The Proceedings of the International Conference on Creationism

Volume 6
Print Reference: Pages 133-148

Article 14

2008

Modeling Biochemical Processes as Designed Systems

Steven M. Gollmer
Cedarville University

In A. A. Snelling (Ed.) (2008). Proceedings of the Sixth International Conference on Creationism (pp. 133–148). Pittsburgh, PA: Creation Science Fellowship and Dallas, TX: Institute for Creation Research.

Modeling Biochemical Processes as Designed Systems

Steven M. Gollmer, Ph.D., Department of Science and Mathematics, Cedarville University,Cedarville, OH 45314-0601

Abstract

Being in the post-genomic era, there is a need for new methodologies from an interdisciplinary perspective, which can complement current genomics research. Bioinformatics and systems biology are rapidly growing research areas that are meeting this need. Operating with the assumption that there is design with a purpose, creationists provide a unique perspective for discovering order in the complexity of genes, regulatory networks, and biochemical reactions.

Since the genome acts as an information storage system, it seems reasonable to apply design concepts, originating from computer and network programming, to make sense of genomic information. One such concept is that of design patterns, which has been formalized by programmers and analysts working with object-oriented programming (OOP). Several patterns are introduced and related to biochemical systems in the cell.

A more detailed analysis of the observer pattern is made in the context of galactose metabolism in *Saccharomyces cerevisiae*. Since design patterns embody good OOP practice and do not specify a specific implementation, it is possible to explore a variety of implementations that can achieve regulation of galactose metabolism. This methodology can complement current research approaches by clarifying what is meant by system homology at the biochemical level.

Keywords

Design patterns, Galactose metabolism, Object oriented programming, *Saccharomyces cerevisiae*, Systems biology, Systems theory, Transcription networks

Introduction

Information gathered through genetic sequencing has increased dramatically in the past decade. Entire genomes of many organisms have been sequenced and the process has culminated in the completion of the human genome project. Although the sequencing of different organisms is still a productive avenue of research, the greater challenge, in a post-genomic era, is determining the structure and regulation of the genome and its expressed proteins. Prior to genomic sequencing, physical maps were generated relating protein expression to chromosome location. However, a genomic sequence provides a complete and more accurate picture of the information stored in an organism's genome. Prior to the sequencing of *Saccharomyces cerevisiae*, budding yeast, its physical map consisted of about 1,200 genes (Feldmann, 2000). With its genome completely sequenced by 1996, it was found that the *S. cerevisiae* genome consists of more than 6,000 genes (Cherry et al., 1998).

With the wealth of information gained through genetic sequencing, it was necessary to integrate sophisticated techniques of data analysis with biological understanding. This has given rise to the inter-disciplinary field of bioinformatics. Through pattern matching and statistics, it has been possible to identify Open Reading Frames (ORFs), which are locations of potential protein expression (Velculescu et al., 1997). For *S. cerevisiae*, 72% of its genome consists of coding sequences with the remainder containing signals for replication and regulation of gene expression (Feldmann, 2000). Using microarrays it is possible to determine the presence of mRNAs and proteins expressed by a cell given particular environmental conditions. In addition, protein structure and function can be inferred from the amino acid sequence coded in the genome. However, the current understanding of the *S. cerevisiae* genome has 1,145 ORFs listed as uncharacterized and 815 listed as dubious (Saccharomyces Genome Database, 2007).

Systems biology goes beyond bioinformatics by studying an organism as "an integrated and interacting network of genes, proteins and biochemical reactions which give rise to life" (Institute for Systems Biology, 2006, p. 1). Having a much broader scope, systems biology draws from the expertise of biologists, chemists, physicists, mathematicians, and computer scientists. Providing a broader conceptual framework for this holistic approach to biology is general systems theory, which was introduced by Ludwig von Bertalanffy in

the 1950s (Skyttner, 1996). General systems theory proposes that there are ordering principles inherent in complex systems that are independent of the particular system being studied (Bertalanffy, 1968). This allows physicists, computer scientist, and scientists of other disciplines to apply their intuition of complex systems to biological systems. The focus of systems biology is to model cellular systems by discovering interaction networks and performing computer simulations so as to determine recurring patterns of order (Hieronymus & Silver, 2004; Vazquez, Dobrin, Sergi, Eckmann, Oltvai, & Barabasi, 2004).

Modularity

The biochemical and regulatory mechanisms of the cell are numerous and involve overlapping composite networks (Yeger-Lotem et al., 2004). In spite of this complexity, systems biology research indicates that cellular functions are modular (Hartwell, Hopfield, Leibler, & Murray, 1999; Herrgard, Covert, & Palsson, 2003). Rives and Galitski (2003, p. 1132) studied the yeast filamentation network to develop a model that "reduces the complexity of this network to a small number of connected units of structure and function". Snel and Huynen (2004, p. 392) compared functional modules across 110 genomes and concluded that they are "significantly more modular than random;" however, they also concluded that there is limited modularity in yeast transcriptional data sets. While studying the galactose utilization pathway, de Atauri, Orrell, Ramsey, and Bolouri, (p. 29, 2004) concluded that there are "recurring, dynamic organizational principles in biochemical pathways" and "computer modeling and simulation can be used to identify and study such 'evolutionary design principles'".

These "evolutionary design principles" are regulatory mechanisms involving positive and negative feedback. The types of order reported by the systems biology community are of limited complexity and involve network motifs that lead to optimal gene circuits (Alon, 2007; Milo, Shen-Orr, Itzkovitz, Kashtan, Chklovskii, & Alon, 2002). It is the hierarchical design of modular structures that lead to the orderly behavior of biochemical systems on the cellular level (Ravasz, Somera, Mongru, Oltvai, & Barabasi, 2002). Since their research is approached from an evolutionary paradigm, it is assumed that the optimal interactions are chanced upon by the system due to random processes and shifting goals (Kashtan & Alon, 2005; Cordero & Hogeweg, 2006). To acknowledge the existence of design principles is a bold statement given the observation by de Atauri et al. (2004, (p. 28) that "many experimentalists argue that cellular pathways are the idiosyncratic result of eons of evolutionary tinkering whose behavior cannot be understood in terms of engineering principles".

Design

In stark contrast to the evolutionary paradigm is the claim of scripture that states "His invisible attributes are clearly seen, being understood by the things that are made, even His eternal power and Godhead, so that they are without excuse" (NKJV, Romans 1:20). From the Genesis account of creation it is clear that God, a purposeful and personal creator, made the universe and all that is in it. The design of His creation is clearly evident so that no one has any excuse to dismiss His existence. This worldview was present in the writings of John Ray, William Paley, and the authors of the Bridgewater Treatises. Opposition to the design argument was present at this time and Paley used the first chapter of his book *Natural Theology* to summarize them. Without surprise, the arguments against design are the same today only updated in their terminology. With the publication of Darwin's *Origin of the Species*, Aristotelian final causes were rejected and natural theology lost influence in the scientific world.

The modern day intelligent design movement is dominated by two significant ideas. The first is irreducible complexity, which was proposed by Michael Behe (1996) in his book *Darwin's Black Box*. Behe proposes that biochemical and cellular systems can be reduced to a minimal functional system, which is still too complex to explain through natural causation. The second is specified complexity as described by William Dembski (2001) in *The Design Inference*. Dembski proposes that complexity by itself is insufficient to indicate design, but complexity must be linked to a specification or pattern that is unlikely to occur given the probabilistic resources present in the known universe. Other design arguments exist, but they invariably lead back to the extreme improbability that the state of the universe and the existence of life in the universe could occur through natural causes.

Although, the intelligent design movement distinguishes itself from natural theology, it still falls short of establishing itself as a productive scientific paradigm. Dembski (2001) observed that the natural theology movement used observations of the natural world to draw conclusions about reality beyond this world (p. 1). In contrast, the intelligent design movement separates the question of "Can design be detected in the natural world?" from the question of "Once design is acknowledged, who is the designer?" The first question is seen as scientific while the second is seen as a theological or philosophical question. In spite of this distinction, both natural theology and intelligent design are based on an argument of incredulity, "Isn't it amazing that…." This is an a posteriori approach where the conclusion is made after the facts are given. This approach is important

in science; however, to have a productive research paradigm there must also be an a priori approach. This predictive component drives scientists towards new discoveries.

Although Ray's work *The Wisdom of God Manifest in the Works of the Creation* (Ray, 1979) predates the industrial revolution, the popularity of natural theology in England was framed in this context. Paley's arguments allude to the precision of manufacturing and the intermeshing of gears to create machines (Paley, 1850). This approach was used to explain the design of biological systems, such as the eye. However, this classical view of design creates the notion that there is a platonic idealized form, or best design, which is unchanging and imperfectly represented in the physical world (Ruse, 1999, p. 3). It is clear from selective breeding that, although there are limits, there is still a significant amount of variability within species to give rise to different coloration, size, and feature shapes (Wood, Wise, Sanders, & Doran, 2003) When Darwin proposed natural selection as a driver for variation, it supplanted the rigid view of design proposed by natural theology.

The view of design proposed by natural theology resonates with modern day creationists given that God's creation was perfect before the fall. However, to limit God's creation to a rigid design lessens the wonder of God's handiwork and prevents design from being a productive paradigm for scientific research. In two centuries, mankind has moved from an industrial revolution to an information revolution. Individuals experienced with managing information and complex systems recognize that good design is a balance between competing constraints. Since these constraints can change, that design must also be reusable, robust, and adaptive.

The remainder of this paper proposes a means of extending the definition of good design. Relating the behavior of biological systems to that of complex computer software systems, a broader view of good design will be developed. In section 1, biological systems are related to the Object Oriented Programming (OOP) paradigm and the concept of design patterns is introduced. In section 2, a design pattern is applied to the transcription regulatory networks of the model organism *S. cerevisiae*. In section 3, this new paradigm is used to propose a means of studying biochemical systems and potentially restoring lost function in biological systems affected by the fall.

Biochemical systems and OOP

Cellular systems

At the cellular level, an organism consists of numerous components that interact in a predictable manner. Each component has a set of delegated responsibilities such as the mitochondria handling oxidative metabolism and the nucleus containing and controlling the genetic information of the cell. Responsibilities within the cell are neither exclusive nor simple. Although the nucleus is responsible for the replication and transcription of DNA, the mitochondrion contains its own DNA, which is transcribed and replicated. The information storage mechanism of the cell is a double strand of DNA consisting of a sequence of base pairs strung together by a phosphate-deoxyribose backbone. However, the base pair sequence not only stores information for protein expression and regulation, but also provides structural information that allows the DNA to wind around histones to form nucleosomes, which in turn pack together to form chromatin fibers. Neither shared responsibilities nor complexity diminishes the fact that these are interacting components.

An object representation of cellular components embodies the requirement that complex biological systems behave in a predictable and reliable manner. Restricting a series of biochemical reactions to the interior of a mitochondrion prevents those reactions from interfering with the complex control mechanisms within the nucleus. If all cellular processes were exposed to each other, there would be a dilution effect as well as unexpected interference between reactions. Considering 1,000 biochemical species, the number of potential interactions could be nearly a half a million. Restricting processes within physical structures or carrying out processes over different timescales reduces the number of interactions and, therefore, reduces the complexity of the organism. This statement is made not to infer that biological systems are simple, but that they are less complex than they potentially could be.

Just as biological systems can conform to an object representation; it appears that this is also true of biochemical reactions. Since each biochemical species falls within a family of compounds, the number of chemical interactions is reduced. This reduction in possibilities not only makes biochemical interactions more predictable, but also assists researchers as they search for proteins of homologous structure and function (Pritsker, Liu, Beer, & Tavazoie, 2004). The specificity of protein interaction allows multiple enzymes and transcription factors to occupy the same cellular component without producing unexpected interactions.

Information systems and OOP design

In like manner, computer based information systems also require a component structure to their software. Failing to restrict the interaction between segments of computer code and data leads to fragile and error prone systems. This approach to programming was historically known as writing spaghetti code and was

generated either through a pragmatic easiest route to a solution or through a necessity to generate a highly efficient, tightly interacting program to satisfy a very specific need. The first motivation for spaghetti code indicates little foresight about future changes and parallels a bottoms-up approach to design analogous to the evolutionary paradigm. The second motivation for spaghetti code indicates a high level of planning for the express purpose of generating a static optimal design commensurate with an industrial revolution view to design.

As software systems increase in size and complexity, it becomes necessary to sacrifice optimal speed and size to achieve reliability and reusability in the face of change. Although a number of programming paradigms exist, it is acknowledged that the OOP paradigm can improve software reliability and reusability through the use of encapsulation, inheritance, polymorphism, and abstraction.

To achieve encapsulation, data and methods for operating on the data are placed within objects. This makes the interior of the object a black box. Access to the object's function and data is restricted to a limited number of well defined interactions. Although DNA is not physically isolated from the biochemistry of the nucleus, its data is still encapsulated in functional space with access granted under specific conditions, such as attachment of a DNA polymerase in the presence of a primer to initiate replication.

Inheritance allows a family of objects with similar data and function to be constructed from a common parent object. In the context of transcription regulation, transcription activators consist of two domains. The first binds to specific sequences on the DNA while the second binds to the transcription machinery. A whole family of activators can be generated by changing the DNA binding domain in order to recognize different DNA sequences. The *GAL4* activator, discussed later in this paper, has this property and is used in the yeast two-hybrid system to identify cDNAs (Feldmann, 2005, p. 9).

Polymorphism makes use of inheritance to provide a change of behavior through the interchange of different family members. A family of binding proteins has the ability to recruit the machinery necessary for transcription. However, each member of the family recognizes a different DNA binding site within the promoter region of the gene. Therefore, different proteins can be generated by interchanging family members and yet using the same transcription machinery.

Abstraction allows one to access objects at the level of detail that is necessary for the desired interaction. DNA can be viewed at different levels of abstraction. When the cell is in S phase, the DNA sequence of base pairs is exposed through the replication machinery to allow a duplicate to be made. During prophase, the DNA chromatin restricts access to the base pairs and condenses into a chromosome. During anaphase the chromosome is accessed by its centromere so that spindle fibers can attach and separate the daughter chromatids. At each level of abstraction, the object may hide or expose data and functions in order to meet the need at hand.

Design patterns

Those experienced with OOP recognize that there are recurring programming problems that are best solved by isolating the portion of the program or system that is expected to change from that which is more static. This improves the reusability of the static portion and allows the programmer to focus only on the dynamic portion of the code. This requires analysis of the whole system and focuses on establishing appropriate interactions between the components of the system (Shalloway & Trott, 2005).

Through this process of analysis, it was found that there are recurring patterns of best practice solutions for computer and network systems. Gamma, Helm, Johnson, & Vlissides (1995) identifies twenty-three of these patterns and categories them as either creational, structural, or behavior design patterns. The names of the individual patterns by category are provided in Figure 1. Since 1995, the number of design patterns have increased significantly; however, the initial 23 patterns are not as specialized and provide a starting point for relating design patterns to biological systems.

One of the creational patterns, builder, separates the construction of an object from its representation. Figure 2A provides the standard Unified Modeling Language (UML) diagram for the builder pattern (Gamma et al., 1995). The director initiates the construction of an object by communicating with the builder object. The builder could be one of a family of builder objects. When the instruction is received, the

Creational	Structural	Behavioral	
Abstract Factory	Adapter	Chain of Responsibility	
Builder	Bridge	Command	Observer
Factory Method	Composite	Interpreter	State
Prototype	Decorator	Iterator	Strategy
Singleton	Facade	Mediator	Template Method
	Flyweight	Memento	Visitor
	Proxy		

Figure 1. The 23 design patterns identified by Gamma et al. (1995) fall into three different categories. Creational patterns describe ways of dynamically constructing objects once a program is running. Structural patterns provide a means of encapsulating data and function into objects. Behavioral patterns define object interaction, which can be extended to provide reliable complex behavior.

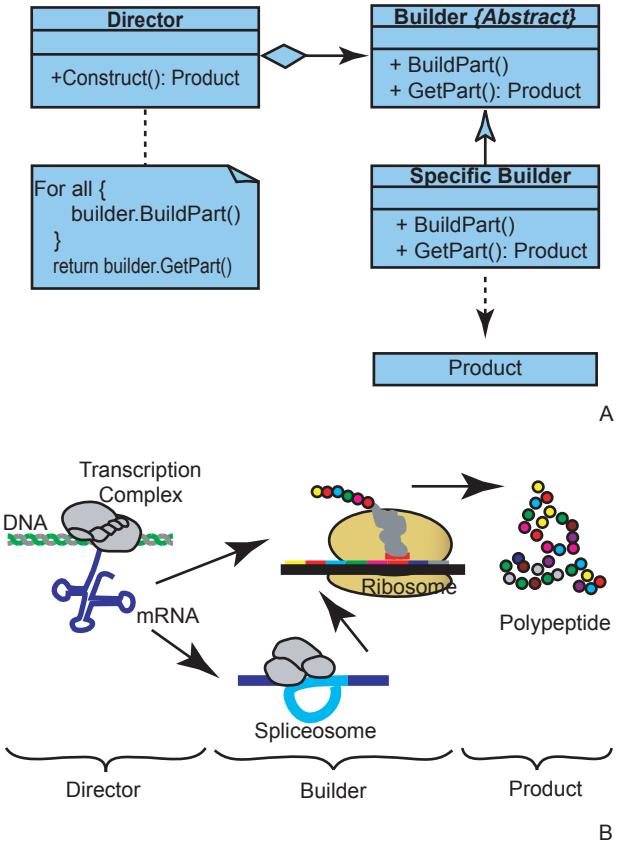

Figure 2. The builder pattern separates the request for constructing an object from the construction process. The standard UML representation of the builder pattern (A) consists of a director, which can access a variety of builder objects. Depending on which builder is accessed, a particular product is generated. The construction of a polypeptide from DNA (B) can be described in the context of the builder pattern. The director is a mRNA molecule, which is transcribed from the DNA. The builder in this process can take on several forms. One is the translation of RNA codons into an amino acid sequence in the presence of a ribosome. A second form also uses a ribosome, but first modifies the RNA strand by removing introns and assembling exons in the presence of a spliceosome.

builder then goes through the process of generating a product. A biological example of the builder pattern is the construction of a protein. As illustrated in Figure 2B, the director is the RNA transcription machinery. When a transcription activator is present, the director generates an mRNA strand, which acts as a message to construct a protein. When the message is received by a ribosome, the protein is constructed. In this illustration, the ribosome is the builder and the protein is the product. It is possible that the building process is more complex than this and a different builder needs to be implemented. Perhaps the builder is a multi-step process which involves a spliceosome and a ribosome. In the presence of exonic splicing enhancers and suppressors, different proteins can be constructed from the same initial strand of RNA (Ast, 2005). The benefit of the builder pattern is that any modifications in the process are encapsulated in the builder and are isolated from the activity of the director.

Adaptor is a structural pattern that mediates between two systems with different input and output requirements. This design pattern allows two potentially incompatible systems to communicate with each other. The client in Figure 3A belongs to the first system and has an expectation on how to give instructions. The adaptor has knowledge of the adaptee, which belongs to the second system, and provides a result based on interaction with the client. Two incompatible information systems in the cell are DNA/RNA, which uses nucleic acid bases, and protein, which uses amino acids. The adaptor between these two systems is tRNA. As illustrated in Figure 3B, the anticodon end of tRNA receives an instruction from the client, mRNA, and delivers an adaptee, the appropriate amino acid, for addition to the polypeptide. The fidelity of the translation processes is achieved by aminoacyl tRNA synthetase, which makes sure the correct amino acid is associated with its corresponding tRNA (Cooper, 1997, p.275). Either end of the tRNA adaptor could be modified

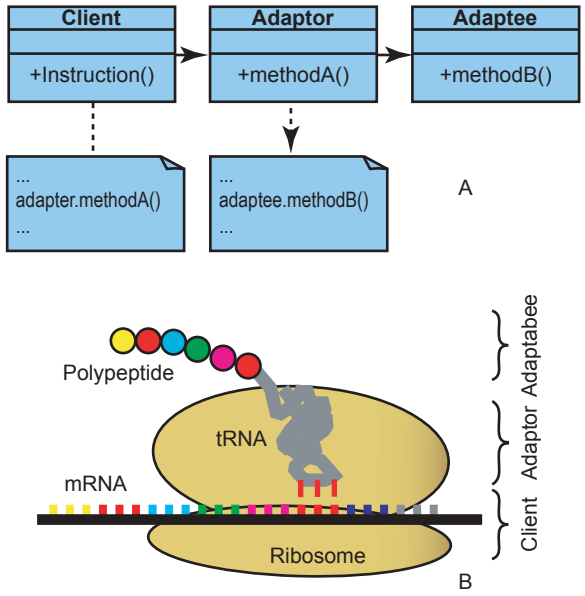

Figure 3. The adaptor pattern allows instructions to be passed between two incompatible systems. The UML representation of the adaptor pattern (A) consists of a client, which issues the instructions, an adaptor, which associates the instructions to a different representation, and an adaptee, which issues the instruction in the new representation. (B) In the process of constructing polypeptides from mRNA in the presence of a ribosome, tRNA serves the role of adaptor. It associates the three base codon to a specific amino acid for construction of the polypeptide.

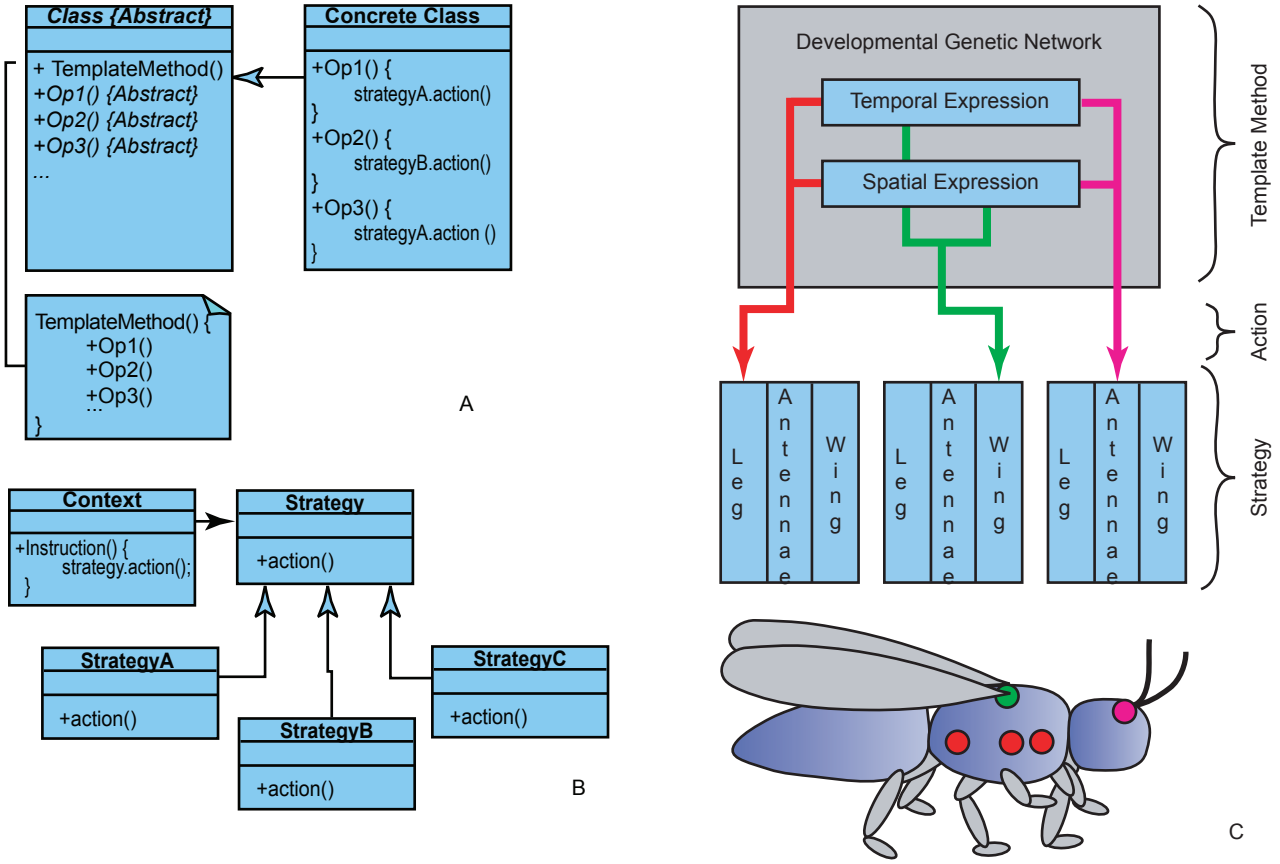

Figure 4. A class implements the template pattern (A) by adding a template method, which calls a sequence of instructions. These instructions are decoupled from the template by establishing a family of concrete classes, which can be called interchangeably. The family of classes called by the template method is another design pattern called a strategy (B). All members of a family have a common means for calling an action. However, each family member is designed to perform different actions. (C) An example of the template pattern and strategy is found in developmental biology. The developmental genetic network of an organism uses spatial and temporal cues to activate development of different body parts. Each body part behaves like a member of a strategy, which can be associated to the action of a particular template method.

to deal with changes in either the DNA codon or its associated amino acid. Because the cell makes use of this pattern, such novel work as a quadruplet codon or incorporating more than 21 different amino acids in a protein is possible (Anderson, Wu, Santoro, Lakshman, King, & Schulz, 2004; Turner, Graziano, Spraggon, & Schultz, 2006).

As an example of a behavioral pattern, the template pattern is used in concert with the strategy pattern. The template method pattern is a framework for conducting a number of different procedures. As illustrated in Figure 4A, the requested procedure can vary, but the manner in which it is requested remains the same. The variation of procedure is accomplished by using the strategy pattern. The strategy pattern is simply a family of objects that can be interchanged to implement functional polymorphism. The context from Figure 4B is just another object, in this case the template method, which has a means of calling one of the strategy's family members. Figure 4C illustrates how development within multi-cellular organisms makes use of the template method and strategy patterns. During development, the body plan of an organism is determined by gradients of proteins generated by toolbox genes, and these proteins activate the development of different appendages at the appropriate location (Carroll, 2005). This is similar to the template method, which determines when and where a procedure will be called, but does not participate in the activity of the procedure. This reduces the complexity of the development process because the body layout delegates development of individual appendages to the strategy pattern. Developmentally, these appendages are wildly different (legs, wings, and antennae); however, they are members of the same family because they implement a common means of initiating development from a targeted portion of the body. Since appendage development is decoupled from the body plan, it is possible to develop a leg in place of an antenna, as in the Drosophila mutation antennapedia (Emerald & Cohen, 2004).

Complementary view of design

It is seen from the examples given above that design patterns are at least analogously expressed in biological systems. A goal of general systems theory is to discover ordering principles of complex systems, which are independent of the system being studied. Within this context Babaoglu et al. (2005) proposes biologically based design patterns for use in distributed computing. Perhaps design patterns are one way of defining these ordering principles. Since design patterns use encapsulation, inheritance, polymorphism, and abstraction to manage variability within a system, these patterns can provide a means of expressing a more dynamic view of design. While the classical view of design focuses on the optimal construction of simple components, the dynamic view of design focuses on the complex interaction between components. This complementary view of design involves analysis of the whole system and requires a choice between numerous possible optimal components.

This complementary approach to design draws upon the strength of design patterns. Shalloway and Trott (2005) identify a number of advantages gained by using design patterns. The following are just a few of those advantages: (1) Focus is on the design, not on what works, (2) Provides a common language for communication, (3) Shifts thinking away from the details to the quality and types of interactions (p. 86). The first advantage acknowledges that there are numerous ways a system can be constructed to provide the desired behavior. However, there are design principles that make some constructions superior to others. The second and third advantages emphasize the necessity for a design language. Biologists have a bio-molecular language that expresses the details of DNA replication and transcription, such as promoters, activators, polymerases, etc.; however, a common design language can provide a means of communicating higher-level concepts such as adapter (moderation between two different means of representing information) and builder (means of separating the information from the construction of a molecular component).

This complementary approach to design also acknowledges the limitations of design patterns. Design patterns are not intended to replace the finely tuned and optimized code that defines the behavior of each component in the system. Likewise, it does not mandate that the code within each component remain static. As long as component interactions, as defined by the pattern, are maintained, the system can dynamically adjust to changes in the system's environment. Also, it is not realistic to expect design patterns to describe a whole system. Patterns are not a programming language, but an embodiment of what is good OOP design. Each system has interactions that are unique and, therefore, require unique solutions.

Biochemical Networks in *S. cerevisiae*

To better understand the role and utility of design patterns in biochemical systems, the regulation of the galactose metabolism in *S. cerevisiae* is studied. *S. cerevisiae* is an appropriate organism for this study since it is the simplest eukaryotic model organism and it has been extensively studied. Its simplicity is relative to other organisms as it is single-celled and has a genome consisting of 12.8 million base pairs (Feldmann, 2000). Since it is a eukaryote, it contains features present in more complex organisms, such as organelles within the cell and linear chromosomes with transposable elements and telomeres. Although only 4% of its genes contain introns, it maintains the sophisticated machinery necessary for splicing RNA molecules (Lopez & Seraphin, 2000).

Galactose metabolism

Galactose metabolism is not a primary source of energy for *S. cerevisiae*; however, when present and not inhibited by the presence of glucose (Rønnow, Olsson, Nielson, & Mikkelsen, 1999), the cell is able to activate the appropriate genes in a concerted manner through the use of Gal4 binding sites. Activation and regulation of expression levels for these genes depend on numerous factors (Travern, Jelicic, & Sopta, 2006); however, de Atauri et al. (2004) identify seven genes directly related to galactose metabolism. Figure 5A illustrates the relative position of these genes within the *S. cerevisiae* genome. The position of the Gal4 binding site for each gene is illustrated as a red square. The regulatory genes have a single binding site while the structural genes consist of multiple binding sites (Hittinger, Rokas, & Carroll, 2004). The role of the structural proteins is illustrated in Figure 5B. *GAL1*, *GAL10*, and *GAL7* are located on chromosome II and code for proteins that catalyze the conversion of galactose into glucose 1-phosphate (de Atauri et al., 2004). *GAL2* is located on chromosome XII and its protein facilitates the transport of galactose across the cell membrane. Expression levels of these four genes are strongly affected by the presence of a dimer of *GAL4*'s DNA binding protein (de Atauri et al.). The remaining two genes, *GAL3* and *GAL80*, code for proteins that regulate the ability of the Gal4 dimer to activate transcription.

This regulatory mechanism is seen by de Atauri et al. (2004) as optimal and is described as follows. In the absence of galactose, Gal80 inhibits the Gal4 dimer from activating the transcription machinery. Genes with single binding sites have a basal transcription level; however, the ones with multiple binding sites have a much lower, but not quite zero expression

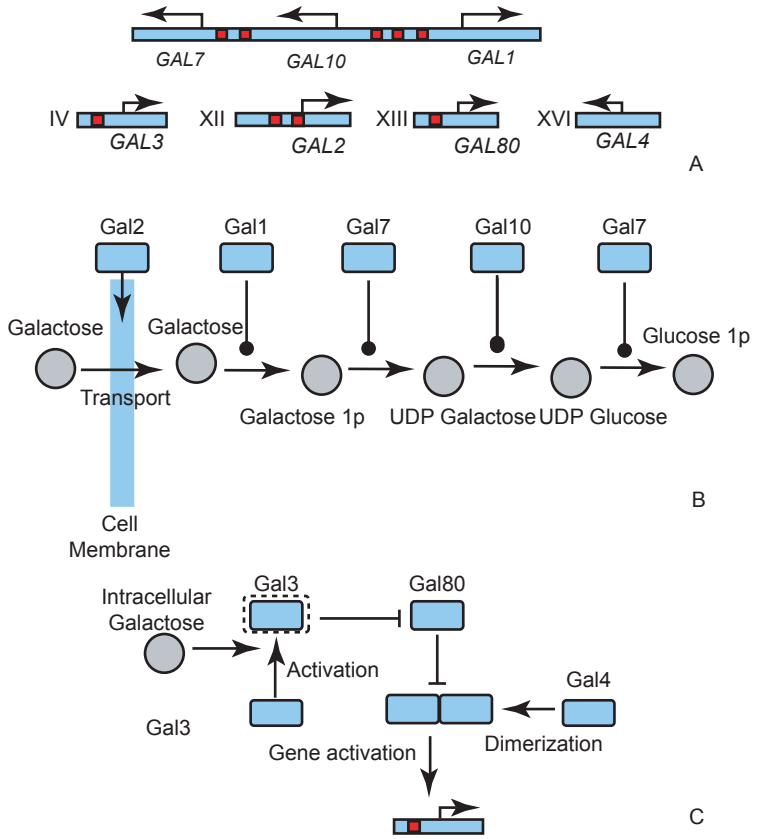

Figure 5. Based on the model of galactose metabolism from de Atauri et al. (2004) and Hittinger, Rokas, & Carroll. (2004), (A) Gal4 activation sites are shared by structural and regulation proteins involved in galactose metabolism as indicated by the red squares. (B) The sequential enzymatic activity of Gal1, Gal7, and Gal10 metabolize galactose into glucose 1-phosphate, while Gal2 facilitates the transport of galactose into the cell. (C) Gal80 normally inhibits a dimerized form of Gal4 from activating transcription. However in the presence of galactose, Gal3 becomes activated and in turn inhibits the action of Gal80. This protein-protein interaction allows Gal4 to quickly activate transcription of the associated genes.

level because Gal80 is more effective at repressing transcription (de Atauri et al.,p.34). Therefore, small quantities of galactose, when present, will be transported into the cell due to Gal2. As illustrated in Figure 5C, when galactose is inside the cell, Gal3 becomes activated and interferes with Gal80's ability to inhibit gene expression by Gal4. This now allows Gal4 to activate transcription. Since inhibition of Gal4 is due to protein-protein interaction (Bhat & Murthy, 2001), this transition takes place very quickly. All of the genes with the Gal4 activation site will turn on and their respective proteins will be expressed. Within minutes the concentration of the galactose structural and regulatory proteins increases. This increase is most pronounced in the structural proteins, whose genes have multiple activation sites. Notice that when Gal1, Gal7, and Gal10 are available to metabolize galactose, the concentration of Gal2 has increased to facilitate the transport of even more galactose within the cell. When galactose is no longer present in the environment, Gal3 will become inactive and Gal80 will again inhibit Gal4's ability to activate transcription. At this point, the structural genes become inactive and the regulatory genes decline to their basal transcription levels.

Regulatory motifs

The use of activation sites, regulatory proteins, and inhibition demonstrates one means of controlling gene transcription. Using transcription networks of this type, Alon (2007) documents regulatory patterns that are common in gene networks. These patterns, or motifs, control expression rates and levels and provide stability for the gene network in the presence of mutation. If a mutation affects either the binding site or the DNA binding protein, the binding affinity can change and modify the expression level of the regulated gene (Liu & Clarke, 2002). This can have a cascading effect and may affect the whole cell. However, by providing feedback mechanisms in the network, the system as a whole is made relatively insensitive to these changes (Orrell & Bolouri, 2004).

Other motifs provide a means of optimizing

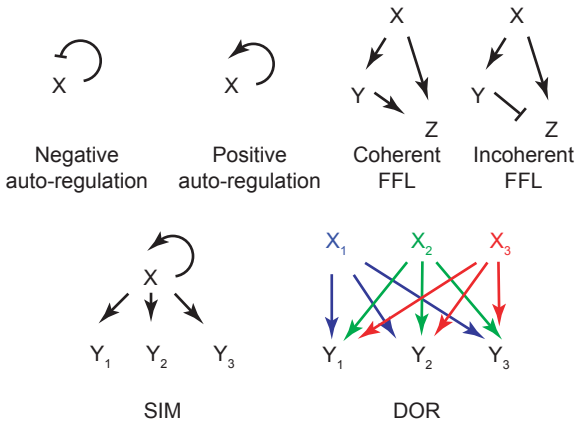

Figure 6. Common motifs exist in transcription networks as documented by Alon (2007). Auto-regulation occurs when the expressed protein either inhibits or enhances its own expression. Feed-forward loops (FFLs) provide means for filtering noise from the transcription network. Single input modules (SIMs) coordinate the expression of a number of genes. The dense overlapping regulon (DOR) can have varying degrees of complexity, but provides a means of generating combinatorial logic in transcription networks.

performance of the regulatory network. The addition of binding sites can enhance the transition between basal and activated expression levels; however, this increase in switching rate is also tied to larger expression levels. To achieve both fast transitions and moderate expression levels, negative auto-regulation can be applied (Alon, 2007, p. 33). Negative auto-regulation is achieved when a protein expressed by a gene is also an inhibitor of the same gene. If this self-regulation is changed from inhibition to activation, the gene acts like a switch, which will remain in a perpetual on state once it is activated. This is called positive auto-regulation.

These two motifs along with others are summarized in Alon (2007, p. 93) and are illustrated in Figure 6. The type-1 coherent feed-forward loop (FFL) is the most common FFL (Milo et al., 2002) and has the advantage of filtering out transient on and off signals. Protein X activates the expression of both Y and Z. Since expression of Z depends on both X and Y, brief interruptions in the presence of X can be moderated by the expression of Y. An incoherent FFL can generate a pulse when Z is initially expressed in the presence of X, but later inhibited by the presence of Y (Mangan & Alon, 2003). The single input module (SIM) demonstrates how X can activate multiple regulatory proteins (Y_1, Y_2, Y_3) and can generate a staggered activation of multiple genes due to the different expression rates of Y_1, Y_2, and Y_3. Dense overlapping regulons (DOR's) produce combinatorial logic as transcription factors interact to produce a secondary series of transcription factors.

These motifs represent optimizations in the expression and regulation of proteins and, when mutation occurs, provide a means of limiting change. As mentioned before, mutations in the binding protein, the activation site, or the inhibiting protein can affect the level of transcription of the controlled protein. Some motifs mitigate the affect of change by controlling the level of expression at the expense of timing. However, some activities in the cell are time sensitive and regulation of timing takes priority over expression levels. In either case, there is room for variation, which enables some genetic variants to optimally perform in a specific metabolic environment.

Although the network motifs described above represent a level of design in biochemical systems, they fall short of the language represented by design patterns. Network motifs describe the implementation of specific programming instructions while design patterns represent the optimal application of OOP goals. Therefore, the implementation of a design pattern may vary, but a network motif always has the same implementation.

Motifs vs. design patterns

The regulation of galactose metabolism demonstrates the difference between motifs and design patterns. The *GAL80* gene is activated by the dimer form of Gal4. As the level of Gal80 increases, it inhibits the action of Gal4 and, therefore, provides negative auto-regulation. Gal4 is always present in the cell since its gene is not regulated; however, the basal transcription level of Gal80 is affected by this feedback mechanism. When galactose is present and Gal3 is activated, the negative feedback is stopped and production of Gal80 increases. When galactose is no longer present, Gal3 becomes inactive and there is a surplus of Gal80 in the cell. This causes a strong inhibition effect on Gal4 and rapidly drops production of Gal80 (de Atauri et al., 2004). However, *GAL80* is not the only gene affected by the inhibition of the Gal4 dimer. The collective regulating effect of the Gal4 dimer is a component of a regulon and represents a design pattern.

The observer pattern illustrated in Figure 7A, consists of a subject and a number of observers. In this case, the subject is the presence of galactose and the observers are all of the genes activated by the Gal4 dimer. The observers are those genes that have the *GAL4* activation site and the notification mechanism is the activation by the Gal4 dimer. Alternate means of notification can be proposed in the observer pattern. Instead of an inhibition/inhibition interaction between Gal3, Gal80, and Gal4, activation could be accomplished directly with the *GAL4* gene. By means of genetic engineering, it seems reasonable

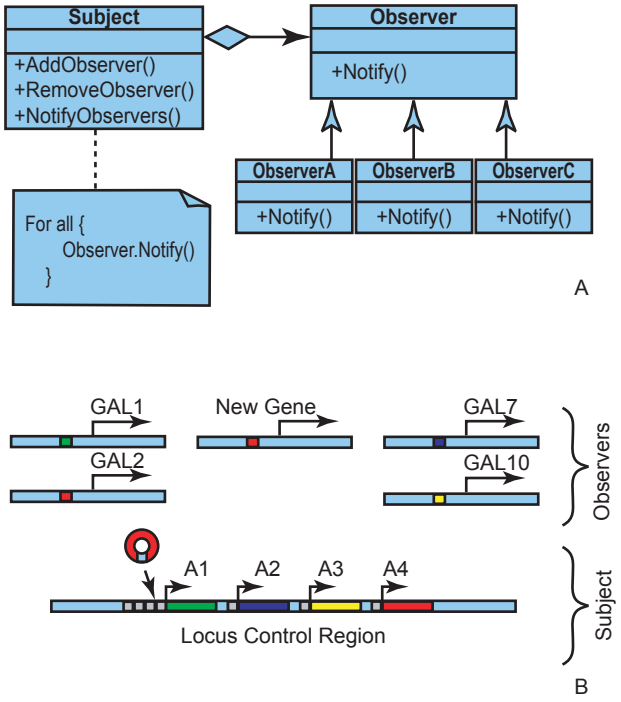

Figure 7. The observer pattern (A) allows multiple objects to be notified when the state of the subject changes. (B) Transcription activation binding sites can act as observers when the appropriate protein is present. Expression of the activation protein may depend on the state of a locus control region, which serves as the design pattern's subject.

that a novel activation site could be introduced to the promoter region of the *GAL4* gene. In addition, a DNA binding protein could be constructed that actively binds to this novel site in the presence of galactose. Although this implementation would work in theory, the rate at which structural genes are activated would be slowed because the concentration of Gal4 would have to build up once the *GAL4* gene was activated. In the context of this exercise, there are a number of ways that the observer pattern could be implemented, but some implementations are better than others. The best implementations correspond to network motifs and their variants.

Just as the observer pattern allows variety in the details of implementation, it also allows for variety in the scope of implementation. The galactose example given above is a very limited form of the observer pattern. It allows each gene to be notified; however, it is not evident that there is a dynamic means of removing a gene from the notification process. Usually in computer programs, the subject keeps a list of observers and individually notifies them when the subject's state changes. New observers can be added by calling a routine within the subject and in like manner observers can be removed from the subject's notification list. This means of notification could be implemented as illustrated in Figure 7B. Each gene could have different activation sites that respond to a series of binding proteins: A1, A2, A3, and A4. These binding proteins could be sequentially arranged in a locus control region and activated by a change of the chromatin structure in the presence of galactose or a galactose activated protein. Registering new observers would involve targeted insertion of new binding proteins in the locus control region. An observer could be removed by maintaining an identification region that allows the gene to be selectively removed intact. Once again, this is one implementation of many that can be imagined to provide the extended capabilities of the observer pattern.

As in the example above, design patterns are not specified structures, but time test best solutions to OOP problems. The context, within which the design pattern is applied, provides constraints for implementing a specific solution. The coordinated activation of genetic elements in the presence of an environmental or developmental change, will invariably involve an observer pattern. However, the best implementation depends on additional requirements of the system. For optimal activation time the notification process involves a rapid protein-protein interaction rather than a slower protein transcription process. It appears that the cell has other mechanisms for regulating protein expression. One of these is RNA interference, which complementarily binds to mRNA and either inhibits translation or reduces the concentration of mRNA through targeted destruction (Mundodi, Kucknoor, & Gedamu, 2005). Modeling of these interactions may provide a means of determining the context where this process is optimal for gene regulation.

Potential for an Extended View of Good Design

In light of the regulation of biochemical processes and more specifically the regulation of galactose metabolism in *S. cerevisiae*, an extended view of good design involves optimal regulatory and biochemical interactions within the context of a top-down OOP description of cellular interactions. The study of network motifs demonstrates that optimal regulatory interactions exist and they are expressed in organisms of all sizes (Alon, 2007, p. 90). These motifs represent modular components that can be assembled to form more complex systems. However, just as instructions of a programming language or the components of an electric circuit are modular components with predictable behavior, motifs define the components, but not the design of the system. The design of the system lies in the holistic interaction of all the system's components. As in OOP, it is necessary to apply good design principles to achieve reliability and reusability and these principles are, at least in part, embodied in design patterns.

Synthetic biology

The application of design patterns to biochemical systems is most clearly seen in the context of synthetic biology. Synthetic biology goes beyond genetic engineering by constructing biological systems that don't exist in nature and redesigning existing systems in order to understand biological systems (Chopra & Kamma, 2006, p. 401). The M.I.T. Registry of Standard Biological Parts consists of over 140 parts, which are segments of DNA that perform a specific function when inserted in a chromosome (Gibbs, 2004, p. 77). These BioBricks are treated as interchangeable components and have the potential of being assembled into complex systems. As understanding of the biochemical processes of the cell increases, the potential for inserting a complex biochemical program into the genome becomes realizable.

The design of a biochemical program involves programming, technical, and ethical considerations. The complexity of the cell at the molecular level is vast and insertion of a program can produce unexpected results. Unlike a computer program, which can be restricted to a single processor or thread, a biochemical program must operate concurrently with other cellular processes in the same physical space and compete for the same resources. To mitigate this problem, biochemical programmers will need to consider design principles that produce fault tolerant operation in an asynchronous distributed environment. This process involves a significant understanding of the whole cellular system and the operation of the biochemical program within this context.

Assuming, adverse interactions can be avoided through the use of good OOP design, technical issues rest upon a complete understanding of gene regulation and replication. Inserting a biochemical program into a cell could use a process similar to infection by a retrovirus. However, without targeted insertion into the genome, cellular function could be lost due to disrupted genes or promoter regions. The size of a biochemical program can be another technical hurdle. The program could be broken up into individual segments and inserted as a series of artificial chromosome vectors. However, with multiple vectors the probability of successfully incorporating all vectors becomes increasingly small. An alternate approach is to place the program within a separate artificial chromosome. This task would require a complete understanding of the structural and regulatory cues embedded within the chromosome for replication, nucleosome formation, and condensation into chromosomes.

Of greater consideration in the development of biochemical programs are the ethical issues. On the pragmatic level, there is a fear that accidental or purposefully hazardous programs could be released into the environment. Given the economic and social loss due to computer viruses, it gives one pause to think of releasing a malicious biochemical program that could potentially infect all of life on the planet (Block, 2001). Even beneficial programs can have unintended effects. By re-engineering organisms to efficiently manufacture expensive chemicals or to clean up hazardous waste, the balance of processes within the organism is changed and affects how it interacts with its environment. Although these specialized organisms are intended to operate within controlled environments, accidental release into the global environment could have unanticipated effects (Joy, 2000).

Beyond pragmatics, one must consider the spiritual implications of modifying living systems. Although, the dominion mandate in Genesis 1:28 gives mankind a directive to understand and rule over the creation, the principle of stewardship provides balance to make sure mankind's rule is benevolent. Although ethical and spiritual issues apply to organisms in general, they become more pronounced when applied specifically to mankind. It is beyond the scope of this paper to address this significant issue, but many ethicists are addressing the issues of personhood, sanctity of life, and the implications of genetic engineering and transhumanism (Waters, 2006). For the remainder of this section, the discussion will focus on the potential of restoring lost function, which is less controversial than the use of bioengineering to extend function.

Restoration of function

In order to restore lost function, it is necessary to know what was originally present in the system. As the genome of an organism is replicated and impacted by its environment, it can experience point mutations, translocations, insertions, and deletions. Identifying lost genetic function and seeking effective therapies to counteract this loss has been the ongoing job of medical researchers for decades. An effective means of accomplishing this goal is to genetically screen a population of individuals suffering from a particular malady. Comparing their genomes with those from a healthy population, it is possible to identify genetic markers that predispose an individual to this malady. At the genetic level, it is possible to identify which gene is mutated and how it should be corrected.

Since correcting a mutation would involve changing the genome of each cell in the organism, it is more practical to identify the protein coded by the gene and provide a medication or therapy that would substitute for the defective protein. This process is not easy since the functions of proteins coded by the genome are not completely known. A method for determining protein function is to search databases

for homologous proteins in other organisms, whose function has been identified (Hodges, McKee, Davis, Payne, & Garrels, 1999). A type of colorectal cancer in humans was linked to a mutation in a gene whose protein corrects for mismatch repair (Watson, Baker, Bell, Gann, Levine, & Losick, 2004, p.241). This protein is homologous to MutS in *E. coli*.

A complimentary approach to the standard medical research paradigm is that of systems biology. This rapidly growing field attempts to understand biological processes in a holistic manner (Powell, 2004). Although proteins perform specific functions, these functions operate in the context of a biochemical system. If the system is well understood, then it is possible to infer what functional proteins must be present to maintain the system (Mak, Daly, Gruebel, & Ideker, 2007). Through the techniques of proteomics (2-D gel electrophoresis, mass spectroscopy, and bioinformatics) a link can be made between proteins expressed when the system is active and the protein coding sequences in the genome (Watson et al., 2004, p.678). This provides a means of determining the function of proteins linked to currently uncharacterized ORFs.

Thinking in terms of a system rather than components enables a researcher to study the broader impact of lost genetic function. Although a component has become inactive due to mutation, the system may compensate such that the impact of the change is relatively minor. For a well designed system, multiple changes may occur before significant impairment is observed. It is in this context that an extended definition of design may provide some insight into the robustness of biochemical systems and the restoration of function due to genetic changes.

At the biochemical level, an extended definition of design consists of a collection of interoperating systems conforming to good OOP design principles. These systems are not restricted to biochemical reactions, but include the genetic information, transcription and translation processes, and all regulatory mechanisms that service the system (Johnston, Chang, Etchberger, Ortiz, & Hobert, 2005). This approach assumes that cellular systems are modular. Although a significant amount of transcription and translation machinery is common to all biochemical systems, this commonality does not invalidate the possibility that biochemical systems can operate somewhat independent of each other. Given the complexity of interactions present within the cell, any hope of untangling the intricacies of its operation will come through an assumption of modularity. Although design patterns will not be able to capture the richness of design provided by the creator, they will provide a means of understanding the reliable interactions of major subsystems of the cell.

As demonstrated by the application of design patterns to biochemical systems, there are multiple ways of implementing a particular pattern. This flexibility allows components of similar function to compensate for each other when necessary. It also allows for significant interchange of modules when conditions of the environment change. In light of this flexibility, restoring lost function is a process of exploring possible implementations of a design pattern deemed necessary for the system to operate well. Residual components of the lost function direct the researcher to a particular implementation. The implementation then provides clues to identifying missing or deteriorated components. These components may be genes or regulatory elements within the DNA. At this point intervention would follow standard medical practices of therapy.

In theory this process looks simple; however, in practice biomedical research is a costly and time consuming process. The loss of function due to mutation may not have an immediate effect due to compensating systems. However, the cumulative effect of multiple mutations will eventually manifest itself in the form of genetic diseases and cancers (Kim, 2007). Some molecular networks have a global impact on the dynamics of a cell. To achieve stable and robust function, many processes flow into a limited number of checkpoints, which control such processes as the cell cycle (Li, Long, Lu, Ouyang, & Tang, 2004). Mutations to p53, a transcription factor tied to cell cycle regulation, can lead to tumor development and restoration of its function can prove a useful therapy (Ventura et al., 2007). Proteins of the Ras family serve as important switches in signal pathways and mutations to this protein are found in a number of different tumor types (Bos, 1989). Cancer cells with this mutation are susceptible to reoviruses and this provides the basis for a viral based tumor therapy (Coffey, Strong, Forsyth, & Lee, 1998; Kim, Chung, & Johnston, 2007). For mutations that are more distant from critical pathways, it will be necessary to reverse engineer the biological system with all of its complexity (Csete & Doyle, 2002). Systems biologists are making progress in this area and the discovered network structure is providing insights on disease (Zhu, Gerstein, & Snyder, 2007). The role of design patterns in this process is to provide insights on cellular interactions beyond those provided by network motifs and based on design principles. Analysis of this type applied to functions related to p53 and Ras are beyond the scope of this paper and provide a direction for future research.

Galactose metabolism

Returning to the galactose metabolism example, it is possible to observe the extent of genetic change

when a system is lost. Hittinger et al. (2004, p. 14146) compared eleven yeasts species, four of which can not metabolize galactose. While three of these species are missing the genes for galactose metabolism, *S. kudriavzevii* maintains pseudogenes corresponding to *GAL7*, *GAL10*, and *GAL1*, which are contiguously located on chromosome II of *S. cerevisiae*. Assuming the genes and intergenic regions related to galactose metabolism in *S. cerevisiae* are relatively stable, it is possible to determine how the associated genes in *S. kudriavzevii* have deteriorated since this function was lost. Obtaining data from GenBank for both *S. cerevisiae* and *S. kudriavzevii*, a comparison was made between segments of these two genomes using the program PipMaker (Benson, Karsch-Mizrachi, Lipman, Ostell, & Wheeler, 2007; Schwartz et al., 2000). Figure 8A clearly illustrates how this segment of the genome for *S. kudriavzevii* has changed over time. The intergenic region preceding *GAL7* has been deleted, as well as, a segment in the interior of *GAL7*. Two other intergenic regions have been deleted along with a significant portion of *GAL10*. Half of *GAL1* and its subsequent intergenic region are still present. Although not visible in this figure, there are a number of point deletions resulting in frame-shift errors.

Choosing a segment of the *GAL10* gene, it is possible to estimate the number of point mutations that have occurred since this function was lost. Figure 8B illustrates the six significant segments analyzed by Pipmaker. Except in the last segment, breaks between segments are due to frame-shift errors. The longest segment is 297 base pairs long and matches *S. cerevisiae* 69% of the time. Although this represents 92 matching errors, this percentage corresponds to about 110 mutation events, assuming all of these errors are due to random substitutions. Although the number of mutations is great since the divergence of *S. cerevisiae* and *S. kudriavzevii*, there is hope that fewer mutations would be present in more complex eukaryotes due to the average cell cycle time for gametes being much larger than the two hours for yeast (Alon, 2007, p. 6).

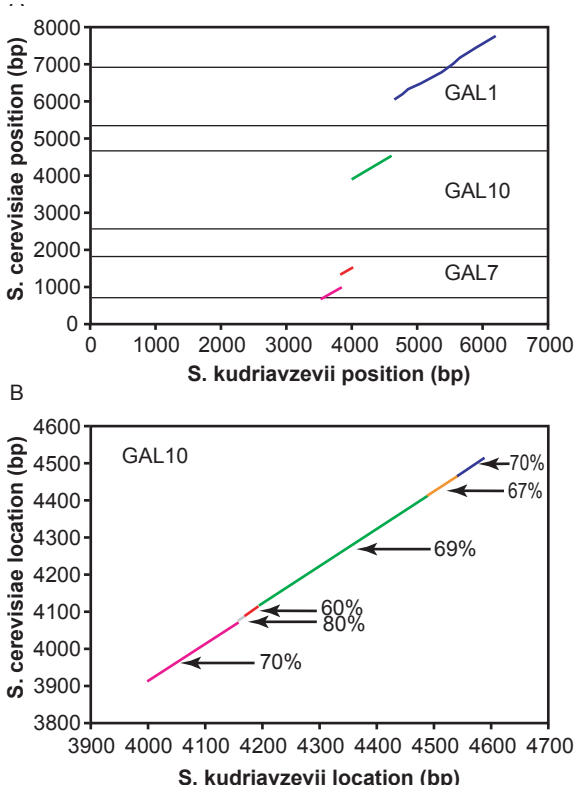

Figure 8. A comparison between gene sequences of *S. cerevisiae* and *S. kudriavzevii* (A) illustrate the deterioration that has occurred since galactose metabolism has been lost in *S. kudriavzevii*. The percentage difference in the *GAL10* gene (B) demonstrates the amount of loss due to point mutations.

Conclusion

An extended definition of good design in biochemical systems involves the use of OOP concepts to determine orderly patterns of interaction between biochemical and regulatory components of the cell. By comparing biochemical systems to known design patterns, it is concluded that these design principles are at least analogous between cellular biology and computer science, if not homologous. By applying design patterns to a specific system, it is possible to envision a greater array of possible implementations of the studied system.

Design patterns as a heuristic provides benefits to both bio-research and bio-engineering. Just as protein function can be identified by comparison with homologous proteins, system components and function can be identified by comparison with homologous systems. Using a design pattern approach could provide a means for identifying protein function in the context of a system and for identifying and potentially restoring system deterioration. OOP design principles applied to biochemical systems will help bio-engineers develop a wider variety of interoperating components and assemble these components into patterns that will minimize adverse interactions when introduced into the cell. Although biologists already use similar approaches, a design pattern language can serve to refine systems level ideas and provide a common language for communicating these ideas.

On the philosophical side, the presence of design patterns in biochemical systems gives some evidence that a top-down approach to design is present in the cell. This begs the question whether OOP principles of abstraction can arise through natural, self-ordering principles. Paley's view of design was rejected because it did not capture the dynamics of organic systems. However, by extending the design paradigm to include the OOP view of good design, it

is possible to view good design that provides a degree of variability and adaptability that was inconceivable in the classical view of design. This extended view of design may provide answers for understanding the plasticity of organisms during development and the rapid adaptation of organisms in the wake of the global flood (Marsh, 1983; Wood, 2003).

Bill Joy (1999), originator of the Java computer language, made the following statement during a panel discussion at the JavaOneSM Conference:

> Systems that are based totally on mechanical principles, Newtonian thinking, tend to be very brittle. If something is slightly the wrong dimension, it tends to break. Biological systems tend to have different properties. … we have to look to a different intellectual tradition, not just to mechanical engineering and physics, but to biology and the evolution of natural systems, what we call complex adaptive systems, going forward.

Although he states that these systems emerge in nature, his recognition for an extended view of design is clear. How much more should the creationist community value an extended view of design, knowing that through the choice of the Creator; the intricate, robust, and adaptable biological world came into being! Computer science has much to learn from the biological world; however, this is a two-way street. By defining a common language of design for biology as well as computer science, both disciplines will benefit.

Acknowledgments

Thanks go to the Fieldstead Institute who provided a grant for the Calvin College Seminars in Christian Scholarship. It was at the *Design, Self-Organization, and the Integrity of Creation* seminar lead by Dr. William Dembski that ideas for this paper began to form. Additional thanks go to the Creation Biology Study Group (BSG), who provided feedback and encouragement while these ideas were in their formative stages. The analytic work for this paper would not have been possible without data and information supplied by GenBank http://www.ncbi.nlm.nih.gov/ and the Saccharomyces Genome Database http://www.yeastgenome.org/. Genome comparisons were done by Pipmaker at the address http://pipmaker.bx.psu.edu/pipmaker/.

References

Alon, U. (2007). *An introduction to systems biology: Design principles of biological circuits* (1st ed.). Chapman and Hall/CRC.

Anderson, J.C., Wu, N., Santoro, S.W., Lakshman, V., King, D.S., & Schultz, P.G. (2004). An expanded genetic code with a functional quadruplet codon. *Proceedings of the National Academy of Sciences, 101*(20), 7566–7571.

Ast, G. (2005). The alternative genome. *Scientific American, 292*(4), 58–65.

Babaoglu, O., Canright, G., Deutsch, A., Di Caro, G., Ducatelle, F., Gambardella, L., Ganguly, N., Jelasity, M, Montemanni, R., & Montresor, A. (2006). Design patterns from biology for distributed computing. *Association for Computing Machinery Transactions on Autonomous and Adaptive Systems, 1*(1), 26–66.

Behe, M.J. (1996). *Darwin's black box: The biochemical challenge to evolution.* New York: Free Press.

Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., & Wheeler, D.L. (2007). GenBank. *Nucleic Acids Research, 35*(Database issue), D21–D25.

Bertalanffy, L. (1968). *General systems theory: foundations, development, applications.* New York, New York: George Braziller.

Bhat, P.J., & Murthy, T.V.S. (2001). Transcriptional control of the *GAL/MEL* regulon of yeast *Saccharomyces cerevisiae*: mechanism of galactose-mediated signal transduction. *Mol. Microbiology, 40*(5), 1059–1066.

Block, S.M. (2001). The growing threat of biological weapons. *American Scientist, 89*(1), 28–37.

Bos, J.L. (1989). *ras* oncogenes in human cancer: A review. *Cancer Research, 49*(17), 4682–4689.

Carroll, S.B. (2005). *Endless forms most beautiful: The new science of evo devo and the making of the animal kingdom* (1st ed.). New York: Norton.

Cherry, J.M., Adler, C., Ball, C., Chervitz, S.A., Dwight, S.S., Hester, E.T., Jia, Y., Juvik, G., Roe, T.Y., Schroeder, M., Weng, S., & Botstein, D. (1998). SGD: *Saccharomyces* genome database. *Nucleic Acids Research, 26*(1), 73–79.

Chopra, P., & Kamma, A. (2006). Engineering life through synthetic biology. *In Silico Biology, 6*(5), 401–410.

Coffey, M.C., Strong, J.E., Forsyth, P.A., & Lee, P.W. (1998). Reovirus therapy of tumors with activated Ras pathway. *Science, 282*(5392), 1332–1334.

Cooper, G.M. (1997). *The cell :A molecular approach* (1st ed.). Washington, D.C.: ASM Press.

Cordero, O.X., & Hogeweg, P. (2006). Feed-forward loop circuits as a side effect of genome evolution. *Molecular Biology and Evolution, 23*(10), 1931–1936.

Csete, M.E., & Doyle, J.C. (2002). Reverse engineering of biological complexity. *Science, 295*(5560), 1664–1669.

de Atauri, P., Orrell, D., Ramsey, S., & Bolouri, H. (2004). Evolution of "design" principles in biochemical networks. *Systems Biology, 1*(1), 28–40.

Dembski, W.A. (1998). *The design inference: Eliminating chance through small probabilities.* Cambridge, New York: Cambridge University Press.

Dembski, W.A. (2001). *In intelligent design a form of natural theology?* Retrieved June, 2007, from http://www.designinference.com/documents/2001.03.ID_as_nat_theol.htm

Emerald, B.S., & Cohen, S.M. (2004). Spatial and temporal regulation of the homeotic selector gene *Antennapedia* is required for the establishment of leg identity in *Drosophila*. *Developmental Biology, 267*(2), 462–472.

Feldmann, H. (2000). Gene function and expression: Four years of the post-genomic era of yeast. *Food Technology and Biotechnology, 38*(4), 237–252.

Feldmann, H. (2005). *Yeast molecular biology: A short compendium on basic features and novel aspects (chapter*

4: Yeast molecular techniques). Retrieved June, 2007, from http://biochemie.web.med.uni-muenchen.de/Yeast_Biol/04%20Yeast%20Molecular%20Techniques.pdf

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (c1995). In Gamma E. (Ed.), *Design patterns: Elements of reusable object-oriented software*. Reading, Massachusetts: Addison-Wesley.

Gibbs, W.W. (2004). Synthetic life. *Scientific American, 290*, 74–81.

Hartwell, L.H., Hopfield, J.J., Leibler, S., & Murray, W.M. (1999). From molecular to modular cell biology. *Nature, 402*(S2), C47–C52.

Herrgard, M.J., Covert, M.W., & Palsson B.O. (2003). Reconciling gene expression data with known genome-scale regulatory network structures. *Genome Research, 13*(11), 2423–2434.

Hieronymus, H., & Silver, P.A. (2004). A systems view of mRNP biology. *Genes & Development, 18*(23), 2845–2860.

Hittinger, C.T., Rokas, A., & Carroll, S.B. (2004). Parallel inactivation of multiple GAL pathway genes and ecological diversification in yeasts. *Proceedings of the National Academy of Sciences, 101*(39), 14144–14149.

Hodges, P.E., McKee, A.H.Z., Davis, B.P., Payne, W.E., & Garrels, J.I. (1999). The yeast proteome database (YPD): A model for the organization and presentation of genome-wide functional data. *Nucleic Acids Research, 27*(1), 69–73.

Institute for Systems Biology. (2006). *Systems biology—the 21st century science.* Retrieved June, 2007, from http://www.systemsbiology.org/Intro_to_ISB_and_Systems_Biology/Systems_Biology_--_the_21st_Century_Science

Johnston, R.J., Chang, S., Etchberger, J.F., Ortiz, C.O., & Hobert, O. (2005). MicroRNAs acting in a double-negative feedback loop to control a neuronal cell fate decision. *Proceedings of the National Academy of Sciences, 102*(35), 12449–12454.

Joy, B. (1999). What's ahead for the Java language and Java technology? *Panel discussion JavaOneSM Conference.*

Joy, B. (2000). Why the future doesn't need us. *Wired, 8*(4), 1–11.

Kashtan, N., & Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences, 102*(39), 13773–13778.

Kim, L. (2007). Accumulation of mutations: cancer or molecule-to-man evolution? *Journal of Creation, 21*(2), 77–81.

Kim, M., Chung, Y.H., & Johnston, R.N. (2007). Reovirus and tumor oncolysis. *Journal of Microbiology, 45*(3), 187–192.

Li, F., Long, T., Lu, Y., Ouyang, Q., & Tang, C. (2004). The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences, 101*(14), 4781–4786.

Liu, X., & Clarke, N.D. (2002). Rationalization of gene regulation by a eukaryotic transcription factor: calculation of regulatory region occupancy from predicted binding affinities. *Journal of Molecular Biology, 323*(1), 1–8.

Lopez, P.J., & Seraphin, B. (2000). YIDB: The yeast intron database. *Nucleic Acids Research, 28*(1), 85–86.

Mak, H.C., Daly, M., Gruebel, B., & Ideker, T. (2007). CellCircuits: A database of protein network models. *Nucleic Acids Research, 35*(Database issue), D538–D545.

Mangan, S., & Alon, U. (2003). Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences, 100*(21), 11980–11985.

Marsh, F.L. (1983). Genetic variation, limitless or limited? *Creation Research Society Quarterly, 19*(4), 204–206.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., & Alon, U. (2002). Network Motifs: simple building blocks of complex networks. *Science, 298*, 824–827.

Mundodi, V., Kucknoor, A.S., & Gedamu, L. (2005). Role of Leishmania chagasi amastigote cysteine protease in intracellular parasite survival: Studies by gene disruption and antissense mRNA inhibition. *BioMed Central (BMC) Molecular Biology, 6*(3).

The new king james bible: New testament (1979). Nashville: T. Nelson.

Orrell, D., & Bolouri, H. (2004). Control of internal and external noise in genetic regulatory networks. *Journal of Theoretical Biology, 230*(3), 301–312.

Paley, W. (1743–1805). ([1850?]). *Natural theology/by William Paley* (From a late London edition ed.). New York :American Tract Society,.

Powell, K. (2004). All systems go. *Journal of Cell Biology, 165*(3), 299–303.

Pritsker, M., Liu, Y.C., Beer, M.A., & Tavazoie, S. (2004). Whole-genome discovery of transcription factor binding sites by network-level conversation. *Genome Research, 14*(1), 1–9.

Ravasz, E., Somera, A.L., Mongru, D.A., Oltvai, Z.N., & Barabasi, A.L. (2002). Hierarchical organization of modularity in metabolic networks. *Science, 297*(5586), 1551–1555.

Ray, J. (1979). *The wisdom of god manifested in the works of the creation:1691.* New York: Garland Pub.

Rives, A.W., & Galitski, T. (2003). Modular organization of cellular networks. *Proceedings of the National Academy of Sciences, 100*(3), 1128–1133.

Rønnow, B., Olsson, L., Nielson, J., & Mikkelsen, J.D. (1999). Derepression of galactose metabolism in melibiase producing baker's and distillers' yeast. *Journal of Biotechnology, 72*(3), 213–228.

Ruse, M. (1999). *The Darwinian revolution: Science red in tooth and claw.* Chicago, Illinois: University of Chicago Press.

Saccharomyces Genome Database. (2007). *Saccharomyces cerevisiae genome Snapshot/Overview.* Retrieved June, 2007, from http://www.yeastgenome.org/cache/genomeSnapshot.html

Schwartz, S., Zhang, Z., Frazer, K.A., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R., & Miller, W. (2000). PipMaker—a web server for aligning two genomic DNA sequences. *Genome Research, 10*(4), 577–586.

Shalloway, A., & Trott, J. (2005). *Design patterns explained: A new perspective on object-oriented design* (2nd ed.). Boston, Massachusetts: Addison-Wesley.

Skyttner, L. (1996). *General systems theory: An introduction.* Basingstoke: Macmillan Press.

Snel, B., & Huynen, M.A. (2004). Quantifying modularity in the evolution of biomolecular systems. *Genome Research, 14*, 391–397.

Travern, A., Jelicic, B. & Sopta, M. (2006). Yeast Gal4: A transcriptional paradigm revisited. *EMBO Reports, 7*(5), 496–499.

Turner, J.M., Graziano, J., Spraggon, G., & Schultz, P.G. (2006). Structural plasticity of an aminoacyl-tRNA synthetase active site. *Proceedings of the National Academy*

of Sciences, *103*(17), 6483–6488.

Vazquez, A., Dobrin, R., Sergi, D., Eckmann, J.P., Oltvai, Z. N., & Barabasi, A.L. (2004). The topological relationship between the large-scale attributes and local interaction patterns of complex networks. *Proceedings of the National Academy of Sciences*, *101*(52), 17940–17945.

Velculescu, V., Zhang, L., Zhou, W., Vogelstein, J., Basrai, M. A., Bassett, D.E., Hieter, P., Vogelstein, B., & Kinzler, K.W. (1997). Characterization of the yeast transcriptome. *Cell, 88*(2), 243–251.

Ventura, A., Kirsch, D.G., McLaughlin, M.E., Tuveson, D.A., Grimm, J., Lintault, L., Newman, J., Reczek, E.E., Weissleder, R., & Jacks, T. (2007). Restoration of p53 function leads to tumour regression in vivo. *Nature, 445*(7128), 661–665.

Waters, B. (2006) *From human to posthuman: Christian theology and technology in a postmodern world*. Burlington, Vermont: Ashgate Publishing Ltd.

Watson, J.D., Baker, T.A., Bell, S.P., Gann, A., Levine, M., & Losick, R. (2004). *Molecular biology of the gene* (5th ed.). San Francisco: Pearson/Benjamin Cummings.

Wood, T.C. (2003). Perspectives on ageing, a young-earth creation diversification model. In R.L. Ivey Jr. (Ed.), *Proceedings of the fifth international conference on creationism* (pp.479–489). Pittsburgh, Pennsylvania: Creation Science Fellowship.

Wood, T.C., Wise, K.P., Sanders, R., & Doran, N. (2003). A refined baramin concept. *Occasional Papers of the Biology Study Group, 3*, 1–14.

Yeger-Lotem, E., Sattath, S., Kashtan, N., Itzkovitz, S., Milo, R., Pinter, R.Y., Alon, U., & Margalit, H. (2004). Network motifs in integrated cellular networks of transcription-regulation and protein-protein interaction. *Proceedings of the National Academy of Sciences, 101*(16), 5934–5939.

Zhu, X., Gerstein, M., & Snyder, M. (2007). Getting connected: Analysis and principles of biological networks. *Genes & Development, 21,* 1010–1024.