

Engineering Approaches Toward Biological Information Integration at the Systems Level

Wenjin J. Zheng*

Department of Biostatistics, Bioinformatics & Epidemiology, and Bioinformatics Core Facility, Hollings Cancer Center, Medical University of South Carolina, 135 Cannon Street, P.O. Box 250835, Charleston, SC 29425, USA

Abstract: Our understanding of biological systems has improved dramatically due to decades of exploration. This process has been accelerated even further during the past ten years, mainly due to the genome projects, new technologies such as microarray, and developments in proteomics. These advances have generated huge amounts of data describing biological systems from different aspects. Still, integrating this knowledge to reconstruct a biological system *in silico* has been a significant challenge for biologists, computer scientists, engineers, mathematicians and statisticians. Engineering approaches toward integrating biological information can provide many advantages and capture both the static and dynamic information of a biological system. Methodologies, documentation and project management from the engineering field can be applied. This paper discusses the process, knowledge representation and project management involved in engineering approaches used for biological information integration, mainly using software engineering as an example. Developing efficient courses to educate students to meet the demands of this interdisciplinary approach will also be discussed.

Keywords: Biological information integration, engineering, process, object-oriented.

INTRODUCTION

Biological systems are extremely complex. For example, a yeast cell has over 6000 genes [1] encoding protein or RNA molecules. These gene products have many different functions and form complexes and sub-cellular structures. Cellular functions are regulated at different levels through transcription, signal transduction and protein modification [2-7]. To understand the complexity of a biological system, a reductionist approach is taken to study its organization, components and the interrelationships among these components: information about genes and their structures are obtained by genome sequencing and analysis [8-10]; gene regulation is studied by microarray analysis [11]; and protein functions are studied by proteomics [12], protein-protein interactions [13-16] and structure analysis [15, 17-32]. The huge amount of data generated by these new technologies needs to be analyzed and integrated to depict comprehensively how biological systems work. However, integrating this knowledge together systematically poses a significant challenge, as the size and the complexity of the data grow exponentially. To meet this challenge, new methodologies have to be developed to integrate and model complex data at the system level [33-38]. Integrating this information to model the complexity of biological systems has been the focus of a major effort in computational biology and bioinformatics.

Many computational methods have been developed to analyze and integrate biological information. Centralized and distributed databases have been developed to capture information about sequences and functions, signaling and

metabolic pathways, and protein structure information [39]. Various modeling diagrams have been used to represent biological information at high levels of abstraction [13, 40-47]. Similarly, software has been developed to analyze, manage and model biological information [48]. To capture, organize and communicate this information, markup languages have also been developed [49-51]. In addition, methodologies have been developed to model and simulate biological systems at different levels [52-70]. While these efforts have greatly facilitated biological information analysis and presentation, the wealth of information gathered through new technologies, such as genome sequencing, microarray and proteomics, still presents a formidable challenge: integrating all the information at the system level [71].

Several studies have compared engineering systems with biological systems. In one such study, Csete and Doyle [2] concluded that there is a converged evolution between these two types of system at the organizational level. Both types of systems are modular with defined protocols for robustness. Engineered control systems were also compared to biological systems by Hasty *et al.* [72]. Focusing on gene regulation, this study has found many similarities between biological and control systems. Circuit engineering principles have also been applied to biological system analysis [73, 74]. Another study has compared computer engineering systems to biological ones [75]. In this analysis, the genome is likened to the hard drive (permanent storage) of a computer that stores both system information and data. Messenger RNAs are considered the memory of a computer, which can temporarily store information from the genome or hard drive. Calculations are realized through protein functions. Biological systems have been further compared to software engineering systems, specifically object-oriented software architecture [76]. Bioentities, such as proteins, are analogous to objects, both of which have attributes and

*Address correspondence to this author at the Department of Biostatistics, Bioinformatics & Epidemiology, Medical University of South Carolina, 135 Cannon Street, Charleston, SC 29425, USA; Tel: (843) 876-1123; Fax: (843) 876-1126; E-mail: zhengw@musc.edu

functions. Relationships among bioentities can be described in the same way as for objects, such as through composition. All these comparisons indicate that biological systems and modern engineering systems share many features. These commonalities suggest that: 1) a biological system can be viewed as an engineering system, 2) biological information integration can be approached by reverse engineering, and 3) biological systems can be represented in the form of modern engineering system architectures. While biological systems can be integrated and represented in the form of various engineering systems, any given representation captures different aspects of the biological system. A well-developed engineering approach, one that includes an engineering process, diagrams, documentation methods, integration and design methodologies, as well as engineering architectures, may be used to describe, define, represent or analyze biological systems. Many of these topics have been discussed elsewhere, and there has been significant work on reverse engineering biological information [2, 4, 33, 34, 77-80]. This review examines several issues not widely discussed: process, knowledge representation and project management associated with representing biological systems using an engineering approach, with examples mainly from the software engineering field. The need for cross-disciplinary education and an efficient curriculum related to this topic will be also discussed.

ENGINEERING PROCESS

Integrating biological information as an engineering system is a process of reverse engineering. Reverse engineering is "...the process of analyzing a subject system to identify the system's components and their interrelationships, and to create representations of the system in another form at higher levels of abstraction" [81]. One major use of reverse engineering is in the software industry, where software engineers implement software systems based on existing ones without knowing their source code and architecture. Information about the existing system comes primarily from observable system behaviors. Specifications are generated based largely on what the system does rather than how it does it. From these specifications, serial steps are taken to analyze and design the new system, and eventually implement a new software system to imitate the old system's behavior. Any available knowledge about the internal structure of the old system is used to constrain the design and implementation of the new one. As a result, the new system architecture can reflect the structure, functions and dynamics of the original.

This reverse engineering process yields a model or representation of the old system in a new form. However, unlike in the field of engineering, no well-defined process exists for reverse engineering biological systems, despite intense and varied investigations [2, 82-90]. A recent study has shown that a software engineering process can be applied to reverse engineer a biological system [76]. In this study, a model biological system, the SARS viral infection, is viewed as an object-oriented software engineering system without design documents, blueprints or source code. During the reverse engineering process, the biological system is the subject system, and its components are gene products. The interrelationships of these gene products are protein-protein interaction maps, the transcription co-regulation of these

genes, pathway relationships, Gene Ontology [91] relationships, and other interactions among cellular components obtained through experiments or data analysis. These experimental data provide the specifications, internal structures and dynamics of the system. Reverse engineering the biological system creates a model of the system in another form, an object-oriented software architecture, at a higher level of abstraction. Through the model, the processes and the dynamics of the biological system are captured, and the actions of one gene product upon another are also modeled. Because a model integrates all the available information about a system, it provides a comprehensive view of how the biological system works [76].

In developing any engineering project, the engineering process employed is crucial, in that it defines sequential steps of engineering system construction. For each step, the input, the construction, the outcome and the roles personnel will play are well-defined. Following a well-defined engineering process is essential to a successful engineering project, especially in the construction of large, complex systems. In the software engineering field, the process is a series of well-defined steps to construct software [92]. For the past several decades, this process has evolved from linear, sequential models to incremental, iterative ones. Several major software processes have been developed: 1) Linear, sequential model – the most widely used, oldest model. It demands a rigid sequential approach involving the following steps: Analysis, Design, Code and Test. Many modern software engineering processes evolved from this model [93]. 2) Incremental Model – develops a product with partial functionality at each release. A core product with essential functionality is developed first. Incrementally, a portion of function is added to the core products and a partially functional product is delivered. The process is continued until a fully functional final product is delivered [94]. 3) Extreme Programming – a new agile process that expedites software development by obtaining continuous feedback from short cycle development, flexible scheduling, close collaboration of programmers and rigorous automatic testing [95]. 4) Cleanroom engineering – applies formal mathematical notations to develop defect-free software. This approach helps eliminate problems such as ambiguity, incompleteness and inconsistency that other software processes encounter. However, the approach requires extensive training for software engineers and a significant initial effort to develop the formal model [96]. 5) The Rational Unified Process (RUP) [97] – developed by the Rational Corporation (now a subdivision of IBM), is an iterative process defined by four phases of software development. In each phase, – Inception, Elaboration, Construction and Transition – activities and their efforts are specified [98]. Considered one of the most advanced processes, it is being increasingly adopted by the software industry.

The software engineering process is important for reverse engineering biological systems into a software system representation because: 1) a good iterative software process can break down complex biological systems into manageable parts and define how reverse engineering can be practiced on each part; and 2) a software process defines different stages for reverse engineering, and with the interdisciplinary nature of reverse engineering biological systems, it can help to

specify the critical roles that biologists and computer scientists play at different stages. Software processes' strengths and weaknesses differ and are therefore particularly suited to differing software projects [99]. For small projects, a linear, sequential model is typically ideal. For larger ones, depending on the nature of the project, the incremental or an RUP model is better suited. Software engineers might likewise determine that either Cleanroom engineering or Extreme Programming approaches would best suit a project, but neither is used extensively in the software industry.

Among these methods, RUP is considered the latest development and has been adopted by many in the software industry. Its iterative, requirement-driven nature and architecture-based approach make it suitable for large, complex software system engineering. Its popularity is also an advantage because engineers with RUP experience are easy to find. Moreover, developed by a software engineering tool company, RUP can be tightly coupled with the Unified Modeling Language (UML) [100], making it the best candidate for Computer Aided Software Engineering (CASE) [101, 102]. These advantages make RUP the best choice to reverse engineer complex biological systems into an object-oriented software representation. By following such a software process, the activities at each stage of the process are defined, and the outcome of each stage is specified. Other engineering architecture representations (e.g. a control system) of biological systems can also be developed. Such development may also require well-developed engineering processes. On the other hand, while a well-developed engineering process can be applied to reverse engineer a biological system, one major obstacle may arise from the unfamiliarity of biologists with these processes. Adapting and making them easier to understand and use by biologists may be necessary. One possible solution is to develop intuitive tools with graphical user interfaces to guide the biologists and engineers to work together and follow the modified processes for reverse engineering.

The outcomes of these engineering processes yield the engineering system representations of biological information in the form of documents and models. Together, these biological representations should be succinct, precise and comprehensive, and improve our understanding of biological systems.

ENGINEERING SYSTEM REPRESENTATIONS

Knowledge representation is an important aspect for information communication, especially for complex systems, such as biological ones, and interdisciplinary projects that require both biologist and computer scientist involvement. Therefore, representations of biological systems have been studied, and many forms of knowledge representation have been developed. The Gene Ontology (GO) [91] is one such system developed to represent the hierarchical relationships of biological functions. Another, the Systems Biology Markup Language (SBML) was developed to capture the biological system information in an Extensible Markup Language (XML) format [49]. A variety of graphic diagrammatic representations have been developed for biological system information representation [42, 43, 45, 103-106]. In addition, graphical displays of pathway databases have been developed to represent metabolic and

signal transduction pathways [105, 107-111]. Several efforts have been developed to adapt Petri net representations for biological systems [43, 45, 104, 105, 112-120]. However, representing a biological system as an engineering system requires adopting engineering system representation methods. For example, in representing a biological system as a control system, a set of differential equations are used [2]. To represent a biological system as an object-oriented software system, the Unified Modeling Language (UML) is used [76, 102, 121, 122].

UML was developed by combining three different methodologies for object-oriented software development [123]. UML provides several diagrammatic representations to capture system information. For instance, a class diagram is used to describe components and their relationships [124]. Sequence and collaboration diagrams are used to describe the dynamics of the system [125, 126]. Activity diagrams and state charts describe the transitions occurring in a system [127, 128]. Packages divide the complex systems into small, manageable subsystems [129]. The comprehensiveness of UML makes it a good candidate to represent complex biological systems. Moreover, well-defined software processes, such as RUP, can be used to define the transition from system requirements to UML artifacts. Implementing software in an object-oriented programming language based on UML documents is already a well defined process. There are even software tools available to convert UML to source code automatically [130]. Nevertheless, limitations exist in applying UML to biological system modeling.

UML was not developed specifically for biological system modeling. Significant effort will be required to adapt UML to biological system modeling. Still, UML is designed to represent an object-oriented software system in general, and biological systems are organizationally similar to object-oriented software systems. This similarity should facilitate transformation of biological information into an object-oriented representation.

A second hurdle to overcome is that UML employs many diagrams to capture different aspects of a system. Even though this divide and conquer approach is the means by which object-oriented technology handles complexity, Peleg and Dori [131] have pointed out that this multi-model aspect could be a source of confusion. Users with different backgrounds and levels of skill have to combine these models together to get a comprehensive view of the system. This potential confusion can be solved by two approaches. First, a well-defined system architecture can organize information into comprehensible units. Second, proper training of the users of these models can help to depict the roles of these models in the context of the whole system. A demonstration of how to apply UML to the representation of a biological system can increase the exposure of UML to the biology community and facilitate communication between biologists and computer scientists [76].

Different forms of representation may be suitable for different users and capture different features. For example, mathematicians or electrical engineers can represent a biological system using a set of differential equations to capture the dynamics of the system. These differential equations can be represented in SBML. On the other hand, tools have been developed to allow biologists using visual

diagrams to develop biological models. These visual models can then be transformed into SBML models [132]. UML has also been used during the development of SBML. Such transformations indicate that a biological model can be represented in different formats for different purposes. However, well-defined and widely accepted modeling languages in the engineering field are essential for communicating biological models and facilitating computational analysis. Such engineering system representations are especially important because of the growing complexity of biological information integration at the systems level [133].

For biological science, the potential impact of engineering system representation is significant. Unlike a database repository, an engineering system representation captures not only the descriptive annotations of the components and static structures of a biological system but also the dynamic relationships among the components and the responsiveness of the systems to environmental changes. Thus, an engineering system can be modeled to simulate the dynamics and behavior of the system under conditions that are difficult to replicate experimentally. Representing biological information in any engineering system architecture is important for us to better understand the biological system. Davis *et al.* [134] have surveyed the roles that knowledge representations play. They concluded that a knowledge representation is: 1) a surrogate, 2) a set of ontological commitments, 3) a fragmentary theory of intelligent reasoning, 4) a medium for efficient computation, and 5) a medium of human expression. Compared to other types of biological knowledge representations, the engineering system representation offers significant advantages in playing these roles and can further improve our understanding of biological systems.

Engineering representations as surrogates. The representation of a biological system in any engineering system architecture is a surrogate that captures different aspects of the biological system. For example, in representing a biological system as a control system, the representation captures the dynamics and the control aspects of the biological system. In that regard, the system behaves just like the original biological system it represents. This close resemblance makes the new representation a surrogate to model and simulate the original biological system. In representing a biological system as an object-oriented software system, an enzyme is modeled as an object. This object captures both the attributes and the functions of the original enzyme, thus serving as a digital surrogate for the enzyme. Likewise, an object-oriented software representation of a more complex biological system also serves as a digital surrogate. Given the complexity of biological systems, one is unlikely to construct a surrogate that resembles the whole biological system in every aspect. However, different surrogates can be constructed to capture various aspects of a single biological system.

Engineering representations as ontology commitments. Ontology is defined as a specification of a conceptualization, an explicit specification of some topic and a formal and declarative representation of some subject area [135]. Each well developed engineering field has its own well-defined terms to describe and represent its domain of

knowledge. Thus, explicitly or implicitly, each engineering domain develops its own ontology. In representing a biological system as an engineering system, the biological system is viewed through the lens of that engineering system. In other words, a biological system would be described using the ontology developed for engineering systems. Davis *et al.* described this ontology commitment as "...in what terms should I think about the world?"[134]. While representing a biological system as an object-oriented software system, bioentities in the biological systems are named "objects", with the properties of these bioentities captured as the attributes of corresponding objects. The functions these bioentities can perform are captured as object member functions. Such an ontological commitment is necessary, in that the well-developed engineering philosophy and methodology can be readily used to analyze and construct the engineering system representation from biological information using a well-developed, controlled vocabulary. This vocabulary has precise unambiguous meaning within an engineering system, and can greatly facilitate communication among engineers and other users who can understand such terms.

Engineering representations as theories of intelligent reasoning. To reason is to infer, to get new information or new expressions from old. Influenced by mathematics, to reason intelligently is to reason logically, in the form of inferences such as Prolog [136]. However, such intelligent reasoning is also influenced by other fields such as psychology, biology, statistics and economics. Thus, other forms of inferences, such as connectionism, causal networks or rational agents can also be viewed as intelligent reasoning [134]. When representing biological information in the form of engineering systems, there are three stages of intelligent reasoning. First, constructing an engineering representation of a biological system from existing knowledge is a process of intelligent reasoning. To represent a biological system as an engineering control system, mathematical logics can be derived from experimental results to represent the system as a set of differential equations. Inferences have to be made from existing knowledge to represent bioentities as objects when representing a biological system as an object-oriented software system. Second, an engineering system representation of a biological system should play roles in its analysis. Well-developed methods to analyze engineering systems can be used to study the structure, the robustness and the complexity of the biological system in its engineering system representation [4, 73, 137]. For example, the methods already widely used to study the complexity of an object-oriented software system can be used to study the object-oriented representations of biological systems. Third, the engineering system representation can be implemented and simulations can be performed to study the behavior of the biological system it represents. For example, while representing a biological system as a controlled system with a set of differential equations, the behavior of the system under different conditions can be inferred [2, 138, 139]. This approach is particularly important, in that many experiments cannot be performed in the lab to study the biological system, and a computational simulation can overcome these obstacles, generating otherwise unavailable results. While these inferences can be made from an engineering representation of a biological system, some limitations may

exist. For example, object-oriented software systems are not designed specifically for logical inference, so Prolog types of reasoning are not supported. Such a limitation may be overcome by additional implementation that adds such functionality to the system.

Engineering representations as media for efficient computation. An engineering system representation is used for computation. In representing a biological system as a controlled system with differential equations, software tools are available for computational simulation of the system. An object-oriented software representation of a biological system can be implemented as executables to simulate the biological system behavior. These representations in an engineering system form are constructed for efficient computational analysis.

Engineering representations as media for human communication. Modern engineering projects are collaborative efforts with extended developmental periods that involve many people with different skills and backgrounds. To accomplish the engineering tasks, many methods have been developed to facilitate communication among people involved across wide time scales and to maintain project continuity. These communication methods are implemented as standardized diagrams, documents and controlled vocabularies. In addition, the ontological commitment of the engineering approach also facilitates communication. As a result, while a software engineer may know very little about a biological system, that engineer will be able to understand an object-oriented representation of the biological system. This understanding can facilitate communication between biologists and computer scientists. Such representations are thereby effective media for human communication.

One obstacle for representing a biological system in engineering form is that such representations need to be understood and used by biologists. However, no widely accepted engineering representation has been developed for biologists. Fortunately, visual modeling languages such as UML are relatively intuitive and easy to learn. Recent work performed under proper guidance by a biologist without prior computer science training demonstrated that biologists of similar background can be trained in a fairly short period of time to apply UML for biological system modeling [76, 140, 141].

ENGINEERING PROJECT MANAGEMENT

Traditionally, research projects are conducted in individual labs that specialize in specific areas. However, biological information integration projects are different. They require collaboration among groups of people with differing backgrounds and skills. Such activities are regularly performed in the form of a consortium, or communities formed by voluntary groups. Thus, project management is particularly important to the success of the effort. Unfortunately, project management has not been a major focus of research for large-scale biological information integration. By treating biological information integration as a reverse engineering process, project management techniques developed in the corresponding engineering field can be applied. In software engineering, people, problems and processes are major focuses of project

management [142]. For example, contrary to conventional wisdom, it has been found that adding more people to a delayed software engineering project will further delay it. More manpower in a project is not necessarily an efficient way to finish the project [143]. Similar findings on project efficiency in software engineering can be applied to research projects involving biological information integration. In addition, to measure the progress and the success of software projects extensive work has been done to develop reliable software metrics [144, 145]. Developing metrics for bioinformation integration projects may improve project management.

While these developments are widely understood and applied in managing software engineering projects, adopting them to biological information integration may require some work. Unlike in the typical software engineering project, both biologists and computer scientists are involved in biological information integration projects. These two groups have widely different backgrounds, skills and technical languages. Scheduling projects so that each group can efficiently deploy divergent knowledge and skill is a major challenge. A second significant hurdle is developing effective communication methods for these two groups. Research in these areas can greatly improve the efficiency of biological information integration projects. Another important aspect of project management is cost estimation. In software engineering, sophisticated methods, such as COCOMO [146], have been developed to estimate the cost of a project. However, there is no consistent and accurate method to estimate the cost associated with large-scale integration projects. Most of the budgets associated with these projects are estimated empirically. Adopting methods from engineering may help to budget such projects and improve cost efficiency.

One important aspect of any engineering project management is coordination and change control. For example, in a software engineering project, a version control system such as CVS [147] is typically used by engineers to maintain documents and implement their various contributions. This CVS allows engineers to keep track of any changes made during the development process. In a large-scale biological information integration project, such a system is also necessary, in that not only are such projects complex with long durations and many changes, but they also involve people from different backgrounds, such as biology and engineering. Errors will likely occur due to miscommunication and misunderstanding. However, a CVS can be very helpful to track and correct such errors. Some successful projects on biological information integration, such as Gene Ontology (GO) and Systems Biology Markup Language (SBML), are using version control systems through SourceForge.net [148] to manage the projects, demonstrating that engineering methodologies can be applied to improve the manageability of biological information integration projects.

CHALLENGE

So far there is no well-defined process for biological information integration. Most activities in the area have been *ad hoc*, even though several projects and groups have attempted to define processes for information integration

[76, 149]. Such efforts have been small in scale and are not widely recognized. Taking a reverse engineering approach to integrate biological information, however, means that information integration projects can draw on the experiences and widely recognized successes of engineering fields. The advantages of these engineering processes include decades of accumulated efforts by engineers. Process advantages and disadvantages are well-known. Of course, these processes were designed for engineering projects and may not be suitable for biological information integration. Significant effort may be needed to modify these processes for biological information integration. Nevertheless, developing processes from scratch for biological information integration appears to be both difficult and unnecessary. Significant research may be needed, which may require a long period of trial and error before the processes become mature and widely adopted.

The cross-disciplinary nature of engineering approaches for biological information integration faces the major challenge of communication. Representing biological information in the form of an engineering architecture poses a challenge for biologists, since a basic knowledge of engineering systems is required. Several approaches may be developed to overcome such an obstacle. First, cross-training can help biologists to understand engineering documents, such as diagrams, and engineers to understand biological information. Second, integrated biological information in the form of an engineering architecture can be represented in a variety of forms. For example, a set of differential equations can be stored in SBML format, and represented as a graphic display that is intuitive to biologists using software tools such as CellDesignerTM [132]. These model transformations can greatly facilitate communication between biologists and computer scientists.

Even though training biologists to be engineers, or vice versa, is not realistic, a basic understanding of the engineering system used for biological information integration is essential. At a minimum, understanding the terminology and a model's basic ontological basis is necessary for efficient communication. For information integration, engineers must have a basic knowledge of the information to be reverse engineered. Educating engineers will increase the efficiency of their work by improving communication between biologists and engineers. The challenge for current graduate training in bioinformatics is to develop an efficient curriculum to teach both biology and computer science at the same time [150-160]. Many graduate programs have a curriculum that requires students to take a mixed course of biology and computer science. However, this approach greatly reduces the amount of knowledge a student can obtain through the program. For example, a typical M.S. program requires 33 to 36 credit hours. For bioinformatics graduate students, these credit hours are split between biology and computer science. While these students have a broad training in both biology and computer science, they do not receive the in-depth training in either field compared to students in either a biology or computer science program (18 versus 36 credit hours in biology or computer science). However, this deep understanding will be necessary to overcome obstacles in the challenging field of biological information integration. Developing an interdisciplinary curriculum manifested within classes is essential for students

to acquire sufficient training to take advantage of the emerging possibilities in biological information integration [150, 151, 153-158, 160-162]. For example, while teaching object-oriented programming classes, real world examples are used. The relationship among shape, point and circle are used to illustrate the inheritance concept. By using biological examples, such as metabolic pathways or gene regulation, an efficient course can be developed to teach biology and object-oriented programming at the same time. As a result, students need only three credit hours to learn both metabolic pathways and object-oriented programming.

In summary, well-developed methodologies for engineering system development can be used for biological information integration. This application can greatly improve the efficiency and the manageability of large-scale projects and tackle the complexity of biological systems. Representing biological systems as various engineering systems can facilitate communication between biologists and other engineering professions, and advance the computational analysis of biological systems.

ACKNOWLEDGEMENTS

The author would like to thank Drs. Daniel Shegogue and Tom Smith for their comments and proof reading of the manuscript, and apologize to the many authors of relevant publications that cannot be cited due to size limitations. W.J. Zheng is partly supported by DOE grant DE FG02-01ER63121 and DOD grant GC-3609-04-43766CM.

REFERENCES

- [1] Mewes HW, Albermann K, Bahr M, *et al.* Overview of the yeast genome. *Nature* **1997**; 387(6632 Suppl):7-65.
- [2] Csete ME, Doyle JC. Reverse engineering of biological complexity. *Science* **2002**; 295(5560):1664-1669.
- [3] Nehaniv CL, Rhodes JL. The evolution and understanding of hierarchical complexity in biology from an algebraic perspective. *Artif Life* **2000**; 6(1):45-67.
- [4] Carlson JM, Doyle J. Complexity and robustness. *Proc Natl Acad Sci USA* **2002**; 99 Suppl 1:2538-2545.
- [5] Smaglik P. Is starting simple the path to complexity? *Nature* **2000**; 404(6777):427.
- [6] Kitts DB. The complexity of living bodies and the structure of biological theories. *Acta Biotheor* **1983**; 32(3):195-205.
- [7] Weiss JN, Qu Z, Garfinkel A. Understanding biological complexity: lessons from the past. *FASEB J* **2003**; 17(1):1-6.
- [8] Winslow RL, Boguski MS. Genome informatics: current status and future prospects. *Circ Res* **2003**; 92(9):953-961.
- [9] Wheeler DL, Church DM, Lash AE, *et al.* Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res* **2001**; 29(1):11-16.
- [10] Kulikova T, Aldebert P, Althorpe N, *et al.* The EMBL Nucleotide Sequence Database. *Nucleic Acids Res* **2004**; 32 Database issue:D27-30.
- [11] Schena M, Shalon D, Davis RW, Brown PO. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science* **1995**; 270(5235):467-470.
- [12] Geisow MJ. Proteomics: one small step for a digital computer, one giant leap for humankind. *Nat Biotechnol* **1998**; 16(2):206.
- [13] Chen Y, Xu D. Computational analyses of high-throughput protein-protein interaction data. *Curr Protein Pept Sci* **2003**; 4(3):159-181.
- [14] Drewes G, Bouwmeester T. Global approaches to protein-protein interactions. *Curr Opin Cell Biol* **2003**; 15(2):199-205.
- [15] Rodi DJ, Makowski L, Kay BK. One from column A and two from column B: the benefits of phage display in molecular-recognition studies. *Curr Opin Chem Biol* **2002**; 6(1):92-96.
- [16] Saito R, Suzuki H, Hayashizaki Y. Construction of reliable protein-protein interaction networks with a new interaction generality measure. *Bioinformatics* **2003**; 19(6):756-763.

- [17] Amato NM, Song G. Using motion planning to study protein folding pathways. *J Comput Biol* **2002**; 9(2):149-168.
- [18] Maggiora GM, Rohrer DC, Mestres J. Comparing protein structures: a Gaussian-based approach to the three-dimensional structural similarity of proteins. *J Mol Graph Model* **2001**; 19(1):168-178.
- [19] Martin NP, Leavitt LM, Sommers CM, Dumont ME. Assembly of G protein-coupled receptors from fragments: identification of functional receptors with discontinuities in each of the loops connecting transmembrane segments. *Biochemistry* **1999**; 38(2):682-695.
- [20] Samudrala R, Xia Y, Levitt M, Huang ES. A combined approach for ab initio construction of low resolution protein tertiary structures from sequence. *Pac Symp Biocomput* **1999**:505-516.
- [21] Shao X, Grishin NV. Common fold in helix-hairpin-helix proteins. *Nucleic Acids Res* **2000**; 28(14):2643-2650.
- [22] Shoichet BK, Kuntz ID. Matching chemistry and shape in molecular docking. *Protein Eng* **1993**; 6(7):723-732.
- [23] Wang Y, Ke C, Brown MB. Shape-invariant modeling of circadian rhythms with random effects and smoothing spline ANOVA decompositions. *Biometrics* **2003**; 59(4):804-812.
- [24] Bahar I, Kaplan M, Jernigan RL. Short-range conformational energies, secondary structure propensities, and recognition of correct sequence-structure matches. *Proteins* **1997**; 29(3):292-308.
- [25] Gray RA, Vander Velde DG, Burke CJ, Manning MC, Middaugh CR, Borchardt RT. Delta-sleep-inducing peptide: solution conformational studies of a membrane-permeable peptide. *Biochemistry* **1994**; 33(6):1323-1331.
- [26] Grishin NV. Fold change in evolution of protein structures. *J Struct Biol* **2001**; 134(2-3):167-185.
- [27] Crivelli S, Eskow E, Bader B, et al. A physical approach to protein structure prediction. *Biophys J* **2002**; 82(1 Pt 1):36-49.
- [28] Fetrow JS, Giammona A, Kolinski A, Skolnick J. The protein folding problem: a biophysical enigma. *Curr Pharm Biotechnol* **2002**; 3(4):329-347.
- [29] Hoffman DL, Laiter S, Singh RK, Vaisman II, Tropsha A. Rapid protein structure classification using one-dimensional structure profiles on the bioSCAN parallel computer. *Comput Appl Biosci* **1995**; 11(6):675-679.
- [30] Kinch LN, Wrabl JO, Krishna SS, et al. CASP5 assessment of fold recognition target predictions. *Proteins* **2003**; 53 Suppl 6:395-409.
- [31] Skolnick J, Fetrow JS. From genes to protein structure and function: novel applications of computational approaches in the genomic era. *Trends Biotechnol* **2000**; 18(1):34-39.
- [32] Xia Y, Huang ES, Levitt M, Samudrala R. Ab initio construction of protein tertiary structures using a hierarchical approach. *J Mol Biol* **2000**; 300(1):171-185.
- [33] Ideker T, Lauffenburger D. Building with a scaffold: emerging strategies for high- to low-level cellular modeling. *Trends Biotechnol* **2003**; 21(6):255-262.
- [34] Kitano H. Computational systems biology. *Nature* **2002**; 420(6912):206-210.
- [35] Thomas R, Kaufman M. Conceptual tools for the integration of data. *CR Biol* **2002**; 325(4):505-514.
- [36] Krishnamurthy L, Nadeau J, Ozsoyoglu G, et al. Pathways database system: an integrated system for biological pathways. *Bioinformatics* **2003**; 19(8):930-937.
- [37] Thomaseth K. Multidisciplinary modelling of biomedical systems. *Comput Methods Programs Biomed* **2003**; 71(3):189-201.
- [38] Harris JR, Gorley RN. ECoS, a framework for modelling hierarchical spatial systems. *Sci Total Environ* **2003**; 314-316:625-635.
- [39] Galperin MY. The Molecular Biology Database Collection: 2004 update. *Nucleic Acids Res* **2004**; 32 Database issue:D3-22.
- [40] Xiang Y, Poh KL. Time-critical dynamic decision modeling in medicine. *Comput Biol Med* **2002**; 32(2):85-97.
- [41] Winslow RL. Bifurcation analysis of nonlinear retinal horizontal cell models. I. Properties of isolated cells. *J Neurophysiol* **1989**; 62(3):738-749.
- [42] Kohn KW. Molecular interaction map of the mammalian cell cycle control and DNA repair systems. *Mol Biol Cell* **1999**; 10(8):2703-2734.
- [43] Peleg M, Yeh I, Altman RB. Modelling biological processes using workflow and Petri Net models. *Bioinformatics* **2002**; 18(6):825-837.
- [44] Matsuno H, Doi A, Nagasaki M, Miyano S. Hybrid Petri net representation of gene regulatory network. *Pac Symp Biocomput* **2000**; 341-352.
- [45] Doi A, Fujita S, Matsuno H, Nagasaki M, Miyano S. Constructing biological pathway models with hybrid functional Petri nets. *In Silico Biol* **2004**; 4(2):0023.
- [46] Kitano H. A graphical notation for biochemical networks. *Biosilico* **2003**; 1(5):169-176.
- [47] Pirson I, Fortemaison N, Jacobs C, Dremier S, Dumont JE, Maenhaut C. The visual display of regulatory information and networks. *Trends Cell Biol* **2000**; 10(10):404-408.
- [48] Matthiessen MW. BioWareDB: the biomedical software and database search engine. *Bioinformatics* **2003**; 19(17):2319-2320.
- [49] Hucka M, Finney A, Sauro HM, et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **2003**; 19(4):524-531.
- [50] Shapiro BE, Hucka M, Finney A, Doyle J. MathSBML: a package for manipulating SBML-based biological models. *Bioinformatics* **2004**; 20(1): 2829-31.
- [51] Taylor CF, Paton NW, Garwood KL, et al. A systematic approach to modeling, capturing, and disseminating proteomics experimental data. *Nat Biotechnol* **2003**; 21(3):247-254.
- [52] Buerk DG, Lamkin-Kennard K, Jaron D. Modeling the influence of superoxide dismutase on superoxide and nitric oxide interactions, including reversible inhibition of oxygen consumption. *Free Radic Biol Med* **2003**; 34(11):1488-1503.
- [53] Bahar I, Jernigan RL. Stabilization of intermediate density states in globular proteins by homogeneous intramolecular attractive interactions. *Biophys J* **1994**; 66(2 Pt 1):454-466.
- [54] Bustamante CD, Nielsen R, Hartl DL. Maximum likelihood and Bayesian methods for estimating the distribution of selective effects among classes of mutations using DNA polymorphism data. *Theor Popul Biol* **2003**; 63(2):91-103.
- [55] Cannon RC, Hasselmo ME, Koene RA. From biophysics to behavior: Catacomb2 and the design of biologically-plausible models for spatial navigation. *Neuroinformatics* **2003**; 1(1):3-42.
- [56] D'Argenio DZ, Schumitzky A, Wolf W. Simulation of linear compartment models with application to nuclear medicine kinetic modeling. *Comput Methods Programs Biomed* **1988**; 27(1):47-54.
- [57] Gaver DP, 3rd, Kute SM. A theoretical model study of the influence of fluid stresses on a cell adhering to a microchannel wall. *Biophys J* **1998**; 75(2):721-733.
- [58] Grishin NV, Phillips MA, Goldsmith EJ. Modeling of the spatial structure of eukaryotic ornithine decarboxylases. *Protein Sci* **1995**; 4(7):1291-1304.
- [59] Guo XE, McMahon TA, Keaveny TM, Hayes WC, Gibson LJ. Finite element modeling of damage accumulation in trabecular bone under cyclic loading. *J Biomech* **1994**; 27(2):145-155.
- [60] Haugh JM, Wells A, Lauffenburger DA. Mathematical modeling of epidermal growth factor receptor signaling through the phospholipase C pathway: mechanistic insights and predictions for molecular interventions. *Biotechnol Bioeng* **2000**; 70(2):225-238.
- [61] Hucka M, Finney A, Sauro HM, Bolouri H, Doyle J, Kitano H. The ERATO Systems Biology Workbench: enabling interaction and exchange between software tools for computational biology. *Pac Symp Biocomput* **2002**:450-461.
- [62] Loew LM, Schaff JC. The Virtual Cell: a software environment for computational cell biology. *Trends Biotechnol* **2001**; 19(10):401-406.
- [63] Merrill SJ, Murphy BM. Detecting autocatalytic dynamics in data modeled by a compartmental model. *Math Biosci* **2002**; 180:255-262.
- [64] Palsson BO, Narang A, Joshi A. Computer model of human erythrocyte metabolism. *Prog Clin Biol Res* **1989**; 319:133-150; discussion 151-134.
- [65] Schwaber JS, Graves EB, Paton JF. Computational modeling of neuronal dynamics for systems analysis: application to neurons of the cardiorespiratory NTS in the rat. *Brain Res* **1993**; 604(1-2):126-141.
- [66] Voit EO. Models-of-data and models-of-processes in the post-genomic era. *Math Biosci* **2002**; 180:263-274.
- [67] Wall ME, Hlavacek WS, Savageau MA. Design of gene circuits: lessons from bacteria. *Nat Rev Genet* **2004**; 5(1):34-42.
- [68] Wang Y, Guo SW. Statistical methods for detecting genomic alterations through array-based comparative genomic hybridization (CGH). *Front Biosci* **2004**; 9:540-549.

- [69] Winslow RL, Scollan DF, Holmes A, Yung CK, Zhang J, Jafri MS. Electrophysiological modeling of cardiac ventricular function: from cell to organ. *Annu Rev Biomed Eng* **2000**; 2:119-155.
- [70] Ward RC, Yambert MW, Toedte RJ, et al. Creating a human phantom for the virtual human program. *Stud Health Technol Inform* **2000**; 70:368-374.
- [71] Ideker T, Thorsson V, Ranish JA, et al. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science* **2001**; 292(5518):929-934.
- [72] Hasty J, McMillen D, Collins JJ. Engineered gene circuits. *Nature* **2002**; 420(6912):224-230.
- [73] McAdams HH, Arkin A. Towards a circuit engineering discipline. *Curr Biol* **2000**; 10(8):R318-320.
- [74] McAdams HH, Shapiro L. Circuit simulation of genetic networks. *Science* **1995**; 269(5224):650-656.
- [75] Wang D, Gribskov M. Examining the architecture of cellular computing through a comparative study with a computer. *Journal of the Royal Society Interface* **2005**; 2(3):187-195.
- [76] Shegogue D, Zheng WJ. Object-Oriented Biological System Integration: a SARS Coronavirus Example. *Bioinformatics* **2005**; 21(10):2502-2509.
- [77] Hunter PJ, Borg TK. Integration from proteins to organs: the Physiome Project. *Nat Rev Mol Cell Biol* **2003**; 4(3):237-243.
- [78] Sauro HM, Hucka M, Finney A, et al. Next generation simulation tools: the Systems Biology Workbench and BioSPICE integration. *Omic* **2003**; 7(4):355-372.
- [79] Tomita M. Whole-cell simulation: a grand challenge of the 21st century. *Trends Biotechnol* **2001**; 19(6):205-210.
- [80] Tomita M, Hashimoto K, Takahashi K, et al. E-CELL: software environment for whole-cell simulation. *Bioinformatics* **1999**; 15(1):72-84.
- [81] Chikofsky EJ, Cross JHI. Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software* **1990**; 7(1):13-17.
- [82] D'Haeseleer P, Liang S, Somogyi R. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics* **2000**; 16(8):707-726.
- [83] Husmeier D. Reverse engineering of genetic networks with Bayesian networks. *Biochem Soc Trans* **2003**; 31(Pt 6):1516-1518.
- [84] Koza JR, Mydlowec W, Lanza G, Yu J, Keane MA. Reverse engineering of metabolic pathways from observed data using genetic programming. *Pac Symp Biocomput* **2001**:434-445.
- [85] Liang S, Fuhrman S, Somogyi R. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Pac Symp Biocomput* **1998**:18-29.
- [86] Repsilber D, Liljenstrom H, Andersson SG. Reverse engineering of regulatory networks: simulation studies on a genetic algorithm approach for ranking hypotheses. *Biosystems* **2002**; 66(1-2):31-41.
- [87] Tegner J, Yeung MK, Hasty J, Collins JJ. Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling. *Proc Natl Acad Sci USA* **2003**; 100(10):5944-5949.
- [88] Wahde M, Hertz J. Coarse-grained reverse engineering of genetic regulatory networks. *Biosystems* **2000**; 55(1-3):129-136.
- [89] Yeung MK, Tegner J, Collins JJ. Reverse engineering gene networks using singular value decomposition and robust regression. *Proc Natl Acad Sci USA* **2002**; 99(9):6163-6168.
- [90] Zak DE, Gonye GE, Schwaber JS, Doyle FJ, 3rd. Importance of input perturbations and stochastic gene expression in the reverse engineering of genetic regulatory networks: insights from an identifiability analysis of an in silico network. *Genome Res* **2003**; 13(11):2396-2405.
- [91] Ashburner M, Ball CA, Blake JA, et al. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet* **2000**; 25(1):25-29.
- [92] Paulk MC, Curtis B, Chrissis MB, Weber CV. Capability Maturity Model, Version 1.1. *IEEE Software* **1993**; 10(4):18.
- [93] Royce WW. Managing the Development of Large Software Systems. In: IEEE Wescon. San Francisco, CA, USA; 1970; 1-9.
- [94] McDermid J, Rook P: Software Development Process Models. In: Software Engineer's Reference Book, CRC Press; 1993; 26-28.
- [95] Beck K: Preface. In: Extreme Programming Explained, Embrace Change. Addison Wesley; 1999; xv-xxi.
- [96] Mills HD, Dyer D, Linger R. Cleanroom Software Engineering. *IEEE Software* **1987**; 4,5:19-25.
- [97] Jacobson I, Booch G, Rumbaugh J. The Unified Software Development Process, Addison-Wesley; 1998.
- [98] Jacobson I, Booch G, Rumbaugh J. Phases within a Cycle. In: The Unified Software Development Process. Addison-Wesley; 1998; 11-13.
- [99] Pressman RS. Chapter 2. The Process. In: Software Engineering, A Practitioner's Approach. Fourth edn. New York: The McGraw-Hill Companies, Inc; 1997; 22-53.
- [100] Rumbaugh J, Booch G, Jacobson I. The Unified Modeling Language Reference Manual. Reading, MA, Addison-Wesley; 1999.
- [101] Lethbridge T, Laganière R: Object-Oriented Software Engineering: Practical Software Development using UML and Java: McGraw-Hill Europe; **2002**.
- [102] Webb K, White T. UML as a cell and biochemistry modeling language. *Biosystems* **2005**; 80(3):283-302.
- [103] Kohn MC, Tohmaz AS, Giroux KJ, Blumenthal GM, Feezor MD, Millington DS. Robustness of MetaNet graph models: predicting control of urea production in humans. *Biosystems* **2002**; 65(1):61-78.
- [104] Matsuno H, Tanaka Y, Aoshima H, Doi A, Matsui M, Miyano S. Biopathways representation and simulation on hybrid functional Petri net. *In Silico Biol* **2003**; 3(3):389-404.
- [105] Nagasaki M, Doi A, Matsuno H, Miyano S. Genomic Object Net: I. A platform for modelling and simulating biopathways. *Appl Bioinformatics* **2003**; 2(3):181-184.
- [106] Doi A, Nagasaki M, Fujita S, Matsuno H, Miyano S. Genomic Object Net: II. Modelling biopathways by hybrid functional Petri net with extension. *Appl Bioinformatics* **2003**; 2(3):185-188.
- [107] Urbance JW, Cole J, Saxman P, Tiedje JM. BSD: the Biodegradative Strain Database. *Nucleic Acids Res* **2003**; 31(1):152-155.
- [108] Kanehisa M. The KEGG database. *Novartis Found Symp* **2002**; 247:91-101; discussion 101-103, 119-128, 244-152.
- [109] Mueller LA, Zhang P, Rhee SY. AraCyc: a biochemical pathway database for Arabidopsis. *Plant Physiol* **2003**; 132(2):453-460.
- [110] Krieger CJ, Zhang P, Mueller LA, et al. MetaCyc: a multiorganism database of metabolic pathways and enzymes. *Nucleic Acids Res* **2004**; 32 Database issue:D438-442.
- [111] Prokop M, Damborsky J, Koca J. TRITON: in silico construction of protein mutants and prediction of their activities. *Bioinformatics* **2000**; 16(9):845-846.
- [112] Chen M, Hofstadt R. Quantitative Petri net model of gene regulated metabolic networks in the cell. *In Silico Biol* **2003**; 3(3):347-365.
- [113] Hofstadt R. Petri nets and the simulation of metabolic networks. *In Silico Biol* **2003**; 3(3):321-322.
- [114] Moore JH, Hahn LW. Petri net modeling of high-order genetic systems using grammatical evolution. *Biosystems* **2003**; 72(1-2):177-186.
- [115] Oliveira JS, Bailey CG, Jones-Oliveira JB, Dixon DA. An algebraic-combinatorial model for the identification and mapping of biochemical pathways. *Bull Math Biol* **2001**; 63(6):1163-1196.
- [116] Pinney JW, Westhead DR, McConkey GA. Petri Net representations in systems biology. *Biochem Soc Trans* **2003**; 31(Pt 6):1513-1515.
- [117] Reddy VN, Mavrouniotis ML, Liebman MN. Petri net representations in metabolic pathways. *Proc Int Conf Intell Syst Mol Biol* **1993**; 1:328-336.
- [118] Schuster S, Pfeiffer T, Moldenhauer F, Koch I, Dandekar T. Exploring the pathway structure of metabolism: decomposition into subnetworks and application to *Mycoplasma pneumoniae*. *Bioinformatics* **2002**; 18(2):351-361.
- [119] Voss K, Heiner M, Koch I. Steady state analysis of metabolic pathways using Petri nets. *In Silico Biol* **2003**; 3(3):367-387.
- [120] Zevedei-Oancea I, Schuster S. Topological analysis of metabolic networks based on Petri net theory. *In Silico Biol* **2003**; 3(3):323-345.
- [121] Roux-Rouquie M, Caritey N, Gaubert L, Rosenthal-Sabroux C. Using the Unified Modelling Language (UML) to guide the systemic description of biological processes and systems. *Biosystems* **2004**; 75(1-3):3-14.
- [122] Johnson CG, Goldman JP, Gullick WJ. Simulating complex intracellular processes using object-oriented computational modelling. *Prog Biophys Mol Biol* **2004**; 86(3):379-406.
- [123] Fowler M. Introduction. In: UML Distilled. Third Edition edn: Addison-Wesley; 2004; 1-17.

- [124] Fowler M: Class Diagrams: The Essentials. In: *UML Distilled*. Third Edition edn: Addison-Wesley; **2004**; 35-52.
- [125] Fowler M: Sequence Diagrams. In: *UML Distilled*. Third Edition edn: Addison-Wesley; **2004**; 53-61.
- [126] Fowler M: Collaborations. In: *UML Distilled*. Third Edition edn: Addison-Wesley; **2004**; 143-146.
- [127] Fowler M: Activity Diagrams. In: *UML Distilled*. Third Edition edn: Addison-Wesley; **2004**; 117-130.
- [128] Fowler M: State Machine Diagrams. In: *UML Distilled*. Third Edition edn: Addison-Wesley; **2004**; 107-115.
- [129] Fowler M: Package Diagrams. In: *UML Distilled*. Third Edition edn: Addison-Wesley; **2004**; 89-95.
- [130] OMG UML 2.0 Tools. <http://www.uml.org/#Links-UML2Tools> Date Last Accessed: August 11, **2005**.
- [131] Peleg M, Dori D. The Model Multiplicity Problem: Experimenting with Real-Time Specification Methods. *IEEE Transactions On Software Engineering* **2000**; 26(8):742-759.
- [132] CellDesigner. <http://www.systems-biology.org/002/> Date Last Accessed: August 11, **2005**.
- [133] Hucka M, Finney A. Escalating model sizes and complexities call for standardized forms of representation. *Mol Sys Bio* **2005**; doi:10.1038/msb4100015.
- [134] Davis R, Shrobe H, Szolovits P. What is a knowledge representation? *AI Magazine* **1993**; 14(1):17-33.
- [135] Gruber TR. A translation approach to portable ontology specifications. *Knowledge Acquisition* **1993**; 5(2):199-220.
- [136] Bratko I: Prolog Programming for Artificial Intelligence, 3rd edn: Addison Wesley; **2000**.
- [137] Kitano H. Systems biology: a brief overview. *Science* **2002**; 295(5560):1662-1664.
- [138] de la Fuente A, Snoep JL, Westerhoff HV, Mendes P. Metabolic control in integrated biochemical systems. *Eur J Biochem* **2002**; 269(18):4399-4408.
- [139] Alvarez-Vasquez F, Sims KJ, Cowart LA, Okamoto Y, Voit EO, Hannun YA. Simulation and validation of modelled sphingolipid metabolism in *Saccharomyces cerevisiae*. *Nature* **2005**; 433(7024):425-430.
- [140] Shegogue D, Zheng WJ. Capturing biological information with class-responsibility-collaboration cards. *Bioinformatics* **2005**; 21(3):415-417.
- [141] Shegogue D, Zheng WJ. Integration of the Gene Ontology into an object-oriented architecture. *BMC Bioinformatics* **2005**; 6(1):113.
- [142] Pressman RS: Software Engineering. In: *Software Engineering Project Management*. Edited by Thayer RH; Matt Loeb; **1997**; 30-47.
- [143] Brooks FPJ: The Mythical Man-month: Addison-Wesley; **1975**.
- [144] Fenton NE, Pfleeger SL: Software Metrics: A Rigorous and Practical Approach, Revised, 2nd edn: Course Technology; **1998**.
- [145] Kidd J, Lorenz M: Object-oriented Software Metrics: Pearson Education POD; **1994**.
- [146] Boehm BW, Horowitz E, Madachy R, *et al*. Software Cost Estimation with Cocomo II: Prentice Hall PTR; **2000**.
- [147] CVS. <http://www.nongnu.org/cvs/> Date Last Accessed: August 11, **2005**.
- [148] SourceForge.net. <http://sourceforge.net/> Date Last Accessed: August 11, **2005**.
- [149] Schacherer F, Choi C, Gotze U, Krull M, Pistor S, Wingender E. The TRANSPATH signal transduction database: a knowledge base on signal transduction networks. *Bioinformatics* **2001**; 17(11): 1053-1057.
- [150] Pevzner PA. Educating biologists in the 21st century: bioinformatics scientists versus bioinformatics technicians. *Bioinformatics* **2004**; 20(14):2159-2161.
- [151] Schadt EE. The making of a bioinformatician. *J Cell Biochem* **2000**; 80(2):212-215.
- [152] Zauhar RJ. University bioinformatics programs on the rise. *Nat Biotechnol* **2001**; 19(3):285-286.
- [153] Friedman CP, Altman RB, Kohane IS, *et al*. Training the next generation of informaticians: the impact of "BISTI" and bioinformatics--a report from the American College of Medical Informatics. *J Am Med Inform Assoc* **2004**; 11(3):167-172.
- [154] Bialek W, Botstein D. Introductory science and mathematics education for 21st-Century biologists. *Science* **2004**; 303(5659):788-790.
- [155] Lyon J, Giuse NB, Williams A, Koonce T, Walden R. A model for training the new bioinformatician. *J Med Libr Assoc* **2004**; 92(2):188-195.
- [156] Gross LJ. Education for a biocomplex future. *Science* **2000**; 288(5467):807.
- [157] Zatz MM. Bioinformatics training in the USA. *Brief Bioinform* **2002**; 3(4):353-360.
- [158] Johnson SB. A framework for the biomedical informatics curriculum. *AMIA Annu Symp Proc* **2003**:331-335.
- [159] Pike LJ, Sadler JE. Proteomics, genomics and the future of medical education. *Mo Med* **2004**; 101(5):496-499.
- [160] Altman RB. A curriculum for bioinformatics: the time is ripe. *Bioinformatics* **1998**; 14(7):549-550.
- [161] Cattley S. A review of bioinformatics degrees in Australia. *Brief Bioinform* **2004**; 5(4):350-354.
- [162] Johns SJ, Thompson SM, Dunker AK. An introductory course in computation molecular biology: rationale, history, observations, and course description. *Pac Symp Biocomput* **1996**:396-407.