sciences. Conversely, biophysics is also a biologist's approach to problems of physical science and engineering, although this aspect has largely been neglected." (Harkness, 2002)

Schmitt eventually designated the second point in this description with the word "biomimetics". It is not clear of the exact date which this word was first used, but it appeared in a conference paper that he wrote in 1969. In 1974, the word made its first appearance in a dictionary with this definition:

"the study of the formation, structure, or function of biologically produced substances and materials (as enzymes or silk) and biological mechanisms and processes (as protein synthesis or photosynthesis) especially for the purpose of synthesizing similar products by artificial mechanisms which mimic natural ones." (Harkness, 2002)

Otto Schmitt's contribution to the subsequent field of biomimetics was more than that of linguistics. His broad expertise allowed him to draw meaningful connections between diverse academic disciplines. This was true in both a technical and social capacity. He was able to link concepts in nature with concepts in technology. Furthermore, he was a networking hub that facilitated professional and social connections between individuals and organizations. Throughout his career at the University of

Minnesota, Schmitt traveled the nation and the world trying to establish biophysics as a unified discipline. He played in important role in founding a number of professional organizations including the IEEE Engineering in Medicine and Biology Society, the Biophysical Society, the Biomedical Engineering Society, the Association for the Advancement of Medical Instrumentation, the International Federation of Medical and Biological Engineering, and the International Union of Pure and Applied Biophysics. (Harkness, 2002)

4.2.3 Jack Steele's "Bionics"

The word "bionics" was coined by Jack Steele of the US Air Force at a Wright-Patterson Air Force Base in Dayton, Ohio in 1960. Steele was another man of varied learning and was born in 1924 in Lacon, Illinois. He attended the University of Illinois in Champaign until he was drafted in 1943 into the Army Specialized Training Program. He was trained at the Illinois Institute of Technology in engineering and then studied pre-medicine at the University of Minnesota. Upon discharge from the army in 1946, he completed medical school at Northwestern University Medical School. During that time, he worked in a research fellowship with a Dr. Ray Snider to study the effects of drugs on a rabbit's brain. He also spent one summer studying atomic physics at the University of California at Berkeley. He audited courses by Fermi and Oppenheimer and strove to better understand semiconductors for use in a "thinking machine". In 1951, he was drafted by the Army once again where he would serve as a doctor. Starting out as a first Lieutenant, he retired from the Army as a Colonel in 1971. (Gray, 1995)

In August of 1958, Steele started using the term "bionics" to represent the use of biology to solve design and engineering problems. This application of biology was not new, but had not been recognized as a formal discipline. He believed that naming it would

facilitate its wider adoption as a field. In June of 1959 the term was first documented in a letter to the Committee on Bioelectronics. The term was constructed from the Greek word "bion" meaning a unit of life with an emphasis on function rather than form ("morphon") and "ics" being a common suffix for areas of disciplinary studies as in mathematics and physics. He defined the term as:

> *"The discipline of using principles derived from living systems in the solution of design problems." (Steel, 1995)*

Jack Steele was also a man of connections. Not only did he connect the disciplines of biology, physics, mathematics, and engineering; he also formed interesting social connections. During his research fellowship at Northwestern, Ray Snider introduced him to Warren McCulloch – the co-inventor of artificial neural networks (See section 4.2.1). Together, they would discuss neural operation, logic, and engineering. Additionally, he was impacted by a brief meeting with Norbert Wiener, the man who created the field of Cybernetics. He was impressed by Wiener's conjecture that mathematics is at best an approximation of reality.

4.2.4 Janine Benyus' "Biomimicry"

A more contemporary development in nature inspired design is the "biomimicry" movement renewed by Janine Benyus in 1997 with her book *Biomimicry: Innovation Inspired by Nature*. Benyus is a biologist, author, and speaker from Stevensville, Montana who was educated at Rutgers University. Her book and international lecture tours have been a catalyst for the awareness and adoption of biomimicry as a source of innovation.

Her dedication to biomimicry led her to found two organizations dedicated to advancing nature inspired design – the Biomimicry Guild and the Biomimicry Institute. She and her organizations have become quite influential as advisors to many commercial, educational, and governmental organizations. The Biomimicry Guild is a for-profit company founded in 1998. It offers education, research, and consulting services to product development organizations, often in the form of a "Biologist at the Design Table". In this offering, a biologist is contracted to facilitate a "Biomimicry Design Process" to address a design problem through careful inspection of nature's solution to similar problems. (The Biomimicry Design Process is described in detail in section 4.4.) The Biomimicry Institute was founded in late 2005 as a not-for-profit organization whose mission is to "promote the transfer of ideas, designs, and strategies from biology to sustainable human systems designs." Shortly after its establishment, the Biomimicry Institute embarked on an initiative with another environmental nonprofit organization called the Rocky Mountain Institute, to develop a "Biomimicry Database". The Biomimicry Database was designed to facilitate the aggregation and sharing of biomimicry knowledge, literature, and products. On February 22, 2006 Benyus hosted a "Biomimicry Portal Workshop" in Toronto, Ontario ((2006b)) to advance the development of this database. Those invited to the workshop included leaders from the worlds of web search engines, open source, wikis, scientific publishing, ontology, digital libraries, as well as users from biology, engineering, design, etc. An invitation to the workshop described the portal as "a bio-inspiration website where innovators can learn from nature's solutions, where biologists can find a whole new audience for their research, and where collaborators can cross-fertilize to create

sustainable, bio-inspired designs ((*J. Benyus, 2005)*). The portal itself describes its purpose as:

> *"The Biomimicry Database is intended as a tool to cross-pollinate biological knowledge across discipline boundaries. It will be a place where designers, architects, and engineers can search biological information, find experts, and collaborate, to find ideas that potentially solve their design/engineering challenges. It attempts to bridge the gaps of terminology and specialization that separate biologists, chemists, and other researchers from engineers and other developers in industry. It is a moderated open-source tool, which makes it not only a knowledge source but also a collaboration forum for researchers in disparate fields." (Biomimicry, 2007)*

An alpha-prototype release of the Biomimicry Database was released in 2006 and can be found at http://database.biomimicry.org. Its use of both biological and technical language allows nature's solutions to be found by both biologists and engineers. Thus, it serves as a sort of "Rosetta stone" to translate between these disciplinary fields.

4.3 Facets of Nature Inspired Design

Biomimicry, bionics, and nature inspired design have now been established as nearly synonymous terms that describe a concept where designers use the natural world as a source of innovation. There are, however, several ways that this innovation can be mined and applied to human designs. Podborschi & Vaculenco in (Podborschi & Vaculenco,

2005) and Lodato in (Lodato, 2005) describe five classifications of Bionics. For convenience, these classifications are reproduced in Table 5.

Classification	Description
Inspiration	Used as a trigger for creativity (for example, the design of London's Crystal Palace inspired by water lily)
Abstraction	The use of an isolated mechanism (for example, fiber reinforcement of composites)
Non-Biological Analogy	Functional mimicry (for example, modern planes and the use of airfoils)
Partial Mimicry	A modified version of the natural product (for example, artificial wood)
Total Mimicry	An object or a material or chemical structure that is indistinguishable from the natural product (for example, early attempts to construct flying machines)

Table 5: Classifications of Bionics as described by (Podborschi & Vaculenco, 2005) and (Lodato, 2005).

The classifications of Bionic methods range in both rigor and intent. Each form has led to innovations and can be applied to problem solving design. Inspiration is perhaps the least structured form of Biomimicry. As stated, it simply triggers a creative idea that may not be representative of the inspiring natural element. Total Mimicry is at the other end of the spectrum and is an attempt to essentially recreate a natural design through human means.

Abstraction is a key form of Biomimetic design. It involves the identification of an underlying principle in the natural world, then interpreting or translating it into a technical solution. An excellent example of this is the design of products based on the "Lotus Effect", which is a self-cleaning principle for surfaces abstracted from the Lotus plant. This unique plant has a remarkable quality in that dirt and grime do not to stick to its leaves. It was observed that their textured surface kept a layer of air between soiling

particles and the leaf itself. This layer of air prevents sticking so well that a drop of oil will roll off the surface of a Lotus leaf like a marble. The underlying principle is now being designed into metal for self-cleaning surfaces.

Beyond the classifications of nature inspired design, Benyus has described three aspects that frame the benefits of biomimicry (J. M. Benyus, 2002). These aspects may help a designer formulate a point-of-view that guides them to leverage natural designs and processes as a source of innovation. These aspects are shown in Table 6.

Aspect	Description
Nature as Model	Biomimicry is a new science that studies nature's models then imitates or takes inspiration from these designs and processes to solve human problems.
Nature as Measure	Biomimicry uses an ecological standard to judge our innovations. After 3.8 billion years of evolution, nature has learned: What works. What is appropriate. What lasts.
Nature as Mentor	Biomimicry is a new way of viewing and valuing nature. It introduces an era based not on what we can extract from the natural world, but on what we can learn from it.

Table 6: Aspects of Biomimicry according to (J. M. Benyus, 2002).

"Nature as Model" is a practical aspect that establishes nature as a "database" of design ideas. "Nature as Measure" is an interesting aspect that does not depend on using biomimicry. Rather, it forces a designer to ask questions about the designs they do create. "Nature as Mentor" is a much more relational construct that fosters an appreciation for the natural world. This appreciation should increase a designer's awareness and respect for nature. Thus, it fosters the first two aspects of biomimicry. These aspects represent valuable thought processes that can be build the ground work for a global tipping point toward innovation, environmental sustainability, and green design.

4.4 Biomimetic Design Methodologies

Although there is a great deal of literature that provides conceptual introductions or commentary on biomimicry, there are surprisingly few sources that discuss methodologies. Noted by (Julian Vincent, Bogatyreva, Bogatyrev, Bowyer, & Pahl, 2006), there is no general theory of biomimetics. Seemingly every new biomimetic design is developed using its own unique process. Regardless, there have been several attempts to capture the steps for biomimetic design. "Bionic Association" is briefly described by (Changquing, Zezheng, & Fei, 2005), the "Bio-Design approach" is described by (Lodato, 2005), the "Biomimicry Design Process" is described by (Biomimicry, 2006a), and "Biomimetic TRIZ" is described by (Julian Vincent, Bogatyreva, Bogatyrev, Bowyer, & Pahl, 2006), (Julian Vincent & Mann, 2000), and (Julian Vincent & Mann, 2002). Ultimately, all these processes provide steps for a design team to consider the question "How would nature solve this problem?"

4.4.1 Bionic Association

Bionic Association is an innovation methodology described by (Changquing, Zezheng, & Fei, 2005). This much generalized approach recognizes that organisms are good examples of "correct" ways to solve problems. Therefore, when one faces a design problem, simply look for an organism that solves that same problem and use it's pertinent mechanism as a reference for the a new artificial design.

Step	Description
1	Observe the organisms' behavior carefully. Take the phenomena of the organisms as the association objects.
2	Analyze the mechanism of the phenomena of the organisms system.
3	Analyze the practical problem. Develop a bionic idea into a problem-solving method or product.

Table 7: Steps in "Bionic Association" by (Changquing, Zezheng, & Fei, 2005).

4.4.2 The Bio-Design Approach

The Bio-Design Approach described by (Lodato, 2005) is nearly identical to Bionic Association, with the notable addition of translating the biological systems into a language familiar to the designers. This important step begins to develop a mechanism for enhancing communication across domains and enables the possibility of codifying the knowledge gained during a design initiative for future reuse.

Step	Description
1	Select features of a living organism that exceed current technological capabilities.
2	Derive principles and processes responsible for their superiority.
3	Develop models and methods to describe biological systems in terms useful to designers.
4	Demonstrate the feasibility of translating this knowledge into dependable and efficient hardware.

Table 8: Steps in the "Bio-Design approach" by (Lodato, 2005).

Both (Bernsen, 2004) & (Lodato, 2005) feature an interesting example of the Bio-Design process which is described here. In 1989, an Italian sports equipment manufacturer named CAMP was about to celebrate 100 years of operation. To celebrate the occasion they worked with designer Franco Lodato to redesign one of their core mountaineering products – the ice axe. The design brief described the need for a multifunctional ice axe that is lightweight with high structural strength and a good grip and can be used in variable positions to penetrate ice. It had to withstand the extreme conditions at altitudes over 5000 meters and temperatures of -20C. With that understanding, Lodato began the Bio-Design process as follows:

Step 1: Lodato contacted a Dr. Moja who was the director of the Natural Science Museum in Milan, Italy to help him identify living organisms that exceeded the

technological capabilities of a typical man-made "hammer". Two organisms emerged as prime examples: the rock lobster and the woodpecker. A rock lobster is known to hammer mussels on rocks with an impact that produces sound waves over 120 dB. A woodpecker weighing only 500g can deliver up to 25 hits per second with an impact of 25g/mm^2, without damaging its spine or brain.

Step 2: Lodato and team selected the woodpecker as the best model. They determined that the woodpecker's body was uniquely designed for this quick hammering motion which is capable of penetrating the hard surface of wood. The design of its spine and the spring action of its tail, when used as a brace, allow it take advantage of its center of gravity as a point of leverage to create high rotational speeds. The configuration of the bones in its skull also allows it to absorb the considerable stress associated with impact. These unique characteristics allow the woodpecker to use its whole body to effectively hammer a tree to withdraw insect larvae.

Step 3: The model of the woodpecker represented principles of simple machines that were presumably familiar to designers. Specifically, the woodpecker model incorporated the lever and the spring. Using its center of gravity as the fulcrum of a lever, the woodpecker is able to create high rotational speed while reducing the amount of load applied to its body. This speeds the blow, which according to Newton's Third Law causes a repercussion. The potential energy resulting from this repercussion is then stored in the spring action of the tail. This spring action is then used to return the woodpecker's beak to its original position at the point of impact over and over again in rapid succession. In this specific case, the added curvature of the woodpecker's spine is also used as a first-class lever and bar spring – thus improving the efficiency of the blow.

Step 4: The principles learned here were then presumably translated into an effective design which was demonstrated through prototypes. Eventually, these prototypes led to a final design which became highly successful for CAMP.

The steps described for this example are just one of at least three bionic inspirations used during the Bio-Design effort of Lodato's ice axe. The second was a hinge mechanism that joins the aluminum point with the inner titanium core of the handle which was inspired by the two valves of a mollusk. The third was the handle grip which was inspired by the epidermis of a shark.

4.4.3 The Biomimicry Design Process

The Biomimicry Institute has developed a design process described in (Biomimicry, 2006a) that can promote the transfer of ideas, designs, and strategies from biology to human systems designs. This relatively detailed approach also introduces an important new step to enable communication across domains, albeit in an opposite order from the Bio-Design Approach. The Biomimicry Design Process describes a step to translate the problem statement into biological terms, thus presenting more opportunity to leverage biologists to identify innovative solutions in nature. Table 9 shows all seven steps in this process.

Step	Name	Description
1	Identify	"Develop a Design Brief of the Human need"
2	Translate	"Biologize the question; ask the Design Brief from Nature's perspective"
3	Observe	"Look for the champions in nature who answer/resolve your challenges"
4	Abstract	"Find the repeating patterns and processes within nature that achieve success"
5	Apply	"Develop ideas and solutions based on the natural models"
6	Evaluate	"How do your ideas compare to the successful principles of nature?"
7	Identify	"Develop and refine design briefs based on lessons learned from evaluation of life's principles"

Table 9: Steps of the Biomimicry Design Process (Biomimicry, 2006a).

The steps in this process are also illustrated graphically in what is called the "Biomimicry Design Spiral" that is shown in Figure 7. The image communicates the iterative (evolving) nature of this methodology by mapping the steps over the spiral design of a Nautilus shell. The Nautilus shell is an appropriate symbol in that it is based on a spiral design with a ratio that is found throughout the natural world and human designs. This repeated design pattern consists of a Fibonacci Series of measurements that is so prolific, that it has been dubbed "The Golden Ratio" ((Livio, 2002)). A specific example of its use in biomimicry is a Mollusk-inspired fan developed by PAX Scientific (USA). This fan design has reportedly reduced energy requirements by up to 85% and noise by up to 75%.

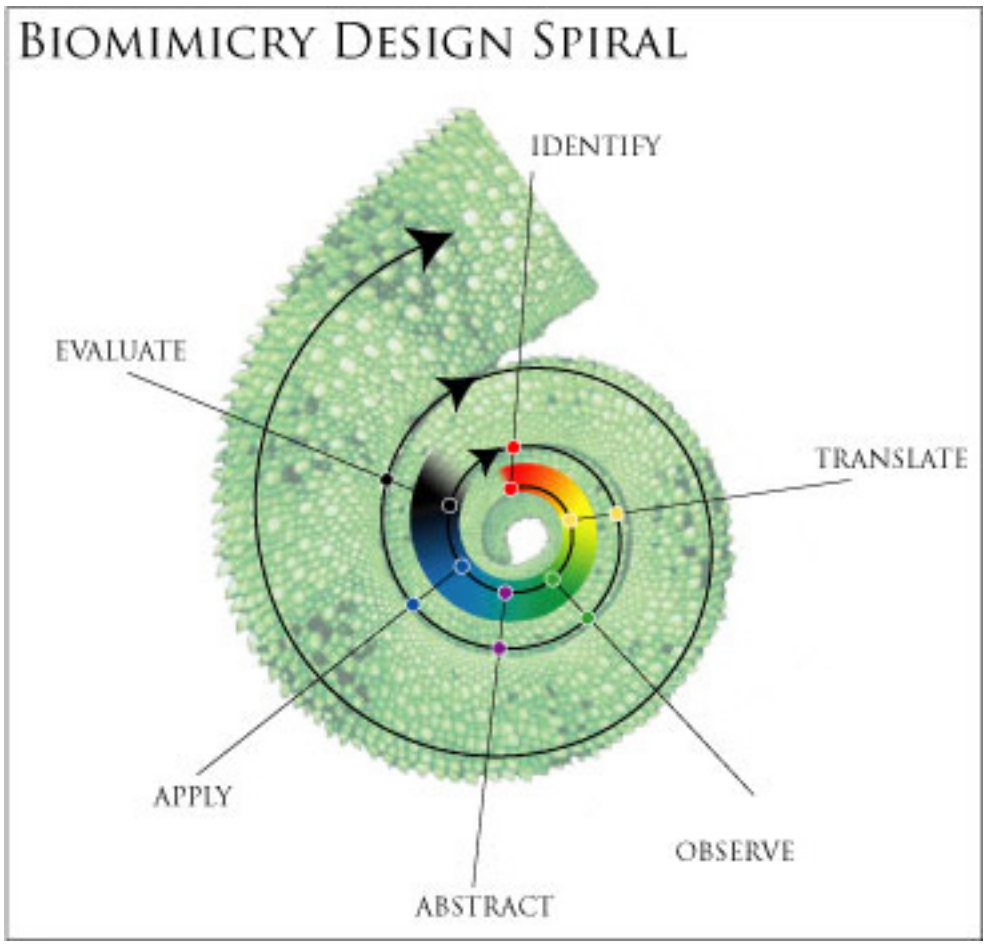

Figure 7: Biomimicry Design Spiral by (Biomimicry, 2006a). Figure reprinted with permission from copyright owner.

Step 1 (Identify) is to develop a design brief that describes the real challenge at hand. Essentially, this step is to document the requirements for the design in such a way that it does not imply a particular solution. This is a traditional step in any design process, but it is worthwhile to give warning that it is a natural tendency to take preconceived notions that drive a particular solution.

Step 2 (Translate) is to translate the design brief into a list of essential functions. These functions will then be used to generate biological questions from Nature's perspective. For example, it is beneficial to ask "How does Nature do this?" and "How does Nature NOT do this?" initially. These questions can be expanded by placing

additional criteria or conditions under which the function is achieved. For example, you might ask "How does nature achieve this function in this environment or under these specific climatic, social, or temporal conditions in this habitat?" These questions will help to narrow down the field of search for natural models.

Step 3 (Observe) is to look for biological designs that answer/resolve the challenges posted in the translation step. Consider the problem from all angles in both a literal and metaphorical sense. Next, seek organisms that are most challenged by it. Seek to identify organisms whose very survival depends on their means to solve this design challenge. There are several approaches to doing this. First, would be to research periodicals, literature, and textbooks on the subject. Second, collaborate with Biologists and other specialists. Their expertise can greatly enhance the quality and quantity of organisms identified. Third, just take a walk outside and observe the organisms and ecosystems that may be doing what you want to do.

Step 4 (Abstract) is to abstract repeating patterns and processes. There are usually many examples of natural solutions to design challenges. Some may be very similar and others quite different. In this step, create a taxonomy of nature's strategy. After building this taxonomy, abstract the repeating principles that allow this strategy to overcome the design challenge at hand.

Step 5 (Apply) is to generate a list of concepts that apply the lessons learned from the sources identified in step 4. These concepts could be inspired by mimicking form, function, or ecosystem. The deeper the understanding of the natural solution, the more likely it is that mimicry will work.

Step 6 (Evaluate) is to evaluate the concepts by comparing them to successful principles of nature. There are many patterns and principles in nature when it comes to design. For example, "Life builds from the bottom-up." This principle can be manifested through modularity, self-assembly, waste-free designs, and more. In this step, evaluate the concepts generated in step 5 based on some successful natural principles. Many of these natural principles were captured by the Biomimicry Guild in an illustration which has been reproduced in Figure 8.

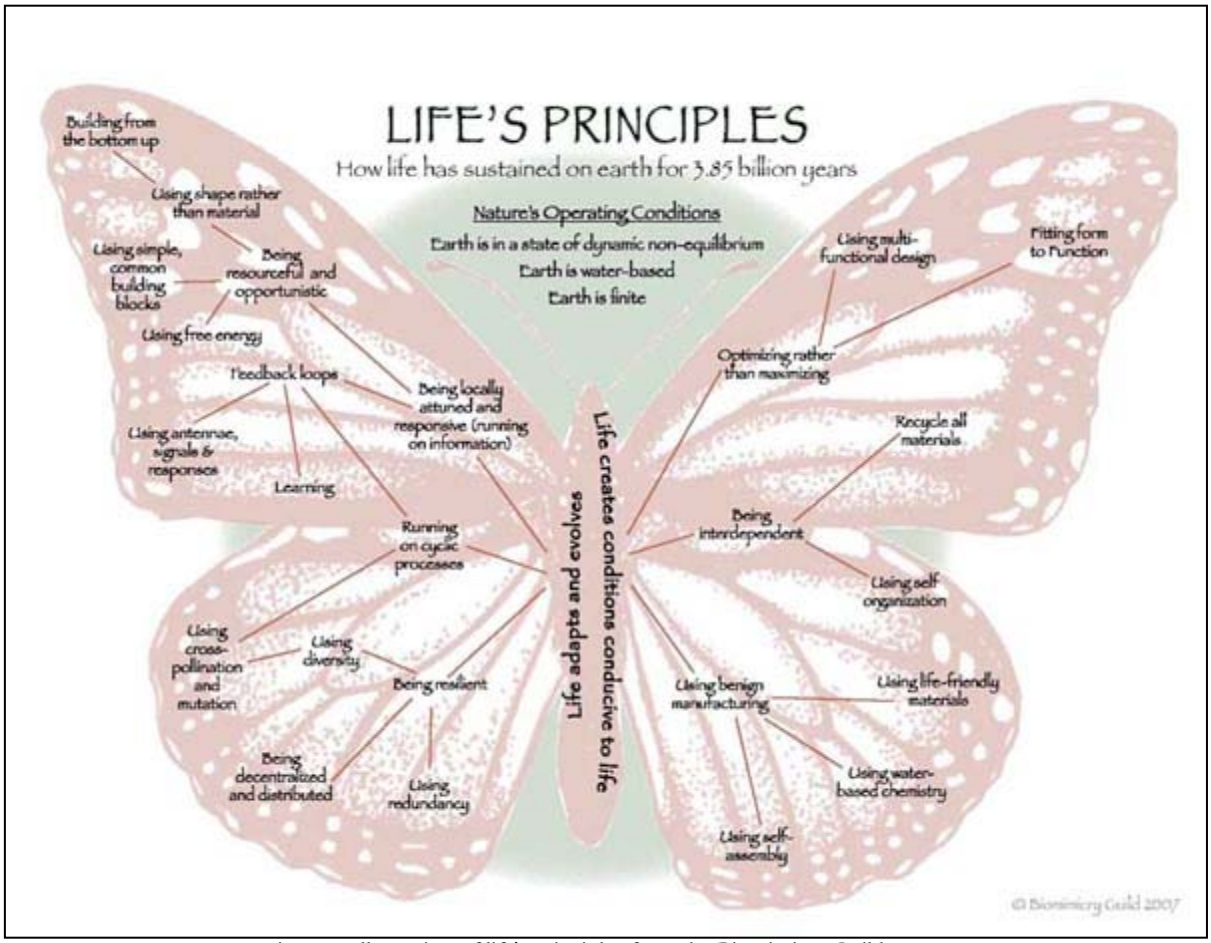

Figure 8: Illustration of life's principles from the Biomimicry Guild 2007 (Biomimicry, 2007). Figure reprinted with permission from copyright owner.

Step 7 is to begin the cycle again for refinement. Take an iterative approach by repeating all the design steps in this process. Nature itself operates with small feedback loops, continuous learning, and adaptation. These principles are also part of many human design processes such as rapid prototyping and agile software development. Frequent iterations with minor refinements can increase our learning, refine our designs, and mitigate risk.

Another version of this evolving Biomimicry Design Process was introduced in late 2007 on the Biomimicry Website (2007). This refined process shown in Figure 9 replaces the "Identify" step with one called "Distill". The steps are very similar, but the newer step is simplified with and emphasis on the purpose of the design. Furthermore, the "Observe" step has been renamed to "Discover", which is perhaps more representative of the various ways one can learn about nature's models. Finally, the "Abstract", and "Apply" steps have been replaced with a new step called "Emulate". Again, the new step appears to reflect a shift toward a simplified, but broader approach to mimicking the natural models through brainstorming and continuously scrutinizing the biological models.

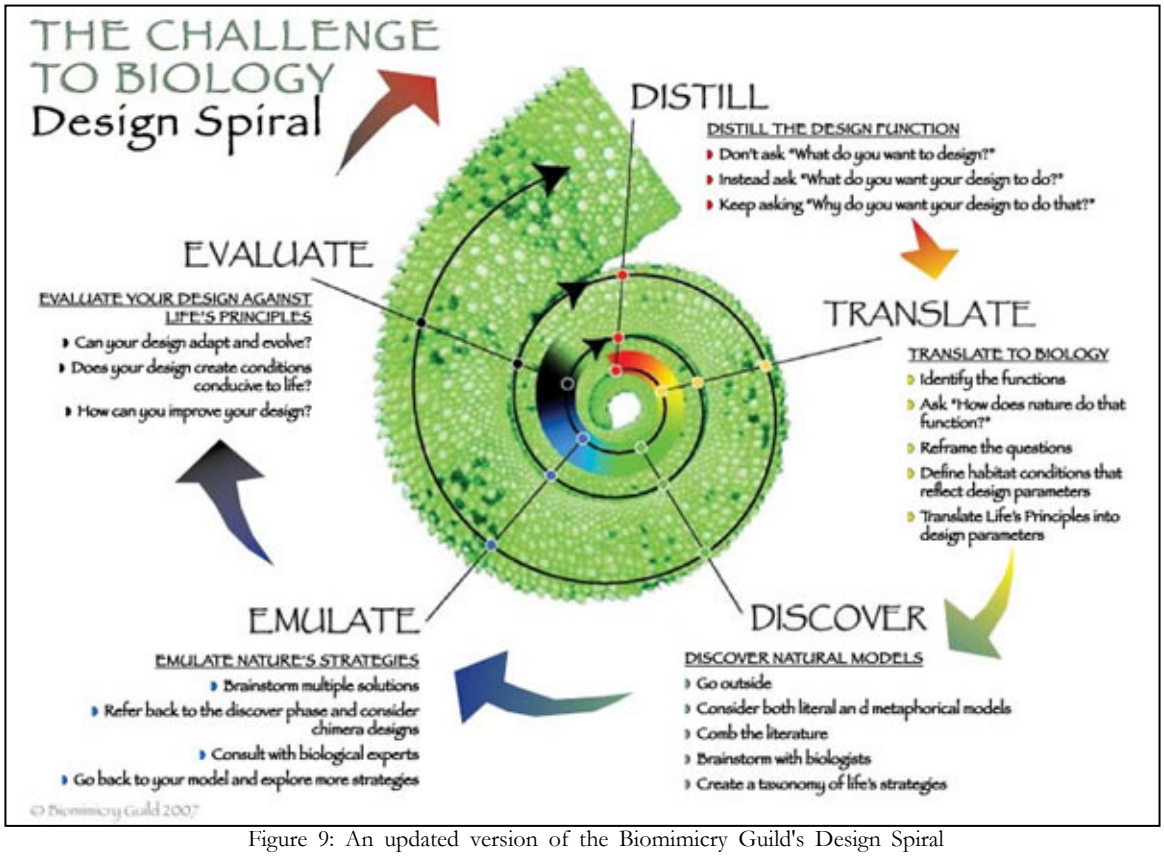

Figure 9: An updated version of the Biomimicry Guild's Design Spiral found in (2007). Figure reprinted with permission from copyright owner.

4.4.4 Biomimetic TRIZ

Biomimetic TRIZ was proposed by Vincent and Bogatyreva, et al in (Julian Vincent, Bogatyreva, Bogatyrev, Bowyer, & Pahl, 2006), (Julian Vincent & Mann, 2000), (Julian Vincent & Mann, 2002) and (Bogatyrev, Pahl, & Vincent, 2002) and is perhaps the most structured and comprehensive approach to biomimicry. Their approach extends an established method of systematic innovation called TRIZ which is recognized for its success in integrating knowledge from disparate domains. Considering this, Biomimetic TRIZ may provide the most opportunities to enhance communication across domains.

The earliest reference to Biomimetic TRIZ can be found in (Julian Vincent & Mann, 2000), which describes an educational experiment of applying TRIZ processes to biology. In this experiment, a class of biology students were given a couple of the classic TRIZ tools and asked to apply them to biological design problems. This experiment implies that nature may face the same set of contradictions, but (Julian Vincent & Mann, 2002) goes on to suggest that nature is not bound by the same set of Inventive Principles that have been identified in TRIZ. In fact, (Julian Vincent, Bogatyreva, Bogatyrev, Bowyer, & Pahl, 2006) indicates that there is only about a 12% similarity between biological solutions and technological solutions. The important implication is that TRIZ is not a fully exhaustive system and that nature can provide us with many new ways to solve problems. Based on this, Biomimetic TRIZ is an effort to expand the reach of TRIZ with a database of nature's solutions.

One of the first tools developed to extend TRIZ for biomimicry is the "Biological Effects Database" described in (Bogatyrev, Pahl, & Vincent, 2002), which serves as a biological equivalent to the patent database used to develop classic TRIZ. Its purpose is to catalog nature's solutions by function. To do this, it was necessary to expand certain TRIZ definitions. Classic TRIZ defines a system as an energy source, an energy transformation device, and an engine and a controller. In this definition, a human operator is considered part of the control subsystem. For the purposes of Biomimetic TRIZ, alternate definitions were required. First, a biological system was defined as "a living system that performs functions to realize its goals, while affecting the environment." A "biological function" then, is the action needed to achieve this goal or "biological effect". This facilitated a new definition for a "technical system", which is a biological system in which some functions

are delegated to technical (non-living) devices. The function of a technical system is the action needed to achieve the useful/desired future condition with the help of a technical device. The result of the technical function is the technical effect. These expanded concepts and definitions have facilitated the creation and functional organization of a biological database of nature's solutions.

Continuing work on Biomimetic TRIZ described in (Julian Vincent, Bogatyreva, Bogatyrev, Bowyer, & Pahl, 2006) and its citations included an analysis of approximately 500 biological phenomena covering over 270 functions and 2500 technical contradictions with their resolutions. To aide in this analysis, Vincent, et al developed a framework based on six fields of operation which can describe all actions with any object. Aligning to the maxim "things do things somewhere", (Julian Vincent, Bogatyreva, Bogatyrev, Bowyer, & Pahl, 2006) claims that these six fields of operation "re-organize and condense the TRIZ classification both of the features used to generate conflict statements and the inventive principles". This new framework was used to create two new tools: PRIZM and Biomimetic TRIZ. PRIZM (the Russian acronym for "The Rules of Inventive Problem Solving Modernized") is a new matrix for identifying the inventive principles of classic TRIZ. BioTRIZ is a new matrix for identifying the inventive principles defined by the biological effects database. These new tools of can be seen in Figure 10.

PRIZM matrix derived from standard TRIZ matrix.

fields	substance	structure	space	time	energy	information
substance	6 10 26 27 31 40	27	14 15 29 40	3 27 38	10 12 18 19 31	3 15 22 27 29
structure	15	18 26	1 13	27 28	19 36	1 23 24
space	8 14 15 29 39 40	1 30	4 5 7–9 14 17	4 14	6 8 15 36 37	1 15–17 30
time	3 38	4 28	5 14 30 34	10 20 38	19 35 36 38	22 24 28 34
energy	8 9 18 19 31 36–38	32	12 15 19 30 36–38	6 19 35–37	14 19 21 25 36–38	2 19 22
information	3 11 22 25 28 35	30	1 4 16 17 39	9 22 25 28 34	2 6 19 22 32	2 11 12 21–23 27 33 34

PRIZM matrix derived from biological effects: BioTRIZ.

fields	substance	structure	space	time	energy	information
substance	13 15 17 20 31 40	1–3 15 24 26	1 5 13 15 31	15 19 27 29 30	3 6 9 25 31 35	3 25 26
structure	1 10 15 19	1 15 19 24 34	10	1 2 4	1 2 4	1 3 4 15 19 24 25 35
space	3 14 15 25	2–5 10 15 19	4 5 36 14 17	1 19 29	1 3 4 15 19	3 15 21 24
time	1 3 15 20 25 38	1–4 6 15 17 19	1–4 7 38	2 3 11 20 26	3 9 15 20 22 25	1–3 10 19 23
energy	1 3 13 14 17 25 31	1 3 5 6 25 35 36 40	1 3 4 15 25	3 10 23 25 35	3 5 9 22 25 32 37	1 3 4 15 16 25
information	1 6 22	1 3 6 18 22 24 32 34 40	3 20 22 25 33	2 3 9 17 22	1 3 6 22 32	3 10 16 23 25

Figure 10: PRIZM and BioTRIZ matrices from (Julian Vincent, Bogatyreva, Bogatyrev, Bowyer, & Pahl, 2006).

The biological effects database, the PRIZM matrix, and the Biomimetic TRIZ matrix added to the classic TRIZ framework provide a powerful toolset for the development of biomimetic solutions. They provide tangible and detailed access to natures solutions without requiring the involvement of a trained biologist. They also provide the means for the methodology that was proposed in (Julian Vincent, Bogatyreva, Bogatyrev, Bowyer, & Pahl, 2006). The steps for this methodology have been reproduced in Table 10.

Step	Description
1	Define the problem in the most general, yet precise way. Avoid limiting terminology or thoughts. Then list the desirable and undesirable properties and functions.
2	Analyze and understand the problem and so uncover the main conflicts or contradictions. The technical conflicts are then identified in the TRIZ matrix 2 and listed. Find the functional analogy in biology (look into the PRIZM) or go to the biological conflict matrix (Biomimetic TRIZ).
3	Compare the solutions recommended by biology and TRIZ. Find the common solutions for biological and engineering fields. List the technical and biological principles thus recommended.
4	Based on these common solutions, build a bridge from natural to technical design. To make the technical and biological systems compatible, make a list of their general recommended compositions.
5	To create a completely new technology, add to the basic TRIZ principles some pure technical or pure biological ones.

Table 10: Steps of Biomimetic TRIZ as described by (Julian Vincent, Bogatyreva, Bogatyrev, Bowyer, & Pahl, 2006).

A full description of classic and Biomimetic TRIZ is beyond the scope of this research. The short lists of steps shown in Table 4 and Table 10 somewhat mask the complexity of these methodologies. The structured tools and procedures of TRIZ are powerful, but often perceived as overwhelming to a beginner. This may be due to its relatively recent introduction to the Western world. Additional information on these methodologies can be found in (Bogatyrev, Pahl, & Vincent, 2002; Changquing, Zezheng, & Fei, 2005; Domb, 1997; Fullbright, 2004; Lerner, 1991; Loebmann, 2002; Mann, 2004; Nakagawa, 2005; Rea, 1999, 2001a, 2001b; Salamatov, 2005; Tate & Domb, 1997; TRIZJ, unknown; Julian Vincent, Bogatyreva, Bogatyrev, Bowyer, & Pahl, 2006; Julian Vincent & Mann, 2000, 2002).

4.4.5 *Comparison of Biomimetic Methods*

Comparison of Biomimetic Methods					
	1. Bionic Association	2. Bio-Design Approach	3. Biomimicry Design Process	4. Biomimicry DP Revised	5. Biomimetic TRIZ
			1. "Identify" -- Develop a Design Brief of the Human need.	1. "Distill" the design function.	1. Define the problem in the most general, yet precise way. Avoid limiting terminology or thoughts. Then list the desirable and undesirable properties and functions.
			2. "Translate" -- Biologize the question; ask the Design Brief from Nature's perspective.	2. "Translate" -- Biologize the question; ask the Design Brief from Nature's perspective.	2. Analyze and understand the problem and so uncover the main conflicts or contradictions. The technical conflicts are then identified in the TRIZ matrix 2 and listed. Find the functional analogy in biology (look into the PRIZM) or go to the biological conflict matrix (BioTRIZ).
	1. Observe the organisms' behavior carefully. Take the phenomena of the organisms as the association objects.	1. Select features of a living organism that exceed current technological capabilities.	3. "Observe" -- Look for the champions in nature who answer/ resolve your challenges.	3. "Discover" – Use all available means to look for the champions in nature who answer/resolve your challenges.	3. Compare the solutions recommended by biology and TRIZ. Find the common solutions for biological and engineering fields. List the technical and biological principles thus recommended.
	2. Analyze the mechanism of the phenomena of the organisms system.	2. Derive principles and processes responsible for their superiority.	4. "Abstract" -- Find the repeating patterns and processes within nature that achieve success.	4. "Emulate" – Emulate natures strategies based on multiple solutions, expert knowledge, and observations.	4. Based on these common solutions, build a bridge from natural to technical design. To make the technical and biological systems compatible, make a list of their general recommended compositions.
	3. Analyze the practical problem. Develop a bionic idea into a problem-solving method or product.	3. Develop models and methods to describe biological systems in terms useful to designers.	5. "Apply" -- Develop ideas and solutions based on the natural models.		5. To create a completely new technology, add to the basic TRIZ principles some pure technical or pure biological ones.
		4. Demonstrate the feasibility of translating this knowledge into dependable and efficient hardware.	6. "Evaluate" -- How do your ideas compare to the successful principles of nature?	5. "Evaluate" -- How do your ideas compare to the successful principles of nature?	
			7. Repeat	6. Repeat	

5 CURRENT INTERDISCIPLINARY AND BIOMIMETIC COMPUTER SCIENCE

This chapter will present the various ways computer science has been involved in crossing disciplinary boundaries in the context of the Chapters 3 & 4.

5.1 Types of Disciplinary Crossings in Computer Science

Interdisciplinary computer science abounds today. This section will describe the various means this has taken place using the terms identified in Section 3.4.

5.1.1 Crossdisciplinary Computer Science

As described in Section 3.4.1, crossdisciplinarity is a way of describing or analyzing an aspect of a particular field through the lens of another field. There are many examples where computer science has been analyzed in this way. Three examples were presented at the InSITE 2004 conference on Information Science and IT Education. First, Lenarcic performed an historical overview of "software psychology" in (Lenarcic, 2004), which discussed the various attempts that have been made to analyze computer programming from the perspective of psychology. Second, Roussev examines software development from an Information Sciences point of view in (Roussev & Roussev, 2004). Third, Michalec introduces a novel look at computer science from a perspective of the arts in (Michalec & Banks, 2004) – specifically by comparing information systems development methodologies with the development of Jazz music. These are just a few examples of how new perspectives have enriched the field of computer science through the use of crossdisciplinarity.

5.1.2 Multidisciplinary Computer Science

As described in Section 3.4.2, multidisciplinarity is side-by-side approach to disciplinary activities. From an educational perspective, most university level computer science programs are multidisciplinary in nature. They require a student to take a balanced curriculum of undergraduate courses in science, mathematics, humanities, social sciences, etc. Traditionally, these programs do not attempt to integrate these subjects. Rather, a student is left to draw any connection between the topics on their own. From a development perspective, there are many examples of multidisciplinary activities. Almost any new commercial development project that incorporates software technology could be considered a multidisciplinary effort. For example, a new electronic widget being developed would require a project team that includes members from marketing, engineering, operations, and software development to deliver an end product.

5.1.3 Interdisciplinary Computer Science

Interdisciplinary computer science is a broad area of interest. At the time of this writing, a simple web search of the words "interdisciplinary computer science" resulted in over 1.1 million results. Arguably, computer science has a role in nearly every discipline ((Grasso, 2003)). One can consider the role of computer science in the realms of interdisciplinarity knowledge, research, education, and theory which were identified by Nissani in (Nissani, 1995) and described in Section 3.4.3. Furthermore, the application of these realms can be characterized by Nicolescu's degrees of interdisciplinarity ((Nicolescu)) which were also described in this section. One could furthermore characterize the various efforts according to Nicolescu's degrees of interdisciplinarity.

In the realm of education, universities are increasingly offering interdisciplinary programs where students can choose a curriculum that integrates classes and research from

computer science and numerous other fields. Graduates of such programs or people with equivalent experience are in great demand. They can be subject matter experts in a specific area, and then use their computer science knowledge to manage or develop software solutions that can be applied to these specialized areas. An example for the degree of application would be using genetic algorithms for optimization problems. Bioinformatics, human factors application development, and computational biology would all be examples of the degree of generation of new disciplines.

5.1.4 Transdisciplinary Computer Science

Broadly speaking, technology introduces many issues into a society that can be considered transdisciplinary. For example, understanding the role of Internet in society and its impact on individuals, groups, societies, economies, terrorism, military, politics and more is certainly a transdisciplinary effort. But even more specific computer science based issues can be considered transdisciplinary. In (Salazar, 2006), Salazar attempts to provide a transdisciplinary perspective on cyber worlds. In it, he examines perspectives from computer science and engineering as well as social science research regarding the psychological, social and cultural aspects of cyber worlds. It is clear that transdisciplinary approaches to research and problems will become increasingly more important as our society relies more and more on computing and networking technologies for nearly every facet of its being.

5.2 Tools for Disciplinary Crossings in Computer Science

5.2.1 TRIZ for Software

Over the last nine years, there has been a quiet thread of research that is gradually attempting to apply the TRIZ innovation method to software development. The first attempt to do so was made by Rea in (Rea, 1999). His initial work in August 1999 for the

TRIZ Journal attempted to address the problem of synchronization in programming concurrency. In this work, he performed a TRIZ analysis of the "Roller Coaster Problem". Using the TRIZ process, he was able to identify "TRIZ Inventive Principle #24 – Mediator" to solve the problem. The Mediator concept is similar to a "Monitor" in computer science that provides exclusive access to critical sections of code. Thus, in this example he was able to use a TRIZ process to arrive at a known solution to the problem of concurrency. The novelty of course, is that he used a problem solving system that had never been applied to computer science before.

By 2001, Rea proposed software analogies for 34 of the 40 Inventive Principles from classic TRIZ. His work in (Rea, 2001b) and (Rea, 2001a) was intended to accelerate the application of TRIZ to software. In them, he draws parallels between the "physical" world and the "virtual" world of software. The remaining 6 analogies were later developed by Fulbright in 2004 in (Fullbright, 2004). Fulbright also summarized the complete list into a two tables, which have been reproduced in Figure 11 and

Figure 12. In these figures, the original TRIZ inventive principles are shown in the left-hand columns and the software analogy are shown in the right-hand columns.

It is worth noting here that in (Rea, 2001b), Rea claims that his continued application of TRIZ to software led him to generate and submit 13 patent applications (The status of these patents could not be determined at the time of this writing).

1. Segmentation a. Dividing an object into independent parts. b. Make an object modular. c. Increase the degree of fragmentation.	 Intelligent Agents C++ templates Confidential Objects
2. Extraction Separate interfering or necessary parts	 Extraction of text in images
3. Local Quality a. Change structure from uniform to non-uniform b. Make parts perform different functions	 Non-uniform access algorithms Higher levels in a single index tree
4. Asymmetry Change from symmetrical to asymmetrical.	 Load balancing, resource allocation
5. Consolidation Make operations contiguous or parallel	 Threading, multitasking
6. Universality Perform multiple functions; eliminate parts	 Personalization of user interface
7. Nesting Place an object into another	 Classes within other classes
8. Counterweight Counter weight with lift	 Shared objects in multiple contexts
9. Prior counteraction Preload compensating counter tension	 Pre-processing
10. Prior action Fully or partially act before necessary	 Pre-compiling
11. Cushion in advance Prepare beforehand to compensate low reliability	 Fair scheduling in packet networks
12. Equipotentiality In a potential field, limit position changes	 A transparent persistent object store
13. Do it in reverse Invert actions	 Transaction rollback
14. Spheroidality Replace linear parts with curved parts	 Circular abstract data structures
15. Dynamicity Find an optimal operating condition	 Dynamic Linked Libraries (DLLs)
16. Partial or excessive action Use "slightly less" or "slightly more"	 Perturbation analysis
17. Transition into new dimension Move in more dimensions	 Multi-layered assembly of classes
18. Mechanical Vibration Oscillation	 Change the rate of an algorithm
19. Periodic Action Periodic or pulsating actions	 Scheduling algorithms
20. Continuity of useful action a. Continue actions b. Eliminate all idle or intermittent actions	 Utilizing processor at full load Eliminatel blocking processes

Figure 11: List of Software Analogies for 1-20 TRIZ Inventive Principles from (Fullbright, 2004).

21. Rushing through Conduct a process at high speed	Burst-mode transmission
22. Convert harm into benefit Eliminate the primary harmful action	Bottleneck DDOS zombies
23. Feedback Introduce feedback	Feedback improving iterations
24. Mediator Use an intermediary	XML-based view generation
25. Self-service Performing auxiliary functions	Periodic auto-update
26. Copying Use simpler and inexpensive copies	Perform a shallow copy
27. Dispose Use multiple inexpensive objects	Rapid prototyping
28. Replacement of Mechanical System Replace mechanical means	Voice recognition/dictation
29. Pneumatic or hydraulic construction Use inflatable gas or liquid parts	Dynamically allocated data structures
30. Flexible films or thin membranes Isolate the object from the environment	Wrapper objects
31. Porous materials Make an object porous	Intelligent tutoring systems
32. Changing the color Change the external view (transparency)	Transparency layers
33. Homogeneity Use same material	Container objects
34. Rejecting and regenerating parts a. Discard portions of an object b. Restore consumable parts	Garbage collection Transaction rollback
35. Transformation properties Change the degree or flexibility	Multi-role objects
36. Phase transition Phase transition phenomenon	Emergent behavior
37. Thermal expansion Use thermal expansion or contraction	System memory
38. Accelerated oxidation Use oxygen-enriched air	Salted encryption
39. Inert Environment Replace normal environment with an inert one	Test harness
40. Composite materials Use composite (multiple) materials	Composite objects

Figure 12: List of Software Analogies for 21-40 TRIZ Inventive Principles from (Fullbright, 2004).

Rea's work in (Rea, 2002) continued to explore means of applying TRIZ to software. In this journal article, he discusses the need for a structured innovation method in computer science to address complexity and to bridge the widening gaps between computing areas of focus in academia and in practice. He asserts that although the application of TRIZ to software is in its infancy it has the potential to become that structured innovation method. To facilitate this, his work in (Rea, 2002) is the enhancement of the TRIZ S-Field tool for use with software.

Shortly after Fulbright completed the software analogies for the TRIZ Inventive Principles in 2004, Darrell Mann published an article on TRIZ for software in *TRIZ Journal*. The article (Mann, 2004) indicates that there had been some opposition to the idea of applying TRIZ to software. The two main arguments against it are that software development is an immature process that is more of an art than a science, and that the 40 principles did not apply to software. Mann's research dismissed those arguments through his analysis of 40,000 software patents that validated the inventive principles for software. His research also included the adaptation and application of additional TRIZ tools. Ultimately, he identified seven TRIZ tools that showed the most promise which he categorized as either problem definition or solution generation tools. The problem definition tools are (1) Ideal Final Result, (2) Problem Explorer, (3) Subversion Analysis, and (4) Contradiction Matrix. The solution generation tools are (1) Inventive Principles, (2) Trends/Evolution Potential, and (3) 'Self-X'. A full description of these tools and how they relate to software are expected in Mann's pending book titled "TRIZ for Software" which appears to be due for release in early to mid 2008.

5.2.2 TRIZ for Software Process Improvement

In 2002, Stanbrook documented an attempt to apply TRIZ to the software development process using a software tool called TechOptimizer. He describes this work in (Stanbrook, 2002). In it, he used the TechOptimizer to analyze a typical waterfall development process using a TRIZ-based process analysis. During this exercise, several suggestions were brought out such as the elimination of inspections and testing in favor of defect prevention methods. This interesting validation for agile development is a source of future research.

5.3 Disciplinary Crossing Computer Science

Computer science has already benefited from the experiences and contributions of people with expertise in fields outside of traditional computer science. Software design patterns and human-computer interactions are two examples that will be briefly introduced.

5.3.1 Software Design Patterns and APIs

Christopher Alexander originally developed the idea of "design patterns" in his 1977 book titled *A Pattern Language*. Surprisingly, he was a building architect, not a software architect. His original work proposed an organized set of recurring problems and solutions in the architectural building field. The book contained approximately 250 patterns of solutions that were known to work. This set of patterns was called a "pattern language". A pattern is comparable to a word in a spoken language. The words stay the same, but they can be combined in different ways to make a sentence.

In the late 1980s, Kent Beck and Ward Cunningham began applying the concept of a design pattern to object-oriented software. They presented their work on (Beck & Cunningham, 1987) in 1987 at the Object-Oriented Programming, Systems, Languages & Applications (OOPSLA-87) conference sponsored by the Association for Computing

Machinery (ACM). This seminal work proposed five initial object-oriented software design patterns. Work continued on the development of design patterns and in 1994 the now famous Gang-Of-Four (Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides) published their book titled "Design Patterns: Elements of Reusable Object-Oriented Software" ((Gamma, Helm, Johnson, & Vlissides, 1995)) which began a wide movement to develop and use software design patterns and pattern languages.

Despite its moderate popularity in the following decade, software design patterns have not yet achieved ubiquity. One criticism of design patterns is that they do not actually provide functional code which can be reused. Rather, they are design abstractions that must be implemented (or re-implemented) for each new application. This is arguably less desirable than software APIs that provide executable functions that can be readily leveraged by an application. To address this issue, some work has been done by Meijer in (Meyer & Arnout, 2006) to componentized (i.e. develop APIs) software design patterns for direct use in applications.

5.3.2 Human-Computer Interactions

Human-computer interaction (HCI) is defined by (Hewett et al., 1992) of the ACM as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them." HCI is an interdisciplinary field which was spawned from areas of computer science such as computer graphics, operating systems, human factors, ergonomics, industrial engineering, cognitive psychology, and computing systems. It has grown dramatically and continues to evolve. There are many aspects of HCI that can be explored, but are beyond the scope of this research. For further reading, please see (Hewett et al., 1992).

An interdisciplinary field called Human-computer Interaction (HCI) has emerged to study relationship between humans and computers. It is defined by (Hewett et al., 1992) of the ACM as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them." HCI is an interdisciplinary field which was spawned from areas of computer science such as computer graphics, operating systems, human factors, ergonomics, industrial engineering, cognitive psychology, and computing systems. It has grown dramatically and continues to evolve. There are many aspects of HCI that can be explored, but are beyond the scope of this research. Many new technologies are emerging to enhance human-computer interactions. Figure 13 shows the Gartner Hype Cycle for Human-Computer Interactions for 2006. For further reading, please see (Hewett et al., 1992).

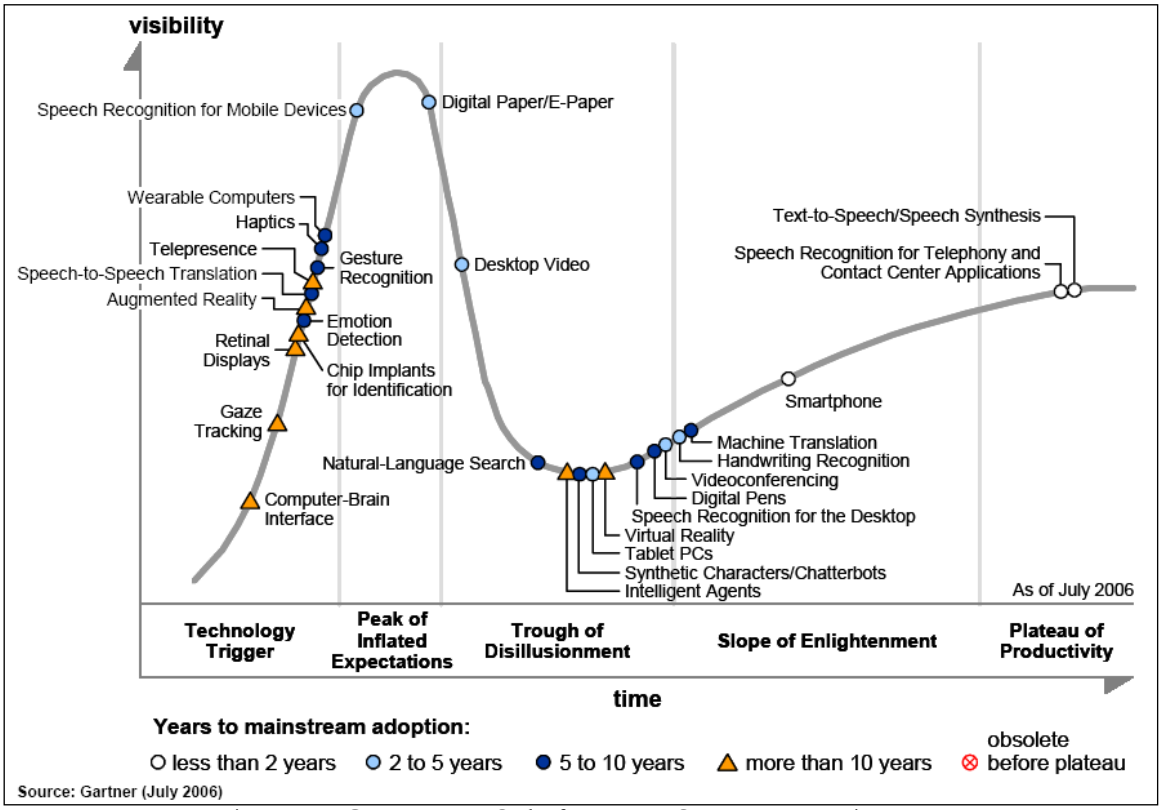

Figure 13: Gartner Hype Cycle for Human-Computer Interaction 2006 (Fenn & al, 2006b). Figure reprinted with permission from copyright owner.

5.4 Biologically Inspired Computer Science

There are already many examples of computer scientists taking inspiration from nature to achieve certain objectives. Some of these objectives include fault tolerance, automation, optimization, and artificial life. By recognizing the inspiration that nature has already brought to the field of computer science, one can see even more potential for taking inspiration from it in the future. Nature's designs are elegant and have been proven successful. Figure 14 is a simple mind map which illustrates some of the many areas of computer science which have already been inspired from nature's successful designs. More specific examples can be found in (Olariu & Zomaya, 2006).

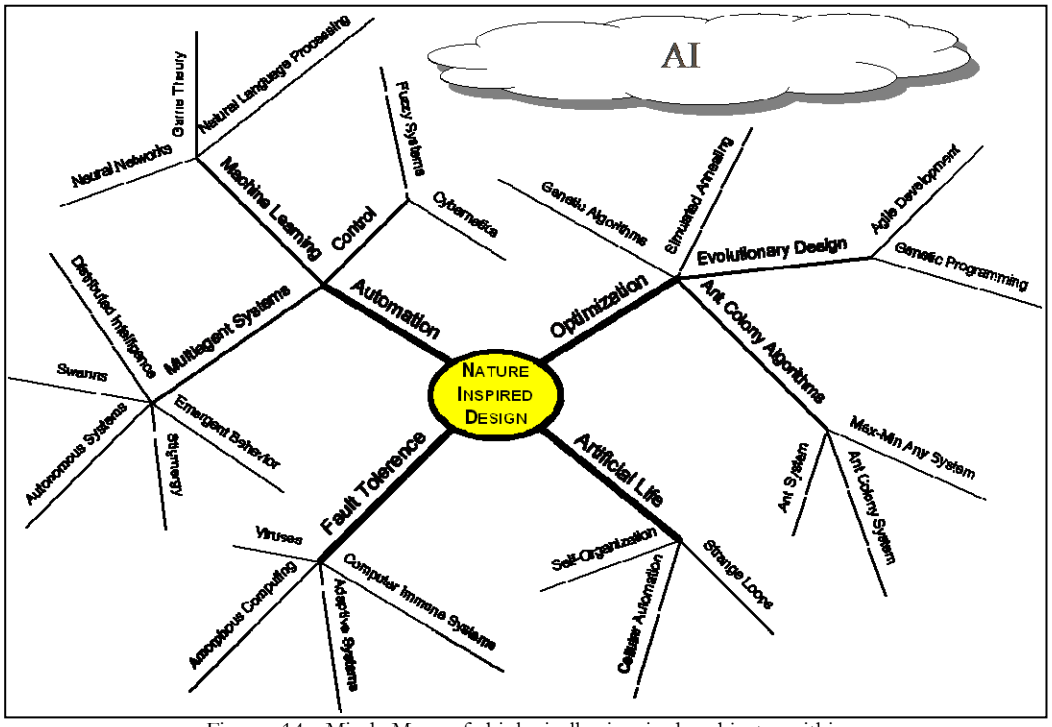

Figure 14: Mind Map of biologically inspired subjects within Computer Science. (Korecki)

5.4.1 Evolutionary Computation

Larry Fogel is credited with developing the concept of evolutionary computing in 1966 ((De Jong, 2008)). He was influenced by Darwin's theory and early work toward computational intelligence like (Friedberg, 1958). Evolutionary computing (i.e. genetic programming) and genetic algorithms attempt to solve complex problems by introducing variation, selection, and heredity to populations of solutions and then evolving them so that desirable characteristics can be carried forward over time. In the case of genetic algorithms, a problem is defined in mathematical terms which can be manipulated to find a best case scenario solution. In the case of genetic programming, the population consists of computer programs that are recombined into new generations of software versions. This "evolution" should lead to more complex, efficient, and functional behavior. Evolutionary

Computation is typically categorized as a method of combinatorial optimization, which is an algorithmic approach to problem solving.

The basic idea of combinatorial optimization is that a problem is defined in mathematical terms so that it can be manipulated in an iterative fashion to find a best case scenario solution. Optimization techniques generally require three basic components. (1) A representation or format potential solutions to a problem. This representation ultimately defines the search space of all possible solutions. (2) A mathematical expression of an objective to be achieved. (3) An evaluation function which maps the search space of possible solutions to a set of numbers. Each solution is assigned a numeric value that indicates its quality. These three components allow you to clearly define a solution set and understand how one solution compares to another. The role of optimization techniques then, is to search the set of all possible solutions for the best (maximized or minimized evaluation) solution without necessarily evaluating every possible solution. To accomplish this, it is often the case where a random solution is selected and evaluated. Once this has been done, a set of solutions that are very similar to the first are evaluated. Solutions that are similar are considered to be in the same "neighborhood". The best solution in a neighborhood would be considered a "local ~~optimum"~~. The best solution in the entire set would be considered a "global optimum". There is often no way of knowing if an individual local optimum is also a global optimum.

Chapter 6

6 A FRAMEWORK FOR SOFTWARE INNOVATION

As established in Chapter 2, the computing systems and the data they contain have increased in complexity to the point that we must seek alternative methods of developing, managing, and interacting with them. These issues can only be approached holistically. A purely one-dimensional technical approach will only exacerbate the problem. In order to take a holistic approach we must introduce more intellectual diversity to the field of computer science through interdisciplinary education and collaboration (Chapter 3). Only through the collaboration and intermixing of diverse perspectives can transcendent problems be solved. To facilitate this interdisciplinary cooperation some specific tools can be used including TRIZ and Biomimicry. TRIZ brings a structured approach to problem solving and a wide breadth of proven solution techniques. Biomimicry brings an interdisciplinary perspective and opens a door (perhaps both figuratively and literally) to a world of innovative designs and solutions. The real challenge for computer science is to integrate these concepts into its core foundations so that a new culture of collaboration and creative problem solving can be fostered.

6.1 Interdisciplinary Participation and Education

As described in sections 3.1, 3.2, and 3.3 the finite nature of the human mind requires individuals to specialize their knowledge. The down side of this specialization is that the continuous nature of reality and unified knowledge becomes segmented and compartmentalized. This compartmentalization causes deep silos of understanding that prevent the flow of knowledge and undermine unity.

However, it is the hypothesis of this research that this compartmentalization can be overcome or at least minimized through interdisciplinary collaboration. Not by simply "throwing two cats in a bag", but by deliberately soliciting participation from specialists with diverse fields of influence. This participation can be practiced in academic environments, research environments, and even commercial environments. Essentially, anywhere creative problem solving is required.

6.1.1 Importance of Interdisciplinary Education

Computer Science is a young discipline. Modern computers and software have only been around for approximately 60 years. Although it has a foundation in mathematics and engineering, it has in many ways departed from its origin. It is a field constrained more by human intellect than by physical limitations of hardware. As such, one will recognize the importance of drawing on all forms of human knowledge to enhance the creative application of computing systems. This human knowledge is distributed across all of the academic and applied disciplines (and indeed across all humanity).

As described in Section 3.6.1, Jack Steel (originator of Bionics) recognized the importance of interdisciplinary knowledge. An individual educated in more than none discipline is positioned to play the essential role of "translator". As translator, they may have a fluency in two distinct disciplinary languages and paradigms. They are uniquely placed at a crossroads where they can frame problems to their monodisciplinary peers on either side. They can therefore become an essential link in the process for drawing out creative solutions that can transfer between the disciplines. Considering the relative youth of computer science, one can expect that a great deal of the knowledge transfer will flow from the more mature fields to computer science. Conversely, the advancements made in

computer science can renew and reenergize mature fields as it enables new methods of collecting, analyzing, and visualizing data (see Figure 3 and discussion in section 2.1.2).

6.1.2 Intellectual Diversity and Solution Optimization

The natural world is an excellent basis for the creation of uniqueness. In genetics, a diverse pool of genes can be combined to create unique individuals. The basis of genetic development starts out with a large pool of individual organisms in a population. The genes of individuals are recombined for the creation of a new generation of that population. As a result, diverse and unique individuals are produced in each consecutive generation. Although some randomness occurs, the diversity of each generation is limited by the genetic material that was found in the first generation. This genetic model has been recognized as a powerful optimization technique for mathematics and computer science. Genetic algorithms have been developed to identify optimized individual solutions from populations of solutions that are too numerous to evaluate comprehensively (see section 5.4.1).

With this powerful process in mind, consider the development of new innovations as an optimization problem. This research anticipates that the genetic model also applies to the development of new ideas. That is, intellectual diversity can be recombined for the creation of innovative solutions. The diversity and uniqueness of the solutions created in a development effort is only limited by the intellectual capital that has been put into that effort.

This paradigm can be applied to software development. Consider for a moment that a software developer is capable of developing or "evolving" a software solution out of his own knowledge and experience. Given a problem domain for a new application, a developer will draw on his or her expertise to develop and optimize the most suitable

solution they can think of to meet the requirements. Put another way, the developer recombines their population of ideas to create a new generation of ideas that can form the basis of a solution. The optimization is done through individual judgment. It is obvious that the developer is unable to produce solutions whose elements are not within their basis of knowledge.

The population of ideas for a solution is multiplied when a developer is part of a software development team. Each member of the team can draw on their respective expertise and experience to propose more ideas. Through collaboration, these ideas can be recombined to create an optimized group solution. If the collaboration is successful, this optimized solution will be better than any of the solutions that an individual would have developed in isolation. In other words, it is more likely to approach global maxima.

For a creative process, diverse input enhances the likelihood of developing creative and unique output. It is anticipated that intellectual diversity will enhance the likelihood of producing unique and creative solutions. This paradigm forms an analogous case to increase the intellectual diversity of development teams.

This paradigm also lends itself to iterative software development processes. Iterative development is a process which produces new generations of solutions on a relatively rapid rate. The purpose is to gradually evolve solutions toward ideality. The aspiration of iterative development is to realize the principle that was stated by Genrich Altshuller in his TRIZ law of increasing ideality (see section 3.7.2). It states that any technical system will evolve in such a way as to increase benefit, reduce cost, and reduce harm. In other words, it will continue to get better – not worse! With iterative

development, the more rapidly each new generation is produced the more rapidly the system will improve.

As an aside, this concept is common between technological and biological systems. In nature, if a system (or species) is not adapting to increased ideality to survive in its environment, then it is likely to become extinct. So it is in technology, where a system becomes obsolete if it does not adapted and improved with time.

6.2 Knowledge Transfer and Discovery

Understanding the importance of interdisciplinary knowledge and collaboration is one thing, but practically implementing it is another. The segmentation of modern disciplines (See section 3.1 and 3.3) has created communication barriers that are difficult to overcome. Tools are necessary to facilitate this communication.

6.2.1 Finding a Common Language

A common language is essential for the transfer of ideas. Computer science itself has arguably struggled to some degree with the concept of a common language even within its own disciplinary boundaries. Rapidly changing technologies and software development platforms have fragmented practitioners (See section 2.1.1) into platform-centric silos. However, there have been some significant efforts to overcome this internal fragmentation. Software engineering principles have been developed which abstract concepts through technology independent approaches to requirements analysis and design. Modeling tools such as UML offer a means for a more general problem solving approach. Finally, software pattern languages have been developed to define common building blocks that facilitate solution reuse.

The problem of exchanging knowledge between computer science and other disciplines is even more difficult. Differing technical languages and thought are even

further removed when crossing disciplines. Once again, this can be mitigated by interdisciplinary individuals acting as translators; however, these individuals are not always accessible. In this case, tools from TRIZ (see section 3.7.1, 3.7.2, and 4.4.4) and the Biomimicry Database (see section 4.2.4) can be used to facilitate the transfer of knowledge across the disciplines. In the case of TRIZ, common structured processes can be used to abstract problems, and then relate them to abstract solutions which cross disciplinary domains. This layer of abstraction can act as a common ground for understanding between multidisciplinary teams. In the case of the Biomimicry database, common problems are documented in the "native language" of two or more disciplines (specifically, engineering and biology) thus acting as a sort of "Rosetta Stone" between them. Both of these approaches can be applied to computer science.

6.2.2 Exchanging Language

Another opportunity for knowledge transfer and discovery is for computer scientists to leverage some of the language tools, organizations, and nomenclatures from other disciplinary fields. For example, biology is particularly good at developing and organizing taxonomies of entities (organisms). Furthermore, biology is particularly interested in understanding the context (ecological environment) in which those entities exist. Biologists use the term "biome" to describe that context. A biome is a major regional or global biotic community, such as a grassland or desert, characterized chiefly by the dominant forms of plant life and the prevailing climate.

Computer scientists could share these constructs to better understand and organize their own domain of knowledge. They too are concerned with the understanding of entities and context. However, in this case the entities of interest are software, data, and processes. The context of concern is the technical, not ecological environment. For example, a Linux,

Apache, MySQL, PHP (LAMP) environment could be considered a computing biome in which dynamic web pages exist.

Conversely, it is also conceivable that some of the many software modeling tools such as UML (among others) may be able to enrich the language and organization of other disciplines. UML is an excellent tool for communicating the design and architecture of systems at a high level. It can represent functional models, structural models, and dynamic behavioral models in an explicit yet concise manner without getting weighed down in too much detail. UML has already been leveraged outside of the software development field to some degree. It has been used for business process modeling, systems engineering modeling, and organizational modeling. It is not hard to conceive how this tool could be applied more broadly.

6.2.3 Finding Common Solutions

As described in section 5.3.1, pattern languages are collections of successful software design solutions. Regardless of whether a successful software design solution is captured as a pattern in a pattern language or componentized as an API, there is still the issue of visibility. That is, software developers may or may not know about the existence of a solution or where to find it. For design patterns, there are many pattern languages which have been developed including the one produced by the Gang-Of-Four. These pattern languages are sometimes proprietary and other times public. They may be published online or in books, but there is no single source that unites them. It is possible that this problem could be addressed with tools that assist in the problem definition and discovery of known solutions – tools like those found in TRIZ.

TRIZ shows great promise for transferring problem definitions and solutions across the disciplines (see section 5.2.1 and 5.2.2). It is an established tool that has already been

used to map solutions across domains. It has been applied to engineering and process development and draws knowledge from solutions that cross many disciplines. Recent work to extend TRIZ to include biological and natural solutions is enriching the tool even further. Further extending TRIZ to software will make it an even more powerful tool going forward. Not only will this work benefit computer science, it will also enhance TRIZ and the other disciplines that use it.

There appear to be several areas of software development where TRIZ could be applied. First, TRIZ could be used to identify and locate software design patterns. Software pattern languages are just another source of known good solutions. These pattern languages could be analyzed for their inventiveness principles much like patents were used to develop the 40 Inventive Principles of TRIZ. There has already been work as described in section 5.2.1 which attempted to draw software analogies for the 40 principles. Additionally, Mann ((Mann, 2004)) started to analyze software patents, but there are many software solutions that have been defined for software pattern languages that were not analyzed. By expanding the breadth of solutions that get fed into TRIZ, it may be possible to develop a means for defining common problems in software, then mapping them to known principles used to derive solutions.

It is suspected that TRIZ would be relatively easy for computer scientists and practitioners to adopt. Generally speaking, they are already proficient in techniques that model problems and solutions. They are used to working with conceptual abstractions and algorithms. It follows that individuals and organizations in the field of software should find a natural aptitude for working with TRIZ processes and tools.

6.2.4 Harnessing Serendipity and Systems of Innovation

There is a classic debate amongst innovators over the idea that a systematic formula can be used for generating new ideas. Some argue that they simply occur spontaneously for creative people. However, one must consider that our knowledge, experiences, and interactions with each other and the world are the key elements that trigger creative thought processes and ideas. Ideas are not created in a vacuum. They come about either consciously or unconsciously as we continuously learn and form conceptual links between diverse ideas. It is these conceptual (and neurological) links that facilitate those "Ah-ha!" moments of understanding and creativity.

Genrich Altshuller (see section 3.7.1) defined a truly new (inventive) idea as one that utilized effects outside of the disciplinary field where that idea was being applied (see his third principle in section 3.7.1). He arrived at this principle after evaluating over 2 million patents, carefully identifying those that were the most inventive. He perceived that inventive ideas were not simply enhancements to something, but rather radically new ideas that uniquely connected previously unrelated concepts. One may conclude then that truly new ideas can be generated from the transfer of ideas from one field to another. But how does this occur?

Sometimes new ideas come about rather mysteriously. This idea may just occur to an individual at an opportune moment though no specific effort was made to conceive it. This type of occurrence can be described as chance or serendipity. Serendipity is defined by the American Heritage Dictionary ([29]) as "the faculty of making fortunate discoveries by accident." A goal of many innovative organizations is to be "open to serendipity". Though they cannot explain it, they attempt to foster an environment that capitalizes on chance ideas as they occur.

A more intentional approach to generating new ideas is through trial and error. In (Salamatov, 2005), Salmatov asserts that the "trial and error" method is by far the most common approach to invention. He uses Thomas Edison's countless attempts to identify an adequate filament for the incandescent light bulb as a prime example of blind trial and error. Edison's research conducted over 6000 experiments before he discovered that a filament of charred bamboo (constructed from a Japanese fan-case which Edison borrowed from a lady at a ball) was able to burn for approximately 1200 hours. Salmatov further asserts that most scientific approaches to innovation are simply a method of reducing the search space for trial and error innovation. Sometimes accidental discoveries come about as a side-effect of a focused effort toward another goal.

Another means of innovation is to bring together a multidisciplinary team of individuals to develop a solution to a problem. The so-called "skunk-works" model of innovation made famous by Lockheed Martin for the advanced development of military aircraft. This approach was notable for bringing together interdisciplinary individuals to focus on a problem without getting bogged down in bureaucracy. Group techniques and brainstorming relate to this approach of innovation. A key benefit of group creativity and brainstorming is the back-and-forth development of ideas.

Altshuller's work on TRIZ has been a successful attempt to move beyond the mystery of trial and error and serendipity to form a more structured and repeatable process for invention. It provides a rigorous process which decomposes a problem for analysis, and then uses structured methods of identifying principles that can be applied in a solution (see section 3.7.1 and 3.7.2).

It appears that all four methods of innovation: serendipity, trial and error, skunk works, and TRIZ can all be successful for the generation of new ideas and innovations that are truly unique. It is a hypothesis of this research that all four methods can work together synergistically. The likelihood of creating innovative solutions can come about by forming multidisciplinary teams to perform both freeform and structured activities. Furthermore, the interactions of such teams during those freeform and structured activities will actually enhance the chances that serendipitous inventions will occur. Serendipity can be facilitated by encouraging interactions between people with diverse perspectives.

6.3 Nature as a Product Model

Software development is the science of defining the behavior of machines. Great effort is taken to explicitly describe every step to be taken to accomplish a task. Ultimately, a computer scientist is trying to duplicate his or her own implicit knowledge of a process onto a machine that can perform a task in his or her stead. The machine can in turn perform that task tirelessly and repetitively without complaint.

Software is an incomplete implementation of a behavior conceived in the mind of a developer. It is impossible for a developer to account for every situation or runtime event when designing an application. Runtime errors occur that cannot be anticipated and are therefore not handled. In this situation, the application fails. However, if that error or unexpected event was presented to the developer of the application (or in some cases the end-user of the application), that person could select an appropriate way to handle it. It is in this example that one can see that software tries to mimic human intellect, but ultimately falls short.

If software is an incomplete implementation of a human behavior, then one must look at the source of that behavior. How does mankind conceive of processes and

solutions? Is the human brain the source of all knowledge, or does itself take lessons from a greater source? Knowledge is by definition "the sum or range of what has been perceived, discovered, or learned (Dictionary, 2004)." By this definition, one must consider the source of mankind's learning – the natural world. .

The natural world is a source of great knowledge from which mankind has derived its own. It is therefore reasonable to believe that much of our knowledge of processes and solutions have been derived from nature – either consciously or unconsciously. The remaining sections of this chapter will examine the benefits for computer scientists to examine nature as a source of inspiration and some of patterns found in the natural world are being or could be leveraged by computer scientists.

6.3.1 Biomimetic Software Designs and Patterns Languages

There are many examples of software that are based on biologically inspired solutions (See section 5.4). Evolutionary computation is one specific example that illustrates the value of seeking nature's designs. Based on this topic alone, practitioners and researchers can make a case for computer scientists to dive deeper into biology. In (Lones & Tyrrell, 2001), Lones makes two key points for this. First, evolutionary computation is already a useful application of biology to computer science. Second, biology and computer science are both executable instructions that act on dynamic systems. These observations allude to the potential for biologically inspired software.

There is an immense "database" of nature's solutions that are harvestable by computer scientists. These solutions constitute a spectrum of biomimicry that can be leveraged in software problem solving and design. This spectrum of inspiration spans from very specific to very broad. At one end of the spectrum, software can mimic a specific aspect of a single organism or set of organisms (ex. Ant Colony Optimization). At the

other end of the spectrum, it can mimic a re-occurring pattern or principle that is found in many different types of organisms in different environments (ex. Swarms). Arguably, these can all be considered design patterns in nature.

In a pure sense, a software design pattern is just a way of documenting a known successful solution to common problems. For software, there is typically a "rule of three" for design patterns. That is, for a software design to be considered a pattern it must be found in at least three real world solutions. It is however, possible to consider that software design patterns could be mined from nature's database of solutions. That is, rather than searching source code for a pattern to appear three times, one could look to nature to find a pattern that has been proven successful.

This approach works very synergistically with the use of TRIZ as a set of tools. The 40 inventive principles are effectively the same as a pattern language. In fact, they are arguably more powerful than a pattern language because they are comprehensive based on the entire population of the 3,000,000 patents used to develop them. A software pattern language could not claim to be this comprehensive. If solutions like software patents, APIs, pattern languages could be mined, it may be possible to identify new principles that do not currently fall in the list of 40. If that were the case, the process would not only benefit computer science, but also enrich TRIZ itself.

This becomes even more impressive when one considers the work being done by Julian Vincent on Biomimetic TRIZ. If TRIZ continues to expand to incorporate solutions found in nature, it would be an excellent tool to facilitate biologically inspired software designs. In this scenario, TRIZ becomes the new translator tool to migrate design solutions between computer science, engineering, biology, physics, chemistry and more.

6.3.2 Mining Some of Nature's Patterns

There are laws that govern the existence and behavior of the natural world. There are also common patterns that seem to be ubiquitous in nature. Some of these laws and patterns have already found application in the field of computer science and show promise for future developments.

6.3.2.1 Autonomy

Autonomy is a condition of independence. In the natural world, every living organism has some level of autonomy. That is, aside from Divinity there is no centralized control that sustains and coordinates everything. Similarly, the Internet consists of independent computing systems. Autonomy in computing systems can be measured by the amount of human intervention required to manage them. Different types of computing systems are designed for varying levels of autonomy. Figure 15 shows a sampling of computing systems and their relative autonomy.

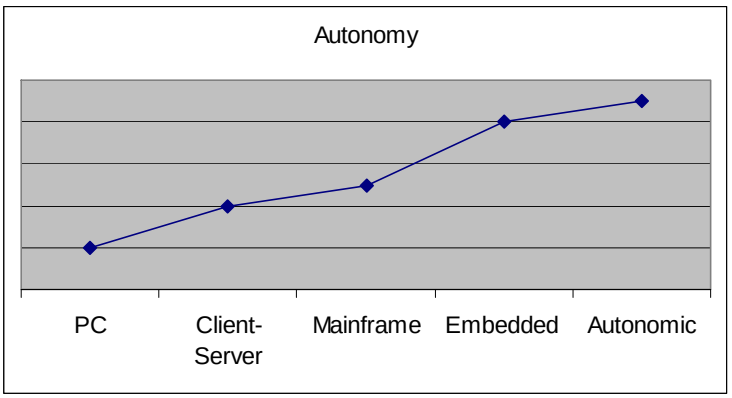

Figure 15: Computing platforms and their relative autonomy. (Korecki)

As computing systems increase in complexity and number, it will be necessary to increase their autonomy. Minimizing human intervention is also critical when considering the anticipated shortage of knowledge workers in the United States.

IBM Research has taken these notions seriously and in response began research efforts in 2001 to develop what they now call “Autonomic Computing”(Ganek & Corbi, 2003). Autonomic computing is set of integrated technologies that enable computing systems to manage themselves. They are capable of self-configuration, self-healing, self-optimization, and self-protection. These ideas in themselves are a close metaphor to nature’s organisms. Most living organisms are capable of sustaining themselves, growing, changing, and healing among other things. These capabilities are what make them autonomous. The need for autonomous systems will continue to grow as computing systems are further embedded and blended into all aspects of every day life.

6.3.2.2 Intelligence

Intelligence is the capacity to acquire and apply knowledge and reason ((Dictionary, 2004)). In the natural world, it allows organisms to make decisions that will help insure their survival. At the lowest level, even simple organisms are able to act to insure self-preservation. At the highest level of biological sophistication, the human brain is capable of complex learning, visualization, creativity, and deep levels of self-awareness.

Artificial Intelligence (AI) appears to be one of the most well established disciplines of bioinspired design. There are countless sources of information on the subject which are beyond the scope of this research. However, at a broad level, AI is an attempt to mimic the brain with computing systems. In (Hofstadter, 1979), Hofstadter contends that an image of “self” is essential for artificial intelligence. He describes a spectrum of self-awareness based on the richness of an entities image of itself. Sophisticated self-aware systems have the potential to examine themselves and change their behavior based on that image. These behaviors may even include self-modification or hereditary modifications.

6.3.2.3 Adaptation and Evolution

Adaptation and evolution are essentially gradual improvements over time. Organisms in the natural world can adapt to changing environmental conditions to insure their survival, often through heredity. In theory, evolution is a result of divergent adaptations, through generations of organisms, ultimately leading to different species. At a cellular level, these changes can take place through cloning and mutation.

In computer science, adaptation and evolution are realized at various levels. At a very basic level, object-oriented software incorporates what it calls “inheritance”. Inheritance is the ability for one class of object to inherit properties from a higher class. At a deeper level, computer scientists use the idea of adaptation and evolution as the basis for “evolutionary” computing (see section 5.4.1). This "evolution" should lead to more complex, efficient, and functional behavior ((Michalewicz & Fogel, 2004)).

6.3.2.4 Diversity

On of nature’s more provocative and self-defining attributes is its diversity. The UN Environment Programme’s Global Biodiversity Assessment (Dowdeswell, 1995) estimates that there are between 13 and 14 million species on earth, of which only 1.75 million have been documented. This diversity makes the natural world extremely robust. It is virtually impossible for a single threat to annihilate all organisms in nature. They are too diverse, and therefore are not susceptible to the same things. This concept is now being applied to modern agriculture as a way of combating disease in crops.

Computing environments should strive for a similar diversity to establish a natural defense against threats. Considering the hostile nature of malware (i.e. computer viruses, SPAM, Trojan Horses, and Botnets), the concept of diversity becomes quite relevant. If every computer ran the same operating system on the same hardware, a single hostile

program could theoretically wipe them all out. It is the responsibility of computer scientists and practitioners to provide enough diversity to prevent such a disaster.

6.3.2.5 Community

There are two forms of community in nature that can be examined as inspiration in computer science. There are homogeneous communities which consist of like members and heterogeneous communities which consist of different members. Natural examples of homogeneous communities include a swarm of bees, a herd of buffalo, or a flock of seagulls. Heterogeneous communities consist of interdependent organisms of different species living in harmony to form an ecosystem. These diverse species may have a variety of relationships with each other including symbiotic, parasitic, and predator-prey.

It is possible for community structures to be implemented for computing systems. Current examples are multi-agent systems, grid computing systems, and multi-tiered architectures. Local Area Networks are a good example of a heterogeneous community consisting of various types of PCs, servers, and peripherals. As ubiquitous Internet access becomes a reality, communities of devices will continue to form and reform.

6.3.2.6 Specialization

Specialization may be the crowning development in the natural world. It is specialization that has allowed many organisms to dominate their environments. There are two aspects of specialization to consider: social and developmental. Social specialization is an explicit form of community in which members perform specific tasks. The formation of a social community enables certain tasks to be allocated to individuals within the community, thus freeing other individuals from having to perform those tasks themselves. Economies of scale can then be leveraged. Honeybees are an example of this, as they have a division of labor defined within a hive. For example, some bees are allocated to rear the

brood while others are allocated to foraging for pollen (see section 7.1). Developmental specialization is the implicit anatomy of an organism consisting of specialized organs.

Computing systems constitute both forms of specialization. They are developmentally specialized because they consist of specialized sub-systems and software that work in concert to define a system. Computing systems also conform to social specialization. Within a community (LAN) of computing devices, certain systems are allocated as file servers, print servers, routers, PC clients, and more. Software interacts with other systems. As computing technology becomes even more embedded and ubiquitous, there will be a further increase of specialized computing systems performing dedicated functions.

6.4 Nature as a Process Model

Nature is in a continuous and unceasing state of change. Individual organisms grow and adjust. Species adapt to new conditions and environments. Nature is incredibly robust and operates as a self-regulating system. This happens through the development of individual organisms and interdependence of emergent communities of organisms.

6.4.1 Organic Development Processes

There seem to be two important developmental principles which contribute to the power and adaptability of nature. First, development happens gradually. Second, nature's designs are always functional. These principles can be illustrated with the biological example of a fetus. Even within the womb, a fetus develops functional organs very early. Within the first five weeks of development a primitive heart and circulatory system are formed and begin to function. Other organs follow as they quickly develop and begin to serve their function. Throughout the gestation these organs continue to develop and grow as they operate and sustain the life of the fetus.

These principles of gradualness and functionality can be easily mapped to software. In many ways, agile software development processes follow this approach. Specifically, feature and test based development methodologies focus on generating a functional product as early as possible, then growing and adapting it with time. New features are introduced gradually and are continually executed and exercised to verify the functionality and operation of the product.

6.4.2 Emergent Development Processes

Developmental processes not only occur for individual organisms, they also occur within communities of organisms. Nature is very distributed and decentralized. Insect societies are particularly good examples of this. For example, ants are able to perform complex tasks though our perception is that they are only capable of executing a small set of simple commands. These simple rules, when followed by an entire community facilitate the emergence of collective patterns. For example, the construction of nests is an emergent effect of a process of called *stigmergy*. There is no master intelligence that facilitates the design and construction of a nest; rather individual ants follow simple localized rules that lead to an emergent effect to produce the nest. Specifically, stigmergy is a form of indirect communication where information is passed through the environment. For the construction of a nest, an ant will transmit a chemical pheromone into a small ball of mud which it purposefully places. Subsequent ants detect this pheromone and elect to perform a similar function. As each ant contributes their ball of mud, an emergent structure starts to form until the entire nest is complete. Ants also use stigmergy for other emergent development such as optimizing the most direct route to a food source.

Arguably, open source projects could be considered to be an analog of stigmergy. An open source project is typically hosted on a public Internet website that has been started

by an individual with a particular concept. Any developer that then finds this website can elect to contribute to the open source development of the concept. If indeed a developer chooses to contribute, they would develop a portion of code and submit it to the project. This source code may then trigger other contributors to provide source code that builds on the original contribution. In this way, an open source development can evolve in many different ways. What emerges, however, is a complete product which was not necessarily guided by a single individual with a master plan.

6.5 A Framework for Software Innovation

A new framework for software innovation can be constructed from the diverse concepts contained in computer science, interdisciplinarity, and biomimicry. A number of tools can be added to a computer scientist's toolbox to facilitate the development of creative solutions that address the problems we are facing in the complexity of software and computing systems. To address this, we must focus on seeking new approaches to the development of innovative software. To do so, there are several strategies that can be used to facilitate creativity.

First, continue to foster and grow interdisciplinary computer science educational programs. This interdisciplinary knowledge will enable individuals to draw on expertise outside of computer science as well as fill the important role of translator on interdisciplinary teams.

Second, solicit diverse disciplinary specialists to participate in the software development process from both inside and outside the problem domain. The added interdisciplinary contributions will help facilitate creative solutions that surpass those of homogeneous teams of developers.

Third, leverage the structured problem definition and analysis tools of TRIZ. TRIZ provides a number of structured processes and proven tools for problem analysis, solution knowledge, analogy, and vision (see section 3.7.2). The concept of ideality and the TRIZ theory of technical evolution may facilitate a better understanding of the potential a given system has to improve.

Fourth, seek inspirational models which can be leveraged or mimicked in the solution being developed. Models can be found in diverse places even outside of computer science. A particularly strong source of these solutions is the natural world. Through the biomimicry design processes, one can seek proven solutions that can be applied to a software product. Access to these models can be facilitated through first hand experience and research, consultation and participation from experts, and tools such as Biomimetic TRIZ (see section 4.4.4).

Fifth, develop a pattern language of nature's solutions. As nature's solutions are identified through biomimicry design processes, they should be abstracted and catalogued as design patterns that can be retrieved using tools like TRIZ (see section 3.7.2) and/or the Biomimicry Database (see section 4.2.4). These contributions would also enrich TRIZ.

Sixth, leverage nature inspired development processes that gradually evolve features to form fully functional and operational source code before moving on to other features. This is a fundamental approach found in nature that validates current feature based development processes. Additionally, nature's processes are iterative and gradual. This also validates the agile focus on iterative development.

Seventh, leverage interdisciplinary efforts to increase the bandwidth of communication between disciplinary fields. Computer science is a young field that could

gain valuable learning from the experience of more mature disciplines. Conversely, other disciplinary fields may be able to learn from the specialized knowledge that has been developed within the field of computer science.

Cumulatively, these strategies have the potential to increase creativity, inventiveness, and innovation for computer science.

Chapter 7

7 A CASE STUDY ON HONEYBEE SPECIALIZATION

An initial focus of this research was to pursue the development of a new software design pattern based on the behavior of honey bees. The intent was to mimic the division of labor in a hive for task allocation on a distributed system. This work soon revealed that the honeybee model was suited for the balancing two tasks across a population. Attempts were made to scale it to N tasks, but this was problematic. However, the elegance and efficiency of the model was still compelling for binary task allocation. It was decided to continue development of the model. This is somewhat counter to a typical process where models were sought out to meet a specific set of criteria, however, the exercise still illustrated the power of interdisciplinary collaboration and the potential for nature as a model. It was difficult to find an application for the new design pattern; however, it did build an appreciation for value of collaborating across disciplines. Working with Dr. Zachary Huang at Michigan State University was a wonderful example of how there can be a synergy between computer science and another field such as biology. This chapter will present a detailed dive into implementing a honeybee simulation with a multiagent system.

As discussed in section 3.1, knowledge and science have been organized into distinct disciplines. In this paradigm, specialization is seen as a powerful problem-solving strategy. So it is in nature, when one considers the diversity of living creatures and the highly visible specializations that are present at all levels. Species, populations, and individuals all hold highly specialized niches in a broad ecosystem of life and the microcosm world in which they live. A beautiful example of specialization in nature is the honeybee. Honeybee colonies are very structured and dynamic populations where

individuals are highly specialized. However, this specialization is complemented by a sophisticated model of social interaction. In this chapter, we will drill deeply into the sophisticated regulatory interactions of honeybee specialization and the ways in which it shows the importance of social interaction. We will then bridge this discussion into a new paradigm for collaborating in interdisciplinary teams.

Advanced insect colonies such as those of honeybees have long been likened to a "super-organism" ((Huang & Robinson, 1992) and references). A colony operates much like a complex organism in itself, but it is composed of many smaller organisms. The individuals that compose the colony are relatively simple and cannot survive in isolation for extended periods of time. However, when these simple individuals form a collective whole, they are able to achieve great feats. Insect colonies are an excellent example of the whole being greater than the sum of its parts.

This paradigm of a "super-organism" has great bearing on the field of computer science. The artificial computing systems that we develop, although quite powerful in our own estimation, are countless levels of magnitude lower than nature's biological computing systems. They do not compare in complexity to DNA, in capacity to the human brain, or in ubiquity to nature. The natural world has organisms that span all levels of complexity, and computer scientists can potentially learn from each of them. The level of complexity at which an organism exists does not diminish the impact that organism can make. The smallest and simplest organisms are capable of supporting life or devastating an entire ecosystem. It is most often the case that simple organisms accomplish these great feats with sheer numbers. Multitudes of simple organisms acting on local rules, based on local

information can achieve things that the most complex organism will never achieve while acting alone.

Some scientists have proposed that organic life began with single-celled organisms. These cells eventually joined to form multi-celled organisms. These single-celled organisms joined because there was an advantage to surviving in a community. Once communities were formed, specialization became another advantageous technique for survival. Resources could be dispatched to achieve economies of scale.

It is possible that computer science is at an equivalent stage in its development. It was once a study of stand-alone machines, but is now a field of networked, clustered, and distributed systems that are in constant communication. Distributed computing systems are now enabling an exponential increase in processing power for a broad set of users. Grid systems have surpassed supercomputer speeds by dividing large jobs into manageable parts and processing them in parallel. The age of networking and community has replaced the age of isolated individuals.

If we consider an organic paradigm further, we may gain insight into the possibilities of where computing systems may evolve into the future. The possibilities are as endless as nature itself. For this reason, computer scientists must broadly consider magnifying the capabilities of artificial computing systems by following nature's lead. In a world where communities of computing systems may achieve new levels of complexity, one must keep a keen eye on nature's method of sustaining and regulating them. They must be dynamic, robust, and self-organized. When considering the parallels of nature and computer science, it is exciting to examine specific examples

7.1 A model of specialization in social honeybees

7.1.1 Introduction to honeybee specialization

The Western honeybee, *Apis mellifera*, is a provocative example of an insect that exists in communities. A typical colony is composed of 15,000 to 40,000 bees with constant fluctuations in population size and age demography. This variability can be due to colony development, time of year, food availability, predation pressure, and climatic conditions (Huang & Robinson, 1996). In spite of this, honeybees are able to maintain a constant balance of critical hive functions such as rearing the young and foraging. Entomologists have spent years observing the mechanisms that make honeybee colonies so dynamic and robust. The mechanisms for the division of labor in the hive have been of particular interest.

It is necessary to understand the typical behavioral development of a worker bee before discussing the mechanisms for the division of labor in a colony. Workers typically live for about 6 weeks and perform two major roles during that time. The first 3 weeks of their adult lives are spent inside the nest rearing the young; the final 1-3 weeks are spent outside the nest foraging. Logically, it is the bees which are in the final stages of their lives that assume the more dangerous role of leaving the nest. Entomological research has established a direct correlation between the behavioral development of a worker bee and its responsibility to labor in brood care or foraging

Although this temporal specialization exists under normal conditions, the ages at which a bee changes roles may vary drastically under certain conditions. Research showed that behavioral development can be accelerated, retarded, or even reversed in response to changes in the colony or environmental conditions. This research described in (Huang & Robinson, 1992) and (Huang & Robinson, 1996) led to the identification of two

mechanisms that regulate the behavioral plasticity, in a colony. The first is precocious foraging, which is when a young worker leaves the nest to forage earlier than its normal behavioral development would typically allow. The second is regressive nursing, which is when an older "foraging" bee returns to the nest to resume brood care.

(Huang & Robinson, 1996) studied how workers obtained the information that influenced their behavioral development. Considering the size of a typical honeybee colony, it is unlikely that an individual worker has either the capacity or the means to determine the global state of its colony. None the less, a colony has a "preferred state" and is "plastic" in the sense that it returns to this state after a disturbance. Huang performed extensive empirical tests to introduce dramatic disturbances to the age-structure of a honeybee colony so he could study its plasticity. In the first experiment, he depleted a colony of all of its foraging bees. This resulted in precocious foraging. In other words, young nursing bees left the hive early to make up for the lack of foragers. In the second experiment he confined foragers to their hive. This resulted in a slowed development of nursing bees. In the third experiment, he removed all young bees from a colony. This resulted in regressive nursing in many of the foraging bees. These fascinating experiments were actually able to induce both mechanisms – precocious foraging and regressive nursing – by isolating young nursing bees from old foraging bees. This indicated that social interaction was the local stimuli for balancing the division of labor in a colony of honeybees.

7.1.2 Activator-Inhibitor Theory

(Huang & Robinson, 1992) proposed an "activator-inhibitor" model to explain how the age structure of a colony can impact the behavioral development of honeybees through worker-worker interactions. This model describes in detail how the social interactions

between honeybees impact the behavioral development of a worker and ultimately the division of labor in a colony. In this model, the age that a worker first forages is determined by a ratio of chemical compounds.

The two compounds that regulate the behavioral development of a honeybee worker are designated as an “activator” and an “inhibitor”. The “activator” compound promotes the behavioral development of the worker. (Huang & Robinson, 1992) identified this “activator” to be the honeybee Juvenile Hormone (JH). JH is biosynthesized by a worker bee and increases in concentration with age. As the JH concentration increases the behavioral and physiological development that comes with maturity are activated. The “inhibitor” compound retards the behavioral development of a worker bee. (Leoncini et al., 2004) identified this inhibitor to be a pheromone known as ethyl oleate (EO). This pheromone is transmitted from one worker to retard the behavioral development of another. A bee’s ability to create EO appears to increase with age so that older bees are able to inhibit the development of younger bees, but not vice versa.

This inhibitor would be transferred from an older bee to a younger bee during social interaction inside the hive. This interaction suppresses the behavioral development of the younger bees, thus keeping them in their nursing role inside the hive. If, however, a colony was deficient in older bees, the social interactions would occur less frequently and some younger bees would receive less inhibitor. These uninhibited young bees would consequently experience precocious development causing them to leave the hive early to forage. Conversely, when a hive is deficient in younger bees, the older bees are able to inhibit each other to the point that some will revert back to brood care.

7.1.3 Discussion

As discussed in the section 6.3.2.5, the community pattern exploits a so-called "economy of scale". That is, the large number of organisms in a community allows its members to perform specialized tasks to the exclusion of others. This same economy of scale can be realized in distributed computing systems. There are many techniques for distributing tasks across a set of computers. Some are very rigid, by "hard-coding" each machine to perform a specific task. Others are very complex and facilitate flexibility in task allocation.

To find an elegant balance between simplicity, flexibility, and functionality, one must look no further than nature to see that these same design decisions have been made before. The honeybee specialization is a simple example of balancing a set of entities between two states. Their method of dividing labor has been proven successful and could be used for specific applications in computer science. The potential for such a method starts to surface when one considers the possibilities of ubiquitous computing and ambient intelligence. When very small and simple task specific computing systems are embedded in our environment, they could self-organize to perform supportive tasks. A more immediate area of consideration could be new forms of software licensing, digital rights management (DRM), and data security. The potential in these areas lies in the fact that the artifacts must be in one of two states: accessible or inaccessible.

Although the honeybee model of specialization is binary in the sense that it balances between two states, there are many other types of communities in nature from which we can draw other examples of division of labor. The untapped potential for inspiration is promising. Not only is there a wealth of knowledge on the natural world

today, but mankind is continually uncovering new and exciting discoveries in the natural world from which computer scientists can learn.

7.2 Social Specialization Design Pattern

A simplified pattern can be extracted from the activator-inhibitor model for application as a software design pattern. It actually follows the "rule of three", except the instances of it can all be found in biology. A variety of species of bees use this pattern. Furthermore, cellular biology research has shown indications that cellular development is also based on local stimuli between cells.

- **Pattern Name**: Social Specialization Design Pattern
- **Intent**: To illustrate the use of biology as a model for task allocation in distributed or multiagent systems.
- **Also Known As**: Activator-Inhibitor Specialization, or Honeybee Specialization
- **Motivation (Forces):** A distributed and generalized election algorithm or for balancing two tasks between a group of independent agents.
- **Applicability:** Network communications, embedded systems, distributed systems, Digital Rights Management (DRM).
- **Structure:** The class diagram in Figure 16 illustrates the structure of the Honeybee Specialization design pattern.

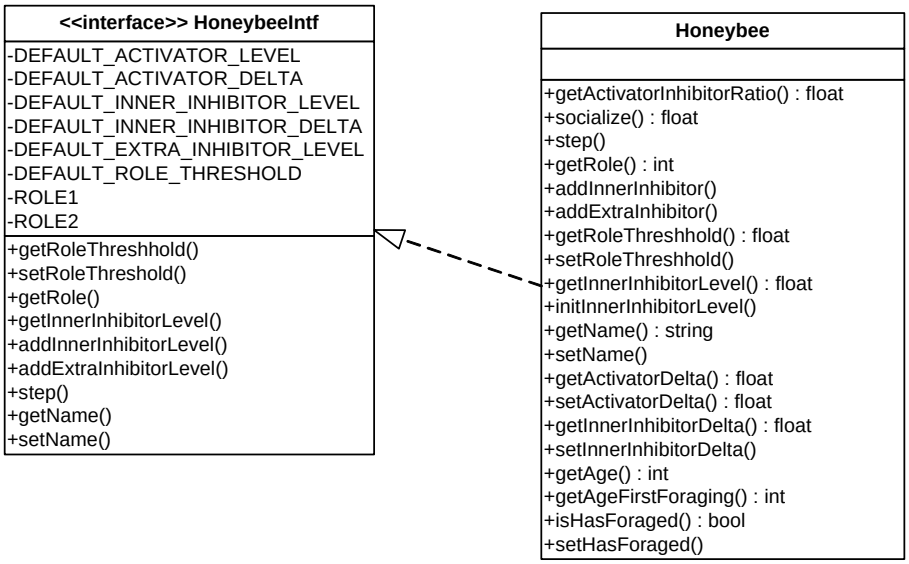

Figure 16: Class diagram for the Social Specialization design pattern.

- **Participants:** Honeybee Interface, Honeybee Instance
- **Collaboration:** Multiple Honeybee instances are intended to interact or "socialize' with each other. Each Honeybee instance has an internal activator and inhibitor variables that is incremented with time. When an interaction takes place the values of these inhibitor values are exchanged between the instances. The younger instance reduces its activator value based on the inhibitor value from the older instance. The internal ratio of activator/inhibitor determines the role of that Honeybee instance.

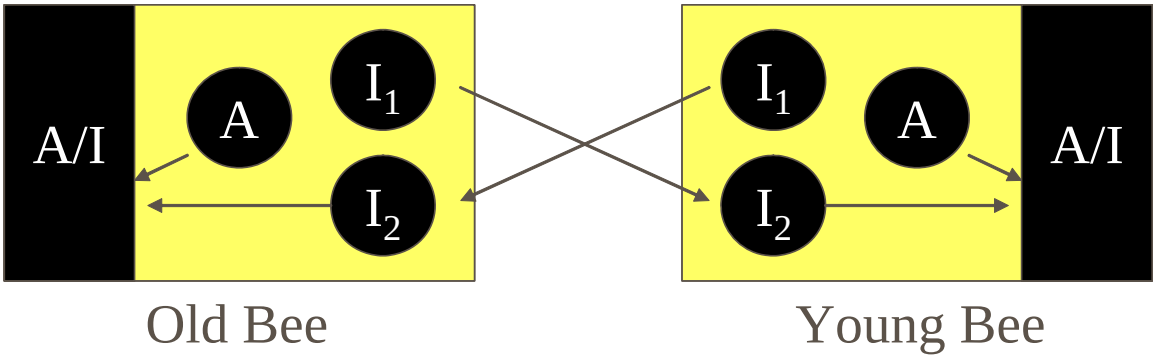

Figure 17: Activator-Inhibitor data flow from (Naug & Gadagkar, 1999)

- **Implementation**: An implementation of this pattern can be done using a multiagent system such as RepastJ. A simulation can be written that defines a space, a schedule, and an agent. It can also be implemented on a set of peer-to-peer processes or distributed systems.
- **Sample Code**: See Appendix A.

7.2.1 Application in Networking and Communications

There is also potential in addressing the communication infrastructure of wireless devices. Mesh networks have been an area of growing popularity. In a mesh network, nodes can communicate with neighboring devices. To reach distant devices, traffic is passed from neighboring device to neighboring device. In some cases a node may act as a bridge to a long range communication channel such as the Internet. This bridge node can then forward traffic from its local mesh network to the Internet.

Consider the scenario where each node had the ability to uplink to the Internet, but only did so when necessary. That is, if on a given mesh network, there were nodes that had an established Internet connection, then the other nodes on that network would route their traffic to it. If on the other hand, there were no nodes connected to the Internet, or those that were connected were already near capacity, then it would establish its own

Internet connection and then offer it to other nodes. This scenario maps nicely to the activator-inhibitor model if one considers a node to be like a worker bee. This node may mature from a "young" node with only a local mesh connection to an "old" node with an Internet connection. A mesh network like this could use the activator-inhibitor model of specialization to self-organize into a balanced ratio of local and bridged nodes dynamically to conserve bandwidth.

In this scenario, a node would have both internal "activator" and "inhibitor" variables. As the activator increased it would promote the behavioral development toward establishing its own Internet connection. If this node was in isolation, it would quickly establish this connection. If, however, it was in contact with a bridged node, that bridge would transmit an "inhibitor" message that retarded the development of the node and prevented it from establishing its own Internet connection. This method of balancing connections would maximize the use of bandwidth for a network of devices. This could be particularly useful in mobile wireless devices such as cell phones, laptop computers, embedded systems, and automotive applications. It would maximize the utilization of bandwidth and reduce the strain of concurrent connections on the infrastructure.

7.2.2 Application as a Distributed Election Algorithm

Synchronization is a critical issue in the field of distributed computing. It is often the case in a distributed system that one process be selected as a coordinator, initiator, or in some way perform a special role. Having a single coordinator is particularly useful when trying to synchronize a set of distributed peer processes.

Election algorithms are often used in distributed systems to designate one process among many to perform a special role – such as coordinator. This is frequently the case when any process in the distributed system is capable of taking this special role, but it

doesn't matter which one actually gets designated. (Tanenbaum & Steen, 2002) discusses two common election algorithms: "The Bully Algorithm" and "The Ring Algorithm". Both of these algorithms assume that each process has a unique numeric identifier. Furthermore, they both identify the process with the highest valued identifier as the coordinator.

Although the Bully Algorithm and the Ring Algorithm are powerful tools, the activator-inhibitor model could be used in a more flexible way. It is capable of electing N coordinators, which is the generalized version of electing a single coordinator. The activator-inhibitor algorithm would be relatively simple to implement as an election algorithm. It would perform the following steps:

- Begin incrementing an "activator" variable upon startup.
- The first node to reach the A/I threshold would become a coordinator.
- This newly appointed coordinator would then begin inhibiting all other nodes to prevent them from doing the same thing.
- This algorithm could be adjusted to support various numbers of coordinators.

7.3 An early alternative to activator-inhibitor

An early alternative to the activator-inhibitor was referenced in (Huang & Robinson, 1992). Apparently (Lindauer, 1952) proposed that honeybee specialization was influenced by an interaction between a bee and the nest. If we consider this concept in conjunction with the activator-inhibitor model, there may be some interesting implications. The basic idea here could be to allow the "nest" to inhibit its "bees" to keep them in a specific state. In the absence of the nest, these "bees" could mature into a new state. This

paradigm could be mapped into a client-server architecture. That is, a client's state (binary) could be regulated by its interaction with a server.

7.3.1 Application in Data Security, DRM, and Software Licensing

Data Security, DRM, and software licensing are matters of great concern today. Government agencies are facing stolen laptops with top secret information. Media companies are losing revenue from illegal copying of their content. Software companies are battling unlicensed use of their products. There are a multitude of schemes and practices to deal with these issues, but there is an ever increasing "arms race" between the providers and hackers. There is still a strong need for a way of managing approved use or information without hindering those that rightfully have access.

There is a need to develop new models of security for highly sensitive data. One approach would be to make data expire when lost or stolen. The activator-inhibitor model of specialization could facilitate a "time-out" for data ("bees") to expire or become unusable when it has been isolated from its source ("nest"). In this way, data would start out as fully accessible, but as it matured it would naturally expire.

To better understand this, let's consider a simplified implementation using a replication-based system such as Lotus Notes. In such a system, database files are replicated from a server to a local machine. These local copies are essentially unregulated and could be compromised. However, it would be possible to embed a monitoring application on the client system that manages an "activator-inhibitor" relationship with the server. This monitor would automatically increment an "activator" variable on the client. This activator would ultimately trigger the monitor to expire the data. This expiration could be enforced by encrypting or deleting the local replicas, depending on the need. If however, the monitoring application is in communication with the Lotus Notes server, it

would receive “inhibitor” messages that would prevent the data from expiring. In this way, the ratio of activator to inhibitor would regulate the accessibility of the local replicas.

7.4 Social Inhibition in Interdisciplinary Collaboration

Some generalizations can be made from the social inhibition theory that may have applications within the realm of interdisciplinary research. A key concept to extract is that social interaction and communication are critical enablers of specialization. So it is with human disciplinary specialization. A specialist is allowed to be such because they do not have the burden of learning every function. If each specialist was required to learn every role, the depth of knowledge in each area would suffer. By collaborating with experts in other disciplinary fields, an individual is freed from having to learn the depths of that discipline.

Chapter 8

8 CONCLUSION

8.1 Conclusion

Human knowledge is continuously advancing. Progress leads to new technological advances, which in turn facilitate further growth of knowledge. Though beneficial, this cycle of knowledge and technology has increased the complexity of the human experience. Individuals deal with more information and technology then ever before. Advancements in knowledge and technology have necessarily led to the formation and specialization of disciplinary knowledge. This specialization has concentrated efforts to increase understanding; however, it has also led to fragmentation and isolation of disciplinary fields. This isolation can be responsible for narrow disciplinary viewpoints, communication barriers, and a tendency to reinvent the wheel. This thesis identifies an opportunity to reconnect fragmented disciplinary knowledge through interdisciplinary collaboration.

This opportunity is particularly relevant to computer science. Computing systems and the software they run are some of the most pervasive and enabling technological advancements in history. Software is the codification of human knowledge and processes and as such increases in complexity with that knowledge. Software also introduces its own forms of complexity. Rampant IT growth, source code complexity, pervasive software, and emerging technologies are pushing individual human limitations and have an impact on nearly all sectors of society.

This thesis has proposed a vision for computer science that recognizes the problems of fragmented knowledge and systemic complexity. This vision promotes knowledge sharing and the tearing down of walls between disciplinary silos. It also builds an

appreciation for the natural world as a source of innovative and proven designs. This vision has been established through the presentation of current and historical examples of individuals with interdisciplinary knowledge who have used nature as a source of inspiration. Neurophysiologist Warren McCullough and mathematician Walter Pitts created the first Artificial Neural Networks. Applied mathematician and electronics innovator Norbert Weiner originated Cybernetics. Otto Schmitt, Jack Steel, and Janine Benyus have formalized and promoted Biomimetics, Bionics, and Biomimicry respectively to enhance human innovation. Finally, Genrich Altshuller used his interdisciplinary knowledge and passion to develop the TRIZ theory of inventive problem solving which is now being extended by Julian Vincent to include nature's solutions. All of these people have diverse disciplinary knowledge and have used that knowledge to bridge disciplinary boundaries and leverage the natural world as a source of innovation. It is through their examples that a new paradigm for software innovation has been presented.

To realize this vision, a framework has been presented to provide strategies that will facilitate the open exchange of disciplinary knowledge and natural design models. This framework includes interdisciplinary education, interdisciplinary collaboration, interdisciplinary tools, biomimetic design, and the creation of new pattern languages based on nature's design solutions. When taken together, the vision and strategies presented are intended to inspire and foster a paradigm that recognizes and harnesses the value of human and natural diversity as a source of innovation.

8.2 Summary of Contributions

This thesis has attempted to address the problems of complexity and fragmentation of knowledge by making the following contributions:

1. It has presented the historical development of disciplinary silos and the various approaches that have been developed to bridge them. This development provides a context for the objectives of the thesis.
2. It has presented an historical survey of nature inspired design and its methods to inform computer scientists of the value and means for using nature as a model for human innovation. This model can be applied to the development of both software products and processes.
3. It has proposed a vision for the open exchange of knowledge between disciplinary silos as a means to increase the breadth and depth of interdisciplinary knowledge in the field of computer science.
4. It has proposed a framework to for injecting creativity and innovation in software development through interdisciplinary collaboration and nature inspired design. This framework can be used as a means to realize the proposed vision.
5. It has identified the use of specific tools as part of that framework. These tools include the Russian "Theory of Inventive Problem Solving" called TRIZ as well as Cybernetics. TRIZ has thus far had very limited application in software development, but shows promise for further advancement.
6. It has proposed the creation of a new nature inspired pattern language as part of the proposed framework. This pattern language may potentially be supplemented with UML and integrated with TRIZ to facilitate the

discovery of successful design solutions during a software development process.

7. It has proposed a new "Honeybee Specialization" software design pattern as a case study of interdisciplinary collaboration and biomimetic design. This pattern was implemented in a multiagent simulation to control the division of labor between agents capable of performing two roles. This design pattern shows promise for various applications including a new generalized election algorithm. The simulation has also been shared with a preeminent biologists doing field research on honeybees and has the potential to enhance their understanding of biology. This shows the opportunity for bidirectional enrichment of disciplinary collaboration.

8.3 Future Research

This research has breached a number of subjects on which careers can and have been built. It has only been possible to scratch the surface of each of these to develop a high-level interdisciplinary knowledge. Further research into crossdisciplinarity, multidisciplinarity, interdisciplinarity, transdisciplinarity, biomimicry, bionics, cybernetics, TRIZ, human-factors, human-computer interactions, and more would ultimately enrich this research further. Ideally, all of this research should be expanded with the help of an interdisciplinary team. With that understanding, there are a number of specific topics of interest for future research.

- To better understand the contributions that Cybernetics can bring to the issues of interdisciplinary collaboration and software complexity. There appear to be opportunities for Cybernetic theory to help inform the

organizational logistics of team formation and interactions. Additionally, its constructs may aid in the development of more self-regulating software.

- To validate the concepts put forth in this thesis by leading an interdisciplinary software development effort that takes the time to explore nature's solutions for inspiration and innovation.
- To monitor current and future research that attempts to adapt TRIZ for use in software development and apply it to a real world software development project.
- To continue to explore the development of a software pattern language based on natural models. Furthermore, to incorporate this pattern language into TRIZ so that common problems and solutions can be captured and reused.
- To further explore the analogy between emergent software development processes and nature's development processes.
- To continue to develop and refine a software model of honeybee specialization. Furthermore, to apply this development to a generalized election algorithm.
- To continue the interdisciplinary collaboration with entomologists to enrich their understanding of honeybee specialization for use in biology.

APPENDIX A: PARTIAL SOURCE CODE FOR SOCIAL SPECIALIZATION

```
/**  This interface defines the attributes and methods needed to  implement division of labor based on the way natural honey bees divide labor in a hive. @author SKS832  */
public interface HoneyBeeIntf {
/** The level of activator pheromone.  This cannot be affected by other  instances of the HoneyBeeIntf. */
      public float DEFAULT_ACTIVATOR_LEVEL=0.0F;
/** The delta value that is added to the innerInhibitorLevel with each incremental step (of time). */
      public float DEFAULT_ACTIVATOR_DELTA=1.0F;
/** Level of internal inhibitor pheromone which cannot be affected by external instances of the HoneyBeeIntf. */
      public float DEFAULT_INNER_INHIBITOR_LEVEL=1.0F;
/** The delta value that is added to the innerInhibitorLevel with each incremental step (of time). */
      public float DEFAULT_INNER_INHIBITOR_DELTA=1.0F;
/** The level of external inhibitor pheromone which is received from other instances of HoneyBeeIntf. */
      public float DEFAULT_EXTRA_INHIBITOR_LEVEL=1.0F;
/** The threshold of the activator/inhibitor ratio which separates the two roles. */
      public float DEFAULT_ROLE_THRESHOLD=15.0F;
/** Constant that identifies the first role.  Ccorresponds to young honey bees in the hive. */
      public static final int ROLE1 = 1;
/** Constant that identifies the second role. Corresponds to the role of older honey bees as foragers. */
      public static final int ROLE2 = 2;
/**  @return The threshhold of the activator/inhibitor ratio which separates the two roles. */
      public float getRoleThreshold();
/**  Sets the value of the role threshhold.
  * @param threshhold The threshhold of the activator/inhibitor ratio which separates the two roles.*/
      public void setRoleThreshold(float threshhold);
/**   Possible values are defined by role constants ROLE1 and ROLE2.
  * @return The role of the HoneyBeeIntf.  */
      public int getRole();
/**  Gets the value of the innerInhibitorLevel, which can be transmitted to the extraInhibitorLevel of another instance of  HoneyBeeIntf by passing it as a parameter to the addToExtraInhibitor  method of the other instance.
* @return The level of internal inhibitor which cannot be affected by external instances of the HoneyBeeIntf. */
      public float getInnerInhibitorLevel();
/**  Resets the innerInhibitorLevel to its initial state.  Should be used immediately after a social interaction that transmitted this innerInhibitor to another instance of HoneyBeeIntf. */
      public void initInnerInhibitorLevel();
/**  Increments the innerInhibitorLevel by the value assigned to the innerInhibitorDelta.  Basically, this does the following calculation:  innerInhibitorLevel = innerInhibitorLevel + innerInhibitorDelta */
      public void addInnerInhibitor();
/**  Adds the value passed into the method to the extraInhibitorLevel.
  * @param transmittedInhibitor Should correspond to the innerInhibitorLevel of the "other" instance of HoneyBeeIntf taking part in a social interaction with this instance.*/
      public void addExtraInhibitorLevel(float transmittedInhibitor);
/** Advances this instance to the next state.  This step should increment both the activatorLevel and the innerInhibitorLevel by their corresponding delta values.  */
      public void step();
/**  Implement method to define interaction between the current instance of the HoneyBeeIntf with another instance of the HoneyBeeIntf.
  * @param other The HoneyBeeIntf instance encountered for this social interaction.
  * @return Resulting ratio of activator/inhibitor after the social interaction. */
      public float socialize(HoneyBeeIntf other);
      public String getName();
      public void setName(String name);
}
```

```
public class HoneyBee implements HoneyBeeIntf {
	public float socialize(HoneyBeeIntf other) {
		System.out.println("HoneyBee.socialize(): " + HoneyBee.numberOfSocialInteractions++
					+ " " + this.getName() +
					" initiated contact with " + other.getName());

		// TODO There is a descrepency that occurs when the initiating agent
		//  gets the inhibitor value of the other, before the other has incremented.

		// Get the values of the other agent.
		float otherInhibitToMe = other.getInnerInhibitorLevel();

		// Let the other inhibit me
		this.addExtraInhibitorLevel(otherInhibitToMe);

		// Inhibit the other with my inner pool of inhibitor
		other.addExtraInhibitorLevel(this.getInnerInhibitorLevel());

		// QUESTION:  Should my inhibitor pool be diminished during an interaction?
		//		Perhaps it is diminished slightly, fully, or not at all.
		System.out.println("HoneyBee.socialize(): "
					+ this.getName() + " age=" + this.getAge()
					+ " A=" + this.activatorLevel + ", iI=" + this.innerInhibitLevel
					+ ", eI=" + this.extraInhibitLevel + ", A/I="
					+ this.getActivatorInhibitorRatio() + " Role=" + this.getRole());

		// If other has higher inhibitor level than me
		if(otherInhibitToMe>this.getInnerInhibitorLevel()) {
			// Reduce my activator and innerInhibitor level so I can't inhibit others as well.
			this.addInnerInhibitor(-(2*this.getActivatorDelta()));
			this.activatorLevel-=2*this.getActivatorDelta();
		}

		// Reset the innerInhibitorLevel for me
		//this.initInnerInhibitorLevel();

		// Reset the innerInhibitorLevel for other
		//other.initInnerInhibitorLevel();

		return 0;
	}
	public void step() {
		this.age++;
		this.activatorLevel+=this.getActivatorDelta();
		this.innerInhibitLevel+=this.getInnerInhibitDelta();
		System.out.println("HoneyBee.step(): "
					+ this.getName() + " age=" + this.getAge()
					+ " A=" + this.activatorLevel + ", iI=" + this.innerInhibitLevel
					+ ", eI=" + this.extraInhibitLevel + ", A/I="
					+ this.getActivatorInhibitorRatio() + " Role=" + this.getRole());
	}
```

```
	public float getActivatorInhibitorRatio() {
		// TODO Should inner and extra inhibitors be sum in the ratio?
		//float airatio = this.activatorLevel/this.extraInhibitLevel;
		//System.out.println("HoneyBee.getRole(): airatio=" + airatio);
		float a = this.activatorLevel;
		//float i1 = this.innerInhibitLevel;
		float i2 = (float)this.extraInhibitLevel;
		return a/(i2);
	}

	public void addInnerInhibitor(float iiDelta) {
		this.innerInhibitLevel = this.innerInhibitLevel + iiDelta;
	}

	public void addExtraInhibitorLevel(float transmittedInhibitor) {
		this.extraInhibitLevel = this.extraInhibitLevel + transmittedInhibitor;
	}

	public void initInnerInhibitorLevel() {
		// When this is zero, it causes a divide by zero for the a/i ratio of
		// the other instance that is inhibited by this one.
		this.innerInhibitLevel=0.0F;
	}

	/**
	 * Sets the age of first foraging.  This method will only set this value
	 * once during the lifespan of a HoneyBee instance.
	 *
	 * @param ageFirstForaging The ageFirstForaging to set.
	 * @return true=Age Set, false=Not set because this instance has already set this variable.
	 */
	public boolean setAgeFirstForaging(int ageFirstForaging) {
		// Make sure this is the first transition to Role2.
		if(this.ageFirstForaging==0) {
			this.ageFirstForaging = ageFirstForaging;
			this.setHasForaged(true);
			System.out.println("HoneyBee.setAgeFirstForaging(): " + this.getName()
					+ " first foraged at age " + this.getAge());
			return true;
		} else {
			// This agent already has an age at first foraging.
			return false;
		}
	}
}
```

BIBLIOGRAPHY

ACM. (2002). Managerial information overload. *Communications of the ACM, 45*(10), pp127-131.

APS. (2000). Warren S. McCulloch Papers. Retrieved 2/27/2007, 2007, from http://www.amphilsoc.org/library/mole/m/mcculloc.htm

Beck, K., & Cunningham, W. (1987, September 17, 1987). *Using Pattern Languages for Object-Oriented Programs.* Paper presented at the OOPSLA-87, Orlando, FL.

Benyus, J. (2005). Biomimicry Design Portal Meeting Invitation. In F. Lodato (Ed.) (pp. Email correspondance between Janine Benyus and Franco Lodato. At the time, Lodato was VP of Design Exploration and Development at Herman Miller. This invitation was forwarded to me to attend the workshop in Toronto in February 2006.). Zeeland, MI.

Benyus, J. M. (2002). *Biomimicry: Innovation Inspired by Nature* (Reprinted Paperback ed.). New York: Harper Perennial.

Bernsen, J. (2004). *Bionics in Action: The Design Work of Franco Lodato, Motorola*: StoryWorks ApS, Denmark.

Biomimicry. (2006a). Biomimicry Design Process (http://www.biomimicry.net/designmethodologyA.html): The Biomimicry Guild.

Biomimicry. (2006b, 2/22/2006). *Biomimicry Portal Workshop.* Paper presented at the Biomimicry Portal Workshop, Toronto, Ontario.

Biomimicry. (2007). Biomimicry Institute Website. *http://biomimicryinstitute.org/*, 2007

Bogatyrev, O., Pahl, A.-K., & Vincent, J. (2002). *Enriching TRIZ with Biology: The Biological Effects database and implications for Teleology and Epistemology.* Paper presented at the 2 ETRIA World Conference, Strasbourg.

Changquing, G., Zezheng, H., & Fei, M. (2005). Comparison of innovation methodologies and TRIZ. *TRIZ Journal*(September 2005), 50-57.

De Jong, K. A. (2008). Evolving Intelligent Agents: A 50 Year Quest. *IEEE Computational Intelligence Magazine, 3*(1), 12-17.

Dictionary, A. H. (2004). The American Heritage Dictionary of the English Language. Fourth Edition. Retrieved 06 Oct. 2005, from http://www.answers.com

Domb, E. (1997). Contradictions: Air Bag Application. *The TRIZ Journal, July 1997*(July 1997).

Dowdeswell, E. (1995). Global Biodiversity Assessment In V. H. Heywood (Ed.), *Global Biodiversity Assessment* (pp. 1152): Cambridge University Press.

Easterling, K. (2001). Walter Pitts. *Cabinet, Winter 2001/02.*

Fenn, J., & al, e. (2006a). *Hype Cycle for Emerging Technologies 2006* (No. ID Number: G00141901): Gartner, Inc.

Fenn, J., & al, e. (2006b). *Hype Cycle for Human-Computer Interaction, 2006* (No. ID Number: G00141150): Gartner, Inc.

Francois, C. (1999). Systemics and Cybernetics in a Historical Perpective. *Systems Research and Behavioral Science, 16*, 203-219.

Friedberg, R. M. (1958). A learning machine: Part I. *IBM Journal, 2*(1), pp. 2-13.

Fullbright, R. (2004). TRIZ and Software Fini. *The TRIZ Journal, August*(August 2004).

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software* (Hardcover ed.). Boston, MA: Addison-Wesley.

Ganek, A. G., & Corbi, T. A. (2003). The dawning of the autonomic computing era. *IBM Systems Journal, 42*(1), 5-18.

Grasso, M. (2003, Spring 2003). Interdisciplinary Computer Science Introduction. *ACM Crossroads, 9*.
Gray, C. H. (1995). An Interview with Jack Steel. In C. H. Gray (Ed.), *The Cyborg Handbook* (Paperback ed., pp. 61-69). New York, NY: Routledge.
Harkness, J. M. (2002). A Lifetime of Connections: Otto Herbert Schmitt, 1913-1998. *Physics in Perspective, 1*(4), p456-490.
Hewett, Baecker, Card, Carey, Gasen, Mantei, et al. (1992). ACM SIGCHI Curricula for Human-Computer Interaction: ACM.
Hofstadter, D. R. (1979). *Gödel, Escher, Bach : an eternal golden braid*. New York: Basic Books.
Huang, Z., & Robinson, G. E. (1992). *Honeybee colony integration: Worker-worker interactions mediate hormonally regulated plasticity in division of labor*. Paper presented at the National Academy of Science.
Huang, Z., & Robinson, G. E. (1996). Regulation of honey bee division of labor by colony age demography. *Behav Ecol Sociobiol, 39*, 147-158.
IFTF. (2006, May 23-25, 2006). *Science & Technology Perspectives 2005-2055*. Paper presented at the IFTF Tech Horizons, San Mateo, CA.
Jorgensen, P. (2001). UML Presentation for CS611. In uml.ppt (Ed.), *Powerpoint* (pp. 1): Grand Valley State University.
Kinnersley, B. (2006). The Language List. Retrieved 04/08/2006, 2006, from http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm
Klein, J. T. (2003). Unity of Knowledge and Transdisciplinarity: Contexts of Definition, Theory and the New Discourse of Problem Solving: Eolss Publishers.
Kostoff, R. (2002). Overcoming Specialization. *BioScience, 52*(10), pp 937-941.
Lattanzi, M. (1998). *Transdisciplinarity: Stimulating Synergies, Integrating Knowledge*. Paper presented at the Symposium on Transdisciplinarity, Val-d'Oise, France.
Lenarcic, J. (2004). *Behavioural Issues in Software Development: The Evolution of a New Course Dealing with the Psychology of Computer Programming*. Paper presented at the InSITE 2004 (Informing Science + IT Education), Rockhampton, Queensland Australia.
Leoncini, I., Le Conte, Y., Costagliola, G., Plettner, E., Toth, A. L., Wang, M., et al. (2004). Regulation of behavioral maturation by a primer pheromone produced by adult worker honey bees. *Proc Natl Acad Sci U S A, 101*(50), 17559-17564.
Leong, L., & Basso, M. (2005). *Wide Array of Communications Overwhelms Users* (Brief): Gartner, Inc.
Lerner, L. (1991). Genrich Altshuller: Father of TRIZ. *Ogonek*.
Lindauer, M. (1952). *Z. Vgl. Physiol, 34*, 299-345.
Livio, M. (2002). *The Golden Ratio: The Story of Phi, The World's Most Astonishing Number* (Paperback 2003 ed.). New York: Broadway Books.
Lodato, F. (2005). The Nature of Design. *Design Management Review, 16*(1).
Loebmann, A. (2002). The TRIZ-Methodology - an always ongoing innovative cycle. *The TRIZ Journal, March 2002*(March 2002).
Lones, M. A., & Tyrrell, A. M. (2001). *Biomimetic Representation in Genetic Programming*. Paper presented at the Computation in Gene Expression.
Lyman, P., & al, e. (2003). *How Much Information 2003?* (Research Report): UC Berkeley's School of Information Management and Systems.
Mann, D. (2004). TRIZ For Software? *The TRIZ Journal, October 2004*(October 2004).
McNeil, D. A. a. G. (1992). *Artificial Neural Networks Technology*. Rome, NY: Rome Laboratory.
Meyer, B., & Arnout, K. (2006). Componentization: The Visitor Example. *IEEE Computer, 39*(7), 23-30.

Michalec, L., & Banks, D. A. (2004). *Information Systems Development Methodologies and all that Jazz*. Paper presented at the InSITE 2004 (Informing Science + IT Education), Rockhampton, Queensland, Australia.
Michalewicz, Z., & Fogel, D. B. (2004). *How to solve it: modern heuristics* (2nd ed. ed.): Berlin; New York: Springer.
Nakagawa, T. (2005, April 17-19, 2005). *Software Engineering and TRIZ: Structured Programming Reviewed with TRIZ*. Paper presented at the TRIZCON2005: The 7th Altshuller Institute TRIZ Conference, Detroit, MI.
Naug, D., & Gadagkar, R. (1999). Flexible Division of Labor Mediated by Social Interactions in an Insect Colony - a Simulation Model. *Journal of Theoretical Biology, 197*, 123-133.
Nicolescu, B. *The transdisciplinary evolution of learning*: Learning Development Institute.
Nicolescu, B. (2002). *Manifesto of Transdisciplinarity* (t. f. t. F. b. K.-C. Voss, Trans.). New York: State University of New York (SUNY) Press.
Nissani, M. (1995). Fruits, Salads, and Smoothies: A Working Definition of Interdisciplinarity. *Journal of Educational Thought, 29*, pp119-126.
Olariu, S., & Zomaya, A. Y. (2006). *Handbook of Bioinspired Algorithms and Applications*. Boca Raton, FL: Chapman & Hall/CRC.
Patterson, D. A., Brown, A., Broadwell, P., Candea, G., Chen, M., Cutler, J., et al. (2002). *RecoveryOriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies* (Technical Report No. UCB//CSD-021175). Berkeley, California: UC Berkeley Computer Science.
Podborschi, V., & Vaculenco, M. (2005). Study of Natural Forms - The Source of Inspiration in Product Design. In D. Talaba & T. Roche (Eds.), *Product Engineering: Eco-Design, Technologies and Green Energy* (pp. 111-120): Springer Netherlands.
Raskino, M. (2005). *Findings From the 'Symposium/ITxpo Cannes 2005' Research Community: Information Overload is a Material Issue* (Brief No. ID Number: G00136452): Gartner, Inc.
Rea, K. C. (1999). Using TRIZ in Computer Science - Concurrency. *The TRIZ Journal, August*(August 1999).
Rea, K. C. (2001a). TRIZ and Software - 40 Principle Analogies, Part 2. *The TRIZ Journal, November*(November 2001).
Rea, K. C. (2001b). TRIZ and Software - 40 Principle Analogies, Part 1. *The TRIZ Journal, September*(September 2001).
Rea, K. C. (2002). Applying TRIZ to Software Problems -- Creatively Bridging Academia and Practice in Computing. *The TRIZ Journal, October*(October 2002).
Reitsma, F. (2002). A response to simplifying complexity [Electronic Version]. *Geoforum*, 34, 13-16. Retrieved 03/15/2008 from http://hdl.handle.net/1842/1057.
Roussev, B., & Roussev, Y. (2004). *Software Development: Informing Sciences Perspective*. Paper presented at the InSITE 2004 (Informing Science + IT Education), Rockhampton, Queensland, Australia.
Salamatov, Y. (2005). *TRIZ: The Right Solution at the Right Time: A Guide to Innovative Problem Solving* (O. Kraev, Trans. 2nd ed.). Krasnoyarsk, Russia: Intitute of Innovative Design.
Salazar, J. (2006). *Oil and Water do Mix: Social Science meets Engineering Towards a Transdisciplinary Perspective on Cyberworlds*. Paper presented at the Cyberworlds 2006, Lausanne, Switzerland.
Seipel, M. (2004). *Introducing Interdisciplinarity*. Paper presented at the Conference Name|. Retrieved Access Date|. from URL|.
Stanbrook, T. (2002). *TRIZ for software process improvement*. Paper presented at the 28th Annual International Computer Software and Applications Conference (COMPSAC 2002).

Steel, J. (1995). How do we get there from here? In C. H. Gray (Ed.), *The Cyborg Handbook* (Paperback ed., pp. 55-59). New York, NY: Routledge.
Tanenbaum, A. S., & Steen, M. v. (2002). *Distributed Systems : principles and paradigms*. Upper Saddle River, N.J.: Prentice Hall.
Tate, K., & Domb, E. (1997). 40 Inventive Principles With Examples. *The TRIZ Journal, July 1997*(July 1997).
TRIZJ. (unknown). What is TRIZ? [Electronic Version]. *The TRIZ Journal* from http://www.triz-journal.com/whatistriz_orig.htm.
Vincent, J., Bogatyreva, O., Bogatyrev, N., Bowyer, A., & Pahl, A.-K. (2006). Biomimetics: its practice and theory. *Interface: Journal of The Royal Society, 3*(9), 471-482.
Vincent, J., & Mann, D. (2000). TRIZ in Biology Teaching. *The TRIZ Journal*.
Vincent, J., & Mann, D. (2002, November 7-9, 2001). *Systematic technology transfer from biology to engineering*. Paper presented at the World Conference "TRIZ Future 2001", Bath, England.