Scalable Computational Architecture for Integrating Biological Pathway Models

by

V.A. Shiva Ayyadurai

S.B., Electrical Engineering and Computer Science, MIT (1987)
S.M., Media Arts and Sciences, MIT (1989)
S.M., Mechanical Engineering, MIT (1990)

SUBMITTED TO THE DEPARTMENT OF BIOLOGICAL ENGINEERING IN PARITAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN BIOLOGICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[September 2007]
AUGUST 2007

Signature of Author: [illegible]
Department of Biological Engineering
August 29, 2007

Certified by: [illegible]
C. Forbes Dewey, Jr.
Professor of Mechanical and Biological Engineering
Thesis Supervisor

Accepted by: [illegible]
Alan J. Grodzinsky
Professor of Electrical, Mechanical and Biological Engineering
Chairman, Biological Engineering Graduate Program Committee

V.1

Thesis Defense Date: August 13, 2007 [1].

[1] The thesis was presented on August 13, 2007, which also marks the 89th birthday of Dr. Frederick Sanger (b. August 13, 1918). Dr. Sanger won two Nobel prizes in Chemistry. In 1958, he was awarded his first Nobel prize in Chemistry for proving that proteins have definite structure by sequencing insulin. In 1980, he was awarded his second Nobel prize in Chemistry along with Walter Gilbert for the sequencing the first DNA-based genome. This discovery has been the basis of the Human Genome Project. In 1992, the Sanger Centre near Cambridge, England was founded in honor of Dr. Sanger. The visit to the Sanger Institute by the author in October 2003 marked the start of the author's journey into the field of systems biology and biological engineering.

To my loving parents

Scalable Computational Architecture for Integrating Biological Pathway Models

by

V.A. Shiva Ayyadurai

Submitted to the Department of Biological Engineering
on August 13, 2007, in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy in Biological Engineering

Abstract

A grand challenge of systems biology is to model the cell. The cell is an integrated network of *cellular functions*. Each cellular function, such as immune response, cell division, metabolism or apoptosis, is defined by an interconnected ensemble of *biological pathways*. Modeling the cell or even one cellular function requires a computational architecture that integrates multiple biological pathway models in a scalable manner while ensuring minimal effort to maintain the resulting integrated model. Scalable is defined as the ease in which more and more biological pathway models can be integrated. Current architectures for integrating biological pathway models are primarily *monolithic* and involve combining each biological pathway model's software source code to build one large monolithic model that executes on a single computer. Such architectures are not scalable for modeling complex cellular functions or the whole cell.

We present *Cytosolve*, a new computational architecture that integrates a distributed ensemble of biological pathway models and computes solutions in a parallel manner while offering ease of maintenance of the integrated model. The individual biological pathway models can be represented in SBML, CellML or in any number of formats. The EGFR model of Kholodenko with known solutions is used to compare the Cytosolve solution and computational times with a known monolithic approach. A new integrative model of the interferon (IFN) response to virus infection is developed using Cytosolve. Each model within the integrated model, spans different time scales, is created by different authors from four countries and three continents across different disciplines, is written in different software codes, and is built on different hardware platforms. A new quantitative methodology and formalism is then derived for evaluating different types of monolithic and distributed architectures for integrating biological pathway models.

As more biological pathway models develop in a disparate and decentralized manner, the Cytosolve architecture offers a unique platform to build and test complex models of cellular function, and eventually the whole cell.

Thesis Supervisor: C. Forbes Dewey, Jr.
Title: Professor of Biological Engineering & Mechanical Engineering

Acknowledgments

While a dissertation is meant to represent the work of a single person, every dissertation is necessarily the result of numerous interactions between the author, the author's advisors, peers, colleagues, friends, and family. This one is no different.

My advisor
Prof. C. Forbes Dewey, a fellow pioneer, is my primary collaborator. Without him, this thesis would not have been possible. He provided me the opportunity to enter into the new field of systems biology and continually offered his wisdom and encouragement to move forward with conviction. Forbes is one of the kindest and insightful people I have ever come across and am honored and fortunate have his friendship and mentorship.

My thesis committee members, readers and advisors
My thesis committee members and other advisors including Prof. John M. Essigmann, Prof. Robert C. Berwick, Prof. Ahmed F. Ghoniem, Prof. Bruce Tidor and Prof. Douglas A. Lauffenburger. Their wisdom, friendship, feedback and encouragement have been invaluable.

My friends
My most sincere thanks and love to my long time friends Sonu Mathews Abraham, Gene Deans, and Hariharan Subramanian, three people who I don't know what I would do without.

New friends and old ones that I reconnected with who shared their compassion and insights both professionally and personally including Simon Dao, Saloni Fadia, David Calvo, Sen Song, Carolyn Dewey, Asawari Desai, Devan Dewey, Sangeetha Modi, David Calvo, Phil and Alise Rheinstein, Sen Song, Marina del Rohrer, Kim Moore, Tenzin Priyardarshi, Tricia Harris, Nevan Hanumara, Martin Feuerman and Dick Kitney

MIT
Catherine Howell afforded her time on many occasions to provide the knowledge needed to understand JSIM and the actin polymerization model, developed by Jim McGrath and Mike Binschadler. It was this model which first introduced me to the area of biological pathway modeling.

Over the past four years, at Hatsopolous Microfluidics Laboratory, is where my home has been at MIT. That home has included a cast of characters: Donna Wilker, Kurt Stiehl, Yu Yao, Aleksandr Rabodzey, Khanh Dang, and many others.

At the start of my doctoral program, given my background across three departments, it was Frank E. Perkins initially and then later Ike Colbert who provided me the guidance to formulate the administration behind my doctoral program, at a time when the Biological Engineering Division, did not even exist.

This journey was also a great opportunity to reconnect with my Freshman Advisor Roger Mark, and also Paul E. Gray, Samuel J. Keyser, Bob Randolph and others who knew me well during my undergraduate years and were kind and gracious to see me back.

I want to also give a special thanks to Leslie Regan, Joan Kravit, Dalia Fares and Michelle Carmichael who took care of all the little details and for their dedication to students and faculty.

During my return to MIT, Joel Moses was very helpful in offering his time to brainstorm on different thesis topics.

Very special thanks to Dawn Metcalf for her compassion, generosity and wisdom.

My friends Dan Burns and Devin McCombie were great partners to have as a part of the study team during the doctoral qualifiers.

John Essigmann is a great teacher. I thank him for his wonderful and positive attitude in my learning his course his BE440. Paul Huang also deserves special thanks for first being my TA in BE440 and later a good friend.

Ram Sasisekharan, Alan Grodzinsky, Forbes Dewey, Roger Kamm. Bruce Tidor, Jacob White, Anthony Patera and Jaime Peraire, deserve much appreciation for their great courses, in which they conveyed new knowledge with grace and elegance, while expanding my universe of biology and numerical computing.

My utmost appreciation and respect to Paul Matsudaira, Doug Lauffenburger and Subra Suresh for their vision in being able to bring together new platforms for education and research including computational systems biology (CSBi), biological engineering (BE) and new programs such as GEM4 and SMA so many can pursue their dreams.

Architecture Development and Testing
Ceryen Tan, Gene Deans and Prasad Jayakumar for providing me information on Web Services, and supporting my time to time programming needs of the distributed solver.

Jia-An (Andrew) Koo for his help at the last minute in helping test the IFN pathway along with Boon Siew Seah's last minute programming efforts on the controller.

Video Production
In August of 2005, I met Philip Pfeifer, my Vipassana teacher. Later Philip and his three artisans, part of a non-profit collective, Brian Sneed, Arik Thusen, Harry Hunsberger performed the 3-D animation and video rendering of the IFN response to virus infection video. That video would not have been possible without them.

Singapore-MIT Alliance (SMA)
The Singapore MIT Alliance for their funding my last three years of research along with their generous travel grants. My special thanks to Sourav Bhowmick for his friendship and making me feel at home during my travels to Singapore.

EchoMail
Leaving EchoMail to take on the PhD was a bigger transition than I thought. My friends and colleagues at EchoMail including Angie Christensen, Roman Zavolly, Evan Siegel made that transition easier. The Board including Larry Weber, Sonu Abraham, V. Ayyadurai (my dad), Ed Fredkin and Ted Johnson deserve my thanks for their many years of support and encouraging of my intellectual desires. Special thanks for EchoMail's financial support of my first year in the PhD program.

My teachers
There are many through the years stretching back to secondary school who are part of this thesis including: Mr. Melvin Roth (my sixth grade teacher), Mr. Walker (my chemistry teacher), Mr. Kramer (my calculus teacher), Mr. Sommer (my algebra teacher), Mrs. Hall and Ms. Payne (my writing teachers). Special thanks to Paul Pitchford and the Heartwood Institute (for teaching me TCM), Rick Buckely (for opening my eyes to the power of observation), Kelly Mara (my Yoga teacher), Philip Pfeifer and S.N. Goenka (who taught me Vippassana), Alise Newton (who taught me to walk right), Warren Senders (who taught me to hear so I could sing).

I will never forget Dr. Leslie P. Michelson for bringing me into the world of computers and giving me my first programming job when I was 13 at the UMDNJ which led to my work on developing one of the world's first E-Mail system. He is one of the most brilliant people I've met. He taught me the importance of writing well and thinking clearly. Without him I would never have gotten such a head start in computer science.

In memory
Two years ago, Swamy Laxminarayan passed away. He was the one who introduced me to biomedical engineering when I was a high school sophomore and gave the opportunity to learn signal processing on a sleep apnea project, for which he also made me a co-author on my first paper in 1985, that took me to Espoo, Finland for the IEEE International Conference on Medical and Biological Engineering.

My maternal grandparent Annamalai who was always there for me and I will always remember her great cooking and kindness.

My paternal grandparents would have been happy to see this work. My grandmother Chinnathai and my grandfather Vellayappa, passed on in 2005. I pay reverence to their loving spirits.

Satguru Sivaya Subramuniyaswami deserves many thanks for his wonderful writings and teachings, which are a source of great truth and comfort.

My family
Last but not least, I want to thank my family.

My mother, Meenakshi Ayyadurai, the most amazing person I know in the entire world. She embodies the independence, resourcefulness and grace of being a powerful woman, blending East and West, being a living example to others who struggle to reach that balance. Without her, I would never have entered the world of computing nor would I have had access to many of the unique doors that she opened that led me to great opportunities at a very early age.

My father, Vellayappa Ayyadurai, my living hero, who as a child experienced bombs being dropped in WWII in Burma, educated himself, electrified villages, mentored thousands, and embodies the wisdom and compassion of a long-lost statesman, the kind the world sorely needs today.

My loving sister Dr. Uma Dhanabalan and my wonderful nephew Shivaji Dhanabalan deserve my love and thanks for always being there. Uma is one the kindest human beings I know who has always been a great source of support and wisdom.

Finally, Romeo, thanks for keeping me company during late night writing sessions.

Biographical Note

V.A. Shiva Ayyadurai is a Fulbright Scholar, MIT-Lemelson Awards Finalist and Westinghouse Science Award recipient. He was born in Bombay, India and moved to the United States at the age of seven. He completed his secondary school education in New Jersey. He is the inventor of one of the world's first E-Mail systems holding the first US Copyright on E-MAIL. Shiva also holds three US Patents in automatic pattern recognition. He has published scientific articles in several conference proceedings and refereed journal articles. In January of 2000, one of Shiva's inventions *EchoMail* was the featured story in *The MIT Technology Review*. He has appeared in columns and articles in *The Wall Street Journal, New York Times, NBC News, USA Today* and other major publications. He was named Top 40 in the *Improper Bostonian*. He is also the author of two books: *Arts and the Internet* and *The Internet Guide to Publicity*. He is a member of *Sigma-Xi, Eta Kappa Nu* and *Tau Beta Pi*. He is a founder of the Shanthi Foundation which raises money to provide scholarships for education to orphaned girls. He is also a supporter of various arts and non-profit organizations. Shiva enjoys yoga, travel, tennis, animals, art and architecture, and lives Belmont, MA, USA.

Content

Preface

This thesis is motivated by my interest in biology, computing, medicine and integrative systems research. Growing up in the outskirts of Bombay, India as a young child, my backyard *was* a true jungle. Animals of all shapes and sizes lurked including large poisonous snakes, parrots, land crabs, mongoose and bugs of all kind. The vegetation was lush with many fruits: mangoes, guava, bananas, cherimoyas, and coconuts. Nearby, and in contrast to this wilderness, was the bustling city of Bombay, filled with road side vendors, snake charmers, colors, smells and sounds that never made any day boring.

My first introduction to medicine came from my grandmother who lived in a village in the depths of South India. I recall here once saying to my uncle, "Go down to the stream and you will see a flower which has pointed green leaves and purple flowers, that are fuzzy, but have thorns at the end of them, go bring those." He had a sinus infection and had to her for help. A few hours later he came back with a handful of leaves and flowers. She processed them to make a formulation. She administered it to him directly through his nose. A few drops of blood appeared, and he immediately felt great relief.

My paternal grandmother was the Shaman of our village *Muhavur* in South India. Over the years, the story goes that traveling Yogis had imparted their secrets and knowledge to her. Each Saturday, numerous people milled at her doorstep waiting for her medical wisdom. She had excellent powers of observation and exceptional diagnosis skills. Using mortar and pestle she would make simple to esoteric formulations. For our village, she was a boon. As a young child, I was both awed and inspired. I developed a curiosity to understand the secrets of her formulations and the holistic medicine she practiced to diagnose and treat without any modern instruments.

My father taught me mathematics and chemistry. I recall him getting me my first chemistry set and making formulations using common concepts that one studies in their first high school undergraduate chemistry class. I was amazed by his wizardry. As a trained chemical engineer, he ironically also went on to become a provider of medicines for many people. He took a different approach by becoming the head of manufacturing in Bombay, India for Parke-Davis, a leading pharmaceutical company. Like his mother, he created formulations; however, these formulations were proprietary and patented, developed by using reductionist principles of modern Western science and employed modern instruments, products of high technology.

My family moved to the United States in the early 1970's and I grew up in two worlds: Indian at home, American at school and work. The stark difference between these two modes of life was as different as the ways in which my grandmother and my father formulated medicines for their customers. I had great regard for both of these different systems, and felt that one of my life's missions was to find ways to integrate them. These differences influenced me to seek unity and holism in any activity that I pursued.

My mother, a mathematician and software programmer, introduced me to computer programming during the mid-70s. As a 13 year old, I attended the New York University, Courant Institute of Mathematical Sciences on an accelerated summer program to learn five different computer languages including: FORTRAN, COBOL, PL/1, SNOBOL and Basic. In 1977, I used that knowledge to build one of the world's first E-Mail Systems. Because of my mom, I became a decent software programmer and learned about architecting large-scale enterprise computing systems. In 1994, while in the midst of my doctoral program at MIT, I took a hiatus to start a company, EchoMail, Inc. During 1994 to 2003, I found and ran EchoMail, a leading enterprise software company that provided an intelligent and scalable E-Mail management system for Global 2000 companies such as American Express, Nike, Citigroup, JC Penney and Allstate.

In the Fall of 2003, I came back to MIT to visit old friends, administrators and professors. I ran into Forbes Dewey on the second floor of Building 3. In the mid-80's, I had worked as a laboratory teaching assistant and student programmer for one of Forbes' courses. He remembered me and invited me to chat with him and shared the vision of systems biology. Forbes said that he was on his way the following Monday, October 27 to attend the I3C Conference in Hinxton, U.K. which was focused on "accelerating drug discovery using software interoperability," and invited me to come. A few days thereafter I was on a plane to the UK. After attending that I3C Conference, understanding Forbes' vision, and touring the Sanger Institute, where critical portions of the human genome were sequenced, it became clear to me that biology was about to change drastically. New technologies that enabled biological information to be transacted and integrated with ease would revolutionize medicine. During this same time, MIT had created the Biological Engineering Division and a program in Computational Systems Biology.

Forbes encouraged me to return to MIT to pursue a doctorate in the field of systems biology. I was enthused, particularly because this field seemed committed to integrate smaller things to find emergent properties of the whole. The field also provided a vision of Personalized Medicine, where one size does not fit all, encouraging a holistic and targeted approach to the development of new medicines versus the highly reductionist approach. More importantly, systems biology offered me the opportunity to integrate two of my interests: Software Development and Medicine.

In 2004, I returned back to MIT and began my journey in pursuit of my doctorate in - a journey that ends on August 13, 2007 with the completion of my Thesis Defense. I hope I have made a contribution to the field through by offering a new paradigm for integrating biological pathway models - a paradigm that leverages the advent of the

World Wide Web and the global use of the Internet, where biologists can create independent models and then collaborate to build new models, with ease, accelerating the development of new discoveries.

The completion of this thesis also has a new beginning. Starting in October of 2007, I will return for nine months back to India on a Fulbright to study a 6,000 year-old form of ancient Indian medicine known as Siddha, the same system my grandmother used which is still practiced today. My task is to consider the cellular function of inflammatory response, but this time approach this phenomenon with two eyes: one of the East and one of the West's. My hope is to integrate two systems of knowledge to discover an integrative link between them. I am confident that in the not too distant future, technologies perhaps even the one developed in this thesis, will enable one to combine knowledge from seemingly disparate medical systems, to discover new formulations to age-old ailments that neither the Eastern system nor the Western system could discover alone.

Belmont, Massachusetts

August 2007

Chapter 1

Introduction

1.0 Background and Significance

A grand challenge of systems biology is to *model* the whole cell. This thesis offers

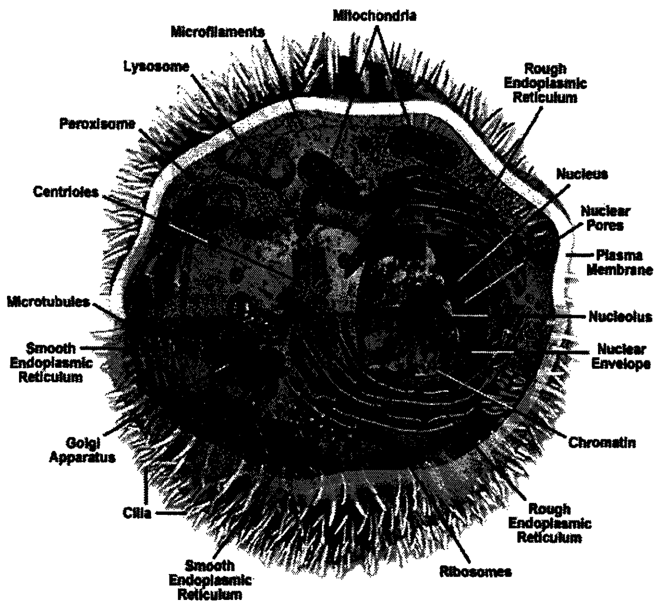


Figure 1-1: Diagram of a eukaryotic animal cell illustrating its key components. Examples of cellular functions by component include: **protein synthesis** at the Ribosomes, **metabolism** within the Mitochondria, **cell motility** using Cilia and Microfilaments (Davidson, 2007).

a new computational architecture to address that challenge. A model in this thesis is defined to be a mathematical representation along with its implementation in software including any input data and documentation.

A cell consists of a set of organelles as the ones shown in Figure 1-1. These organelles interact through the medium of molecular interactions to provide *cellular functions* such as protein synthesis, metabolism, apoptosis, or motility. Systems biology aims to develop a model of the cell by connecting the biochemical kinetics of these interactions at the molecular mechanistic level to derive the quantitative descriptions of higher level cellular functions (Hood, et al., 2004; Ideker and Lauffenburger, 2003; Kitano, 2002; Palsson, et al., 2003; Tomita, et al., 1999). Systems biology as a new field is also interdisciplinary attracting those from a wide range of disciplines. Many who approach this field come with various biases. Core to many of these biases is a failure to understand the complexity of biology and a mistaken notion that systems biology is *not* a new field. The next two sections serve to clarify basic concepts concerning the complexity of biology as well as why systems biology *is* a new field.

1.1 The Complexity of Biology

Biology is a field based on experiments, not first principles (*ab initio*) such as physics or engineering. It is fundamentally an experimental science. Biologists do many experiments to understand genes, proteins, protein-protein interactions.

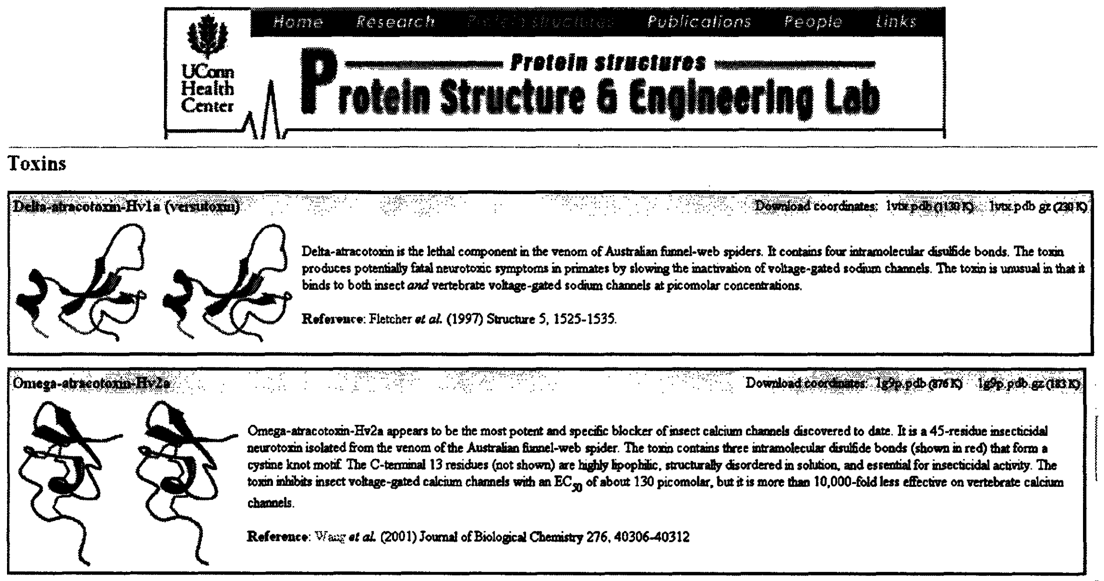
Genes

One example of perhaps the largest experiments in biology is the Human Genome Project (HGP) begun in 1990 and completed in 2005. This effort resulted in the discovery of only 20,000 to 25,000 genes, far less than what was originally theorized (Pennisi, 2003). More interesting is the discovery that this number of genes is in the same realm as that of the nematode *Caenorhabditis elegans* which has approximately 19,000 genes (Hodgkin, 2001). More recently, the genome of the starlet sea anemone - *Nematostella vectensis,* a delicate, few-inch-long animal in the form of a transparent, multi-tentacled tube - was sequenced and found to have 18,000 genes (Putnam, 2007).

Human and a nematode (or sea anemone) have a similar number of genes, but a great difference in complexity of function as whole organisms. This contradiction has led scientists to conclude that perhaps the number of genes in the genome is not connected with the complexity of the organism. Much of an organism's complexity can be ascribed to regulation of existing genes by other substances (such as proteins) rather than to novel genes (Putnam, 2007). The types and kinds of molecular interactions across the nucleus, cytoplasm and organelles, beyond the number of genes in the nucleus, may be the critical element in determining the difference between human and nematode, for example. This reasoning has led to an even greater activity to understand the structure of proteins (e.g. the product of genes) and protein-protein interactions.

Proteins

Approximately 30,000 proteins have been documented across various publications world wide (Peri, 2003). Thousands of research teams across the world have contributed to these discoveries.



Toxins

Delta-atracotoxin-Hv1a (versutoxin)   Download coordinates: 1vtx.pdb (130 K) 1vtx.pdb.gz (230 K)

Delta-atracotoxin is the lethal component in the venom of Australian funnel-web spiders. It contains four intramolecular disulfide bonds. The toxin produces potentially fatal neurotoxic symptoms in primates by slowing the inactivation of voltage-gated sodium channels. The toxin is unusual in that it binds to both insect *and* vertebrate voltage-gated sodium channels at picomolar concentrations.

Reference: Fletcher *et al.* (1997) Structure 5, 1525-1535.

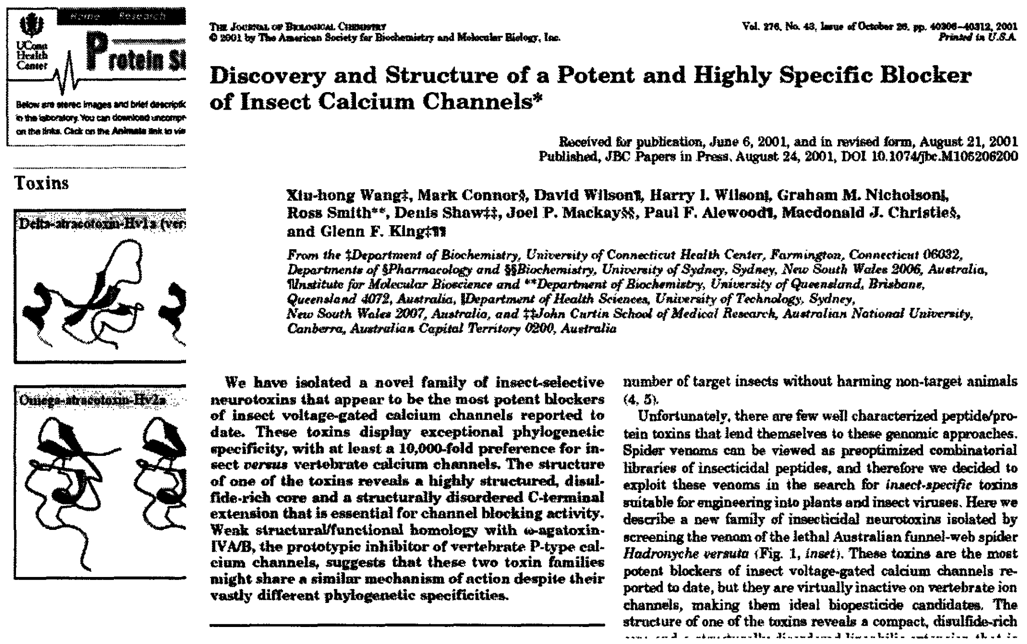
Omega-atracotoxin-Hv2a   Download coordinates: 1g9p.pdb (876 K) 1g9p.pdb.gz (183 K)

Omega-atracotoxin-Hv2a appears to be the most potent and specific blocker of insect calcium channels discovered to date. It is a 45-residue insecticidal neurotoxin isolated from the venom of the Australian funnel-web spider. The toxin contains three intramolecular disulfide bonds (shown in red) that form a cystine knot motif. The C-terminal 13 residues (not shown) are highly lipophilic, structurally disordered in solution, and essential for insecticidal activity. The toxin inhibits insect voltage-gated calcium channels with an EC_{50} of about 130 picomolar, but it is more than 10,000-fold less effective on vertebrate calcium channels.

Reference: Wang *et al.* (2001) Journal of Biological Chemistry 276, 40306-40312

Figure 1-2: Diagram demonstrating how a research group at one laboratory facility in the world focuses on understanding a handful of proteins. In this case, one research team focuses on understanding the structure of certain types of toxins extracted from certain insects.

Discovering the structure of just one protein is a difficult experimental effort. Such efforts are highly domain specific and one research team, for example, by itself may focus on understanding a small set of genes or the structure of a set of specific proteins or the interactions between certain types of proteins. For example, consider the Protein and Structure and Engineering Laboratory at the


Home Research
UConn Health Center
Protein St
Delta-atracotoxin-Hv1a (ver
Omega-atracotoxin-Hv2a

Below are stereo images and brief descriptic
in the laboratory. You can download uncompr
on the links. Click on the Animals link to vie

Toxins

THE JOURNAL OF BIOLOGICAL CHEMISTRY
© 2001 by The American Society for Biochemistry and Molecular Biology, Inc.

Vol. 276, No. 43, Issue of October 26, pp. 40306–40312, 2001
Printed in U.S.A.

Discovery and Structure of a Potent and Highly Specific Blocker of Insect Calcium Channels*

Received for publication, June 6, 2001, and in revised form, August 21, 2001
Published, JBC Papers in Press, August 24, 2001, DOI 10.1074/jbc.M106206200

Xiu-hong Wang‡, Mark Connor§, David Wilson¶, Harry I. Wilson‖, Graham M. Nicholson‖, Ross Smith, Denis Shaw‡‡, Joel P. Mackay§§, Paul F. Alewood¶, Macdonald J. Christie§, and Glenn F. King‡¶¶**

*From the ‡Department of Biochemistry, University of Connecticut Health Center, Farmington, Connecticut 06032, Departments of §Pharmacology and §§Biochemistry, University of Sydney, Sydney, New South Wales 2006, Australia, ¶Institute for Molecular Bioscience and **Department of Biochemistry, University of Queensland, Brisbane, Queensland 4072, Australia, ‖Department of Health Sciences, University of Technology, Sydney, New South Wales 2007, Australia, and ‡‡John Curtin School of Medical Research, Australian National University, Canberra, Australian Capital Territory 0200, Australia*

We have isolated a novel family of insect-selective neurotoxins that appear to be the most potent blockers of insect voltage-gated calcium channels reported to date. These toxins display exceptional phylogenetic specificity, with at least a 10,000-fold preference for insect *versus* vertebrate calcium channels. The structure of one of the toxins reveals a highly structured, disulfide-rich core and a structurally disordered C-terminal extension that is essential for channel blocking activity. Weak structural/functional homology with ω-agatoxin-IVA/B, the prototypic inhibitor of vertebrate P-type calcium channels, suggests that these two toxin families might share a similar mechanism of action despite their vastly different phylogenetic specificities.

number of target insects without harming non-target animals (4, 5).

Unfortunately, there are few well characterized peptide/protein toxins that lend themselves to these genomic approaches. Spider venoms can be viewed as preoptimized combinatorial libraries of insecticidal peptides, and therefore we decided to exploit these venoms in the search for *insect-specific* toxins suitable for engineering into plants and insect viruses. Here we describe a new family of insecticidal neurotoxins isolated by screening the venom of the lethal Australian funnel-web spider *Hadronyche versuta* (Fig. 1, *inset*). These toxins are the most potent blockers of insect voltage-gated calcium channels reported to date, but they are virtually inactive on vertebrate ion channels, making them ideal biopesticide candidates. The structure of one of the toxins reveals a compact, disulfide-rich

Figure 1-3: An example of a publication focused on the structure and discovery of just one protein. Work on this research was the result of effort by three research groups across two countries: United States and Australia.

University of Connecticut's Health Center. One research team at this laboratory focuses on just understanding the protein structure of toxins from certain animals as shown in Figure 1-2. The protein structures are determined from experiments using x-ray crystallography (Laue, 1913). While new proteins are discovered each day, the crystal structure of most proteins is not known. In many cases, understanding just *one* protein structure requires the effort of not just this one laboratory's research team but the effort of multiple research teams spread across the world. Figure 1-3 presents the title and abstract of a paper published in the *Journal of Biological Chemistry* (Wang, 2001). This paper shares the discovery of a particular protein structure. The paper is the result of a combined effort of three research groups across two countries: United States and

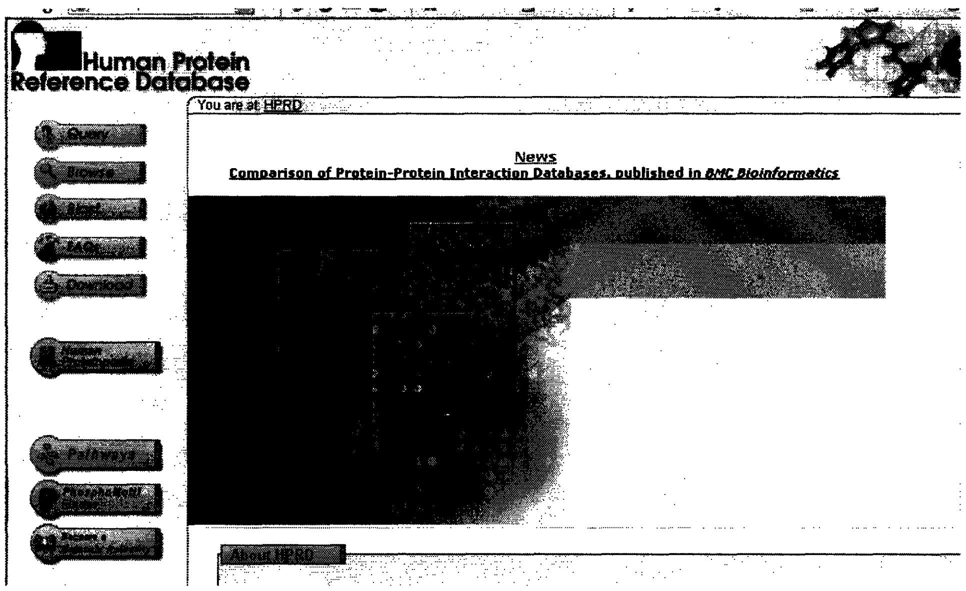
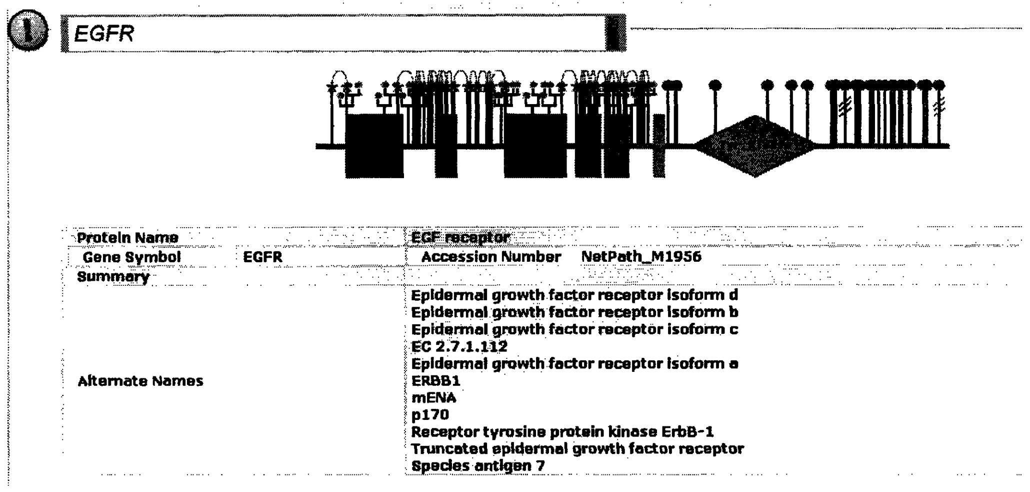

Human Protein Reference Database
You are at HPRD
News
Comparison of Protein-Protein Interaction Databases, published in BMC Bioinformatics
About HPRD

Figure 1-4: The Human Protein Reference Database (HPRD) is an example of one of many web-based repositories of protein data information(Peri, 2003) .

Australia. New databases such as: HPRD, OMIM, PDB, Entrez Gene, HGNC, Swiss-Prot, GenProt are becoming repositories for storing protein structure. A web site for one of them, the Human Protein Reference Database (HPRD), is shown in Figure 1-4 (Peri, 2003). The HPRD contains protein data aggregated from the contribution of over 70 laboratories worldwide. Nearly 45,000 papers were reviewed in developing this database. The database also represents the results of nearly 3,000 experiments.

Protein-Protein Interactions

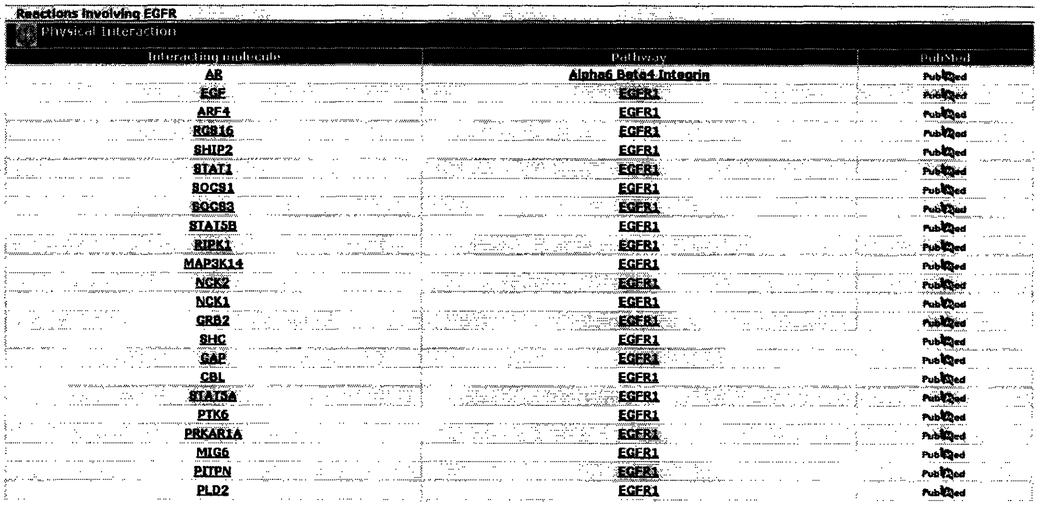
Using this database one can, for example, query to find a particular protein such as EGFR (or Epidermal Growth Factor Receptor) as shown in Figure 1-5. Once the EGFR entry is found, one can then search for the proteins that interact with EGFR, as shown in Figure 1-6. This list shows a small subset of the 60 proteins



Protein Name		EGF receptor	
Gene Symbol	EGFR	Accession Number	NetPath_M1956
Summary			
Alternate Names		Epidermal growth factor receptor isoform d Epidermal growth factor receptor isoform b Epidermal growth factor receptor isoform c EC 2.7.1.112 Epidermal growth factor receptor isoform a ERBB1 mENA p170 Receptor tyrosine protein kinase ErbB-1 Truncated epidermal growth factor receptor Species antigen 7	

Figure 1-5: The entry for the EGFR entry in HPRD.

which interact with EGFR. This database also contains references to the published research articles which detail the experiments documenting the particular protein-protein interaction. For example, we can find the reference to the published article documenting the EGFR-STAT1 protein-protein interaction, as illustrated in Figure 1-7 (Xia, 2002).



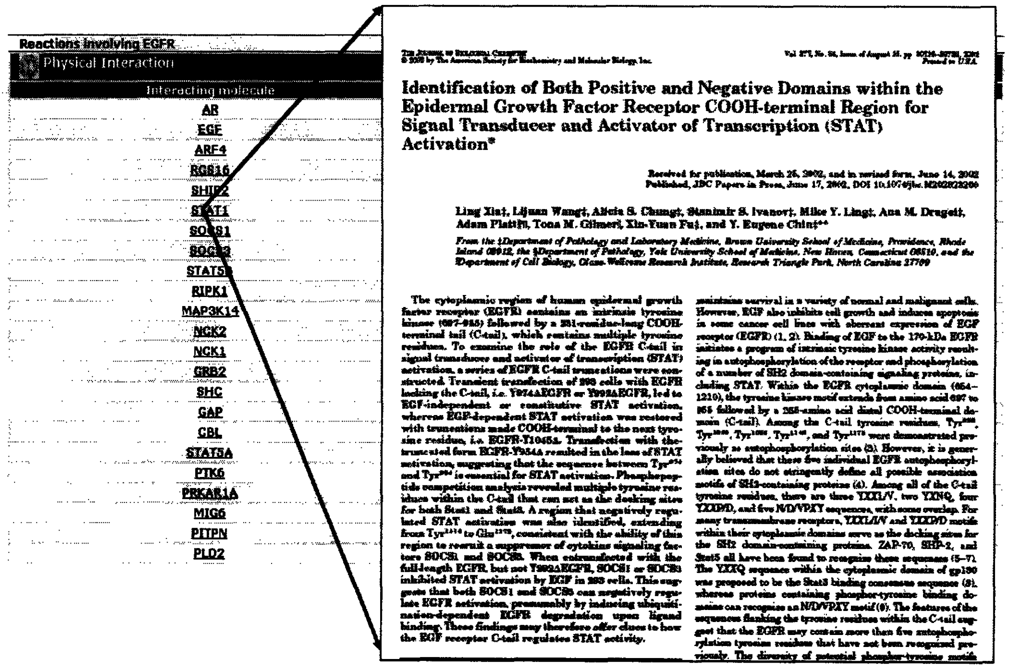
Reactions involving EGFR

Physical Interaction

Interacting molecule	Pathway	PubMed
AR	Alpha6 Beta4 Integrin	PubMed
EGF	EGFR1	PubMed
ARF4	EGFR1	PubMed
RGS16	EGFR1	PubMed
SHIP2	EGFR1	PubMed
STAT1	EGFR1	PubMed
SOCS1	EGFR1	PubMed
SOCS3	EGFR1	PubMed
STAT5B	EGFR1	PubMed
RIPK1	EGFR1	PubMed
MAP3K14	EGFR1	PubMed
NCK2	EGFR1	PubMed
NCK1	EGFR1	PubMed
GRB2	EGFR1	PubMed
SHC	EGFR1	PubMed
GAP	EGFR1	PubMed
CBL	EGFR1	PubMed
STAT5A	EGFR1	PubMed
PTK6	EGFR1	PubMed
PRKAR1A	EGFR1	PubMed
MIG6	EGFR1	PubMed
PITPN	EGFR1	PubMed
PLD2	EGFR1	PubMed

Figure 1-6: Display of a subset of the total of 60 proteins in HPRD which interact with EGFR.

Approximately 40,000 protein-protein interactions are documented today. Each day, new protein-protein interactions are found. In addition, each day, updates



Reactions involving EGFR

Physical Interaction

Interacting molecule
AR
EGF
ARF4
RGS16
SHIP2
STAT1
SOCS1
SOCS3
STAT5B
RIPK1
MAP3K14
NCK2
NCK1
GRB2
SHC
GAP
CBL
STAT5A
PTK6
PRKAR1A
MIG6
PITPN
PLD2

The Journal of Biological Chemistry
© 2002 by The American Society for Biochemistry and Molecular Biology, Inc.

Vol 277, No. 34, Issue of August 23, pp. 30716–30723, 2002
Printed in U.S.A.

Identification of Both Positive and Negative Domains within the Epidermal Growth Factor Receptor COOH-terminal Region for Signal Transducer and Activator of Transcription (STAT) Activation*

Received for publication, March 26, 2002, and in revised form, June 14, 2002
Published, JBC Papers in Press, June 17, 2002, DOI 10.1074/jbc.M202823200

Ling Xia‡, Lijuan Wang‡, Alicia S. Chung‡, Stanimir S. Ivanov‡, Mike Y. Ling‡, Ana M. Dragoi‡, Adam Platt§¶, Tona M. Gilmer¶, Xin-Yuan Fu§, and Y. Eugene Chin‡**

From the ‡Department of Pathology and Laboratory Medicine, Brown University School of Medicine, Providence, Rhode Island 02912, the §Department of Pathology, Yale University School of Medicine, New Haven, Connecticut 06510, and the ¶Department of Cell Biology, Glaxo-Wellcome Research Institute, Research Triangle Park, North Carolina 27709

The cytoplasmic region of human epidermal growth factor receptor (EGFR) contains an intrinsic tyrosine kinase (687-955) followed by a 231-residue-long COOH-terminal tail (C-tail), which contains multiple tyrosine residues. To examine the role of the EGFR C-tail in signal transducer and activator of transcription (STAT) activation, a series of EGFR C-tail truncations were constructed. Transient transfection of 293 cells with EGFR lacking the C-tail, i.e. Y974ΔEGFR or Y992ΔEGFR, led to EGF-independent or constitutive STAT activation, whereas EGF-dependent STAT activation was restored with truncations made COOH-terminal to the next tyrosine residue, i.e. EGFR-Y1045Δ. Transfection with the truncated form EGFR-Y954Δ resulted in the loss of STAT activation, suggesting that the sequence between Tyr^{954} and Tyr^{994} is essential for STAT activation. Phosphopeptide competition analysis revealed multiple tyrosine residues within the C-tail that can act as the docking sites for both Stat1 and Stat3. A region that negatively regulated STAT activation was also identified, extending from Tyr^{1114} to Glu^{1172}, consistent with the ability of this region to recruit a suppressor of cytokine signaling factors SOCS1 and SOCS3. When cotransfected with the full-length EGFR, but not Y992ΔEGFR, SOCS1 or SOCS3 inhibited STAT activation by EGF in 293 cells. This suggests that both SOCS1 and SOCS3 can negatively regulate EGFR activation, presumably by inducing ubiquitination-dependent EGFR degradation upon ligand binding. These findings may therefore offer clues to how the EGF receptor C-tail regulates STAT activity.

maintains survival in a variety of normal and malignant cells. However, EGF also inhibits cell growth and induces apoptosis in some cancer cell lines with aberrant expression of EGF receptor (EGFR) (1, 2). Binding of EGF to the 170-kDa EGFR initiates a program of intrinsic tyrosine kinase activity resulting in autophosphorylation of the receptor and phosphorylation of a number of SH2 domain-containing signaling proteins, including STAT. Within the EGFR cytoplasmic domain (654–1210), the tyrosine kinase motif extends from amino acid 687 to 955 followed by a 255-amino acid distal COOH-terminal domain (C-tail). Among the C-tail tyrosine residues, Tyr^{992}, Tyr^{1068}, Tyr^{1086}, Tyr^{1148}, and Tyr^{1173} were demonstrated previously as autophosphorylation sites (3). However, it is generally believed that these five individual EGFR autophosphorylation sites do not stringently define all possible association motifs of SH2-containing proteins (4). Among all of the C-tail tyrosine residues, there are three YXXL/V, two YXNQ, four YXXP/D, and five N/D/VPXY sequences, with some overlap. For many transmembrane receptors, YXXL/I/V and YXXP/D motifs within their cytoplasmic domains serve as the docking sites for the SH2 domain-containing proteins. ZAP-70, SHP-2, and Stat5 all have been found to recognize these sequences (5–7). The YXXQ sequence within the cytoplasmic domain of gp130 was proposed to be the Stat3 binding consensus sequence (8), whereas proteins containing phosphor-tyrosine binding domains can recognize an N/D/VPXY motif (9). The features of the sequences flanking the tyrosine residues within the C-tail suggest that the EGFR may contain more than five autophosphorylation tyrosine residues that have not been recognized previously. The diversity of potential phosphor-tyrosine motifs

Figure 1-7: Example of a published article in Journal of Biological Chemistry documenting the identification of a protein-protein interaction between EGFR and STAT (Xia, 2002).

of knowledge are made to existing protein-protein interactions. Experiments are used to derive such protein-protein interactions since these interactions cannot be derived from first principles. Moreover, sometimes, different research teams may get differing results for the same pair of protein interactions.

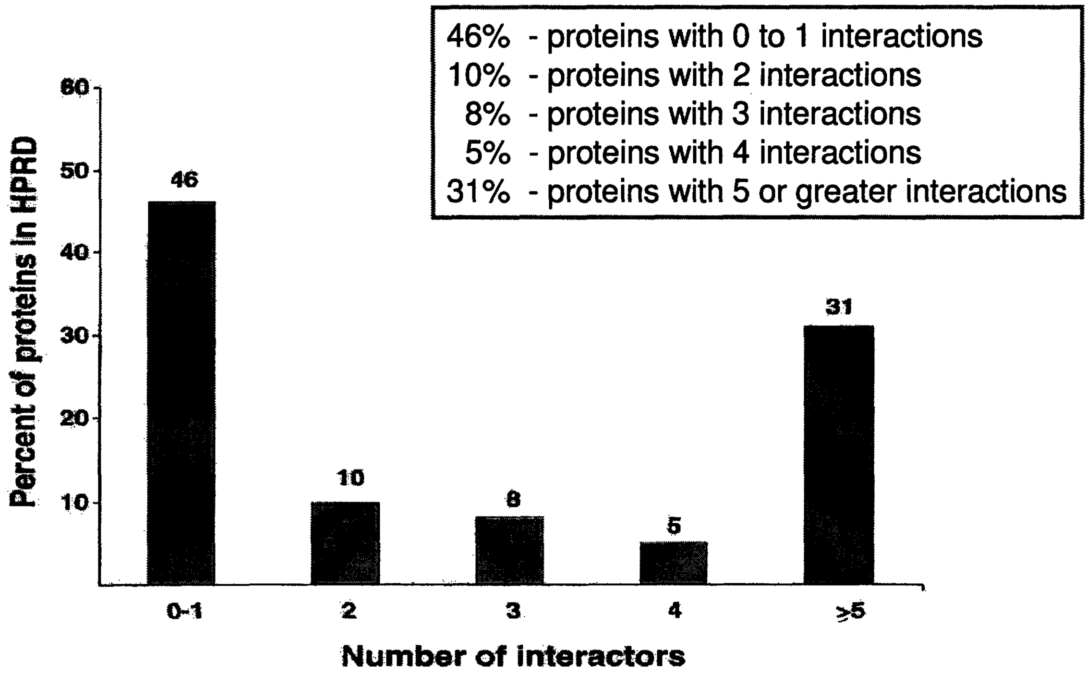


Figure 1-8: Distribution of the number of interactors within the HPRD (Peri, 2003).

Within the HPRD, as shown in Figure 1-8, is the distribution of interactors, or the number of protein-protein interactions of a single protein. In this diagram, it is of interest to note that the majority of proteins either have 0 to 1 interactors or 5 or greater interactors. This means, for example, 10% of the protein-protein interactions involve one protein interacting with two other proteins (the second bar in the above bar chart).

1.2 Biological Pathways

Biological pathways are networks of protein-protein interactions. Single protein-protein interactions can be combined to build biological pathways. Biologists, in addition to understanding the nature and function of genes, proteins and protein-protein interactions, also perform experiments to discover biological pathways. Today, approximately 60,000 biological pathways are recorded across a variety of databases including KEGG, Science STKE, Nature PID, BioPax, BioCarta and others. New biological pathways are being discovered each day. A biological pathway consists of two elements: (1) molecules (also known as

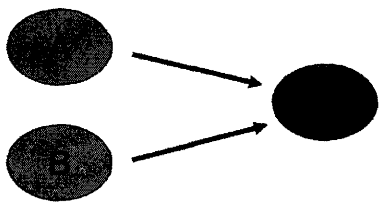


Figure 1-8: Combination of A and B to P.

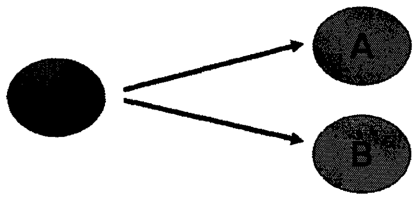
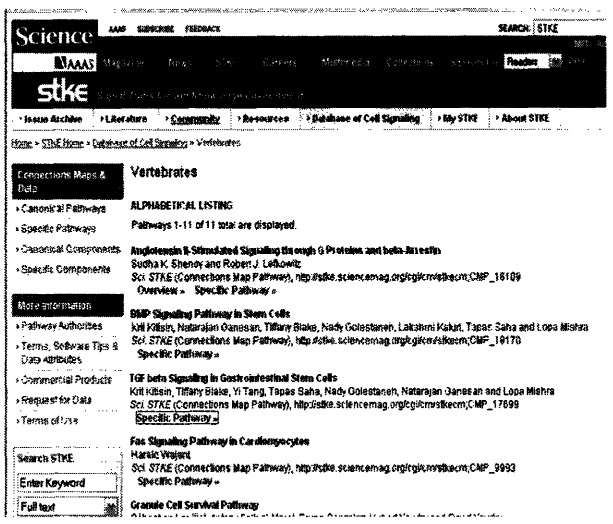


Figure 1-9: Degradation of P to A and B.

molecular species or species) and (2) interactions among those molecules. A biological pathway is visually represented using a "ball and stick" diagram. Each "ball" denoted by circles, rectangles or other geometric shapes represent the individual molecules. Each "stick" denoted by arrows or lines represents the molecular interaction. There are two main types of molecular interactions: (1) *combination* of two or more molecules to create a new molecule as shown in Figure

1-8 or (2) *disassociation* of a molecule to create two other molecules as shown in Figure 1-9.


AAAS SUBSCRIBE FEEDBACK SEARCH: STKE

Issue Archive · Literature · Community · Resources · Database of Cell Signaling · My STKE · About STKE

Home > STKE Home > Database of Cell Signaling > Vertebrates

Connections Maps & Data
- Canonical Pathways
- Specific Pathways
- Canonical Components
- Specific Components

More information
- Pathway Authorities
- Terms, Software Tips & Data Attributes
- Commercial Products
- Request for Data
- Terms of Use

Search STKE
Enter Keyword
Full text

Vertebrates

ALPHABETICAL LISTING

Pathways 1-11 of 11 total are displayed.

Angiotensin II-Stimulated Signaling through G Proteins and beta-Arrestin
Sudha K. Shenoy and Robert J. Lefkowitz
Sci. STKE (Connections Map Pathway), http://stke.sciencemag.org/cgi/cm/stkecm;CMP_18109
Overview » Specific Pathway »

BMP Signaling Pathway in Stem Cells
Krit Kitisin, Natarajan Ganesan, Tiffany Blake, Nady Golestaneh, Lakshmi Kakut, Tapas Saha and Lopa Mishra
Sci. STKE (Connections Map Pathway), http://stke.sciencemag.org/cgi/cm/stkecm;CMP_18170
Specific Pathway »

TGF beta Signaling in Gastrointestinal Stem Cells
Krit Kitisin, Tiffany Blake, Yi Tang, Tapas Saha, Nady Golestaneh, Natarajan Ganesan and Lopa Mishra
Sci. STKE (Connections Map Pathway), http://stke.sciencemag.org/cgi/cm/stkecm;CMP_17699
Specific Pathway »

Fas Signaling Pathway in Cardiomyocytes
Haruki Wajant
Sci. STKE (Connections Map Pathway), http://stke.sciencemag.org/cgi/cm/stkecm;CMP_9993
Specific Pathway »

Granule Cell Survival Pathway

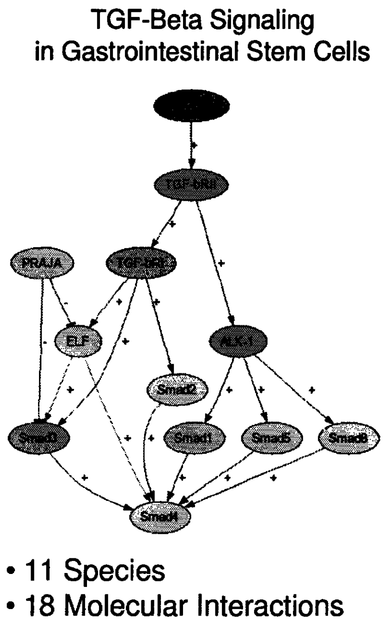


Figure 1-10: Example of a biological pathway diagram containing 11 molecular species and 18 molecular interactions (Ray, 2003).

Figure 1-10 is an example of a biological pathway with many molecular species and many molecular interactions as found in the Science STKE repository (Ray, 2003).

Another example is shown in Figure 1-11. Because experiments not first principles determine the description of a biological pathway (e.g. which molecules will interact with another), biological pathways are constantly changing as new experiments reveal either changes in molecular species or nature of molecular interactions.

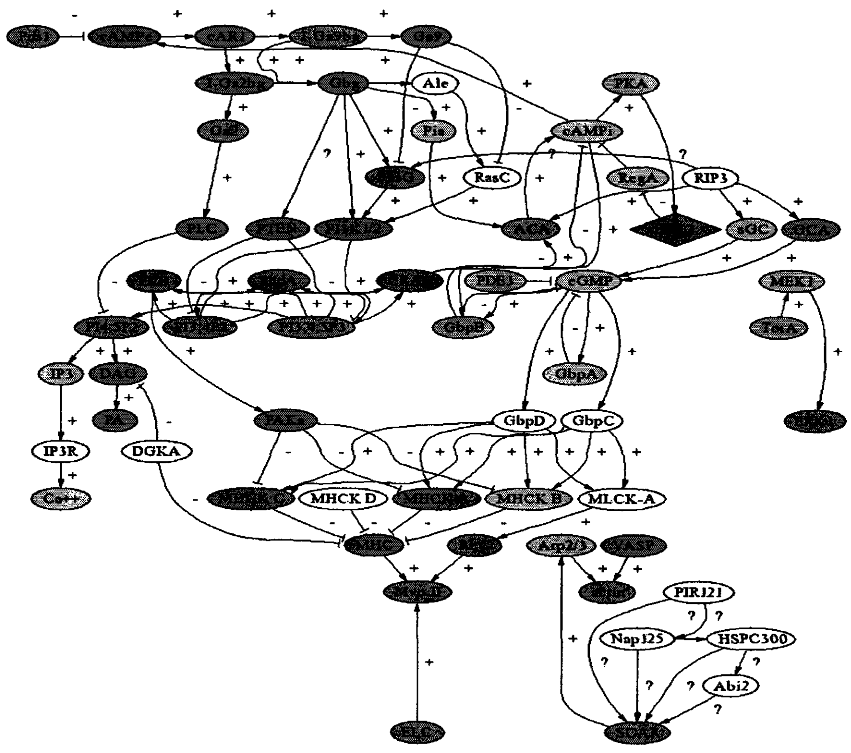
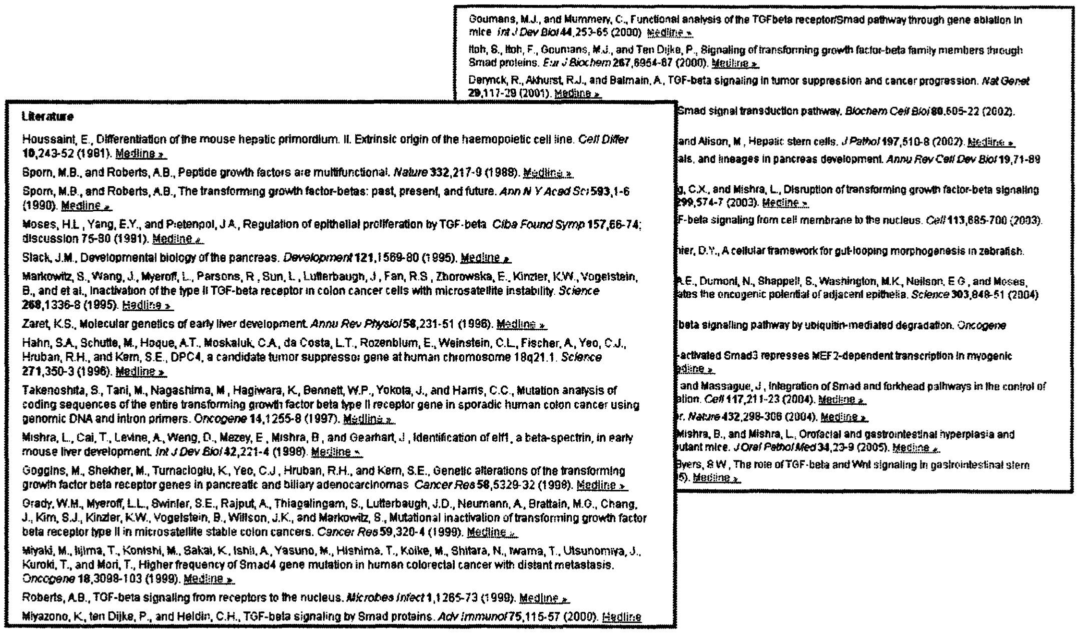


Figure 1-11: Example of a biological pathway diagram containing many molecular species denoted by the geometric shapes of circles, ellipses and diamonds along with the multiple molecular interactions denoted by lines and arrows (Kimmel, 2003).

The biological pathway in Figure 1-10 was the result of aggregating knowledge from nearly 35 different published articles as shown in Figure 1-12. On average each biological pathway consists of approximately 5 to 15 molecular species and approximately 10 to 25 molecular interactions. The development of just this one biological pathway requires the integrated effort of multiple laboratories, spread across the world to construct and maintain. Elements of the biological pathway, the number of molecular species and the types of interactions, are subject to change based on new experimental results.



Goumans, M.J., and Mummery, C., Functional analysis of the TGFbeta receptor/Smad pathway through gene ablation in mice *Int J Dev Biol* **44**,253-65 (2000) Medline »

Itoh, S., Itoh, F., Goumans, M.J., and Ten Dijke, P., Signaling of transforming growth factor-beta family members through Smad proteins. *Eur J Biochem* **267**,6954-67 (2000). Medline »

Derynck, R., Akhurst, R.J., and Balmain, A., TGF-beta signaling in tumor suppression and cancer progression. *Nat Genet* **29**,117-29 (2001). Medline »

Smad signal transduction pathway. *Biochem Cell Biol* **80**,605-22 (2002).

and Alison, M , Hepatic stem cells. *J Pathol* **197**,510-8 (2002). Medline »

als, and lineages in pancreas development. *Annu Rev Cell Dev Biol* **19**,71-89

g, C.X., and Mishra, L., Disruption of transforming growth factor-beta signaling
299,574-7 (2003). Medline »

F-beta signaling from cell membrane to the nucleus. *Cell* **113**,685-700 (2003).

hier, D.Y., A cellular framework for gut-looping morphogenesis in zebrafish.

A.E., Dumont, N., Shappell, S., Washington, M.K., Neilson, E G , and Moses,
ates the oncogenic potential of adjacent epithelia. *Science* **303**,848-51 (2004)

beta signalling pathway by ubiquitin-mediated degradation. *Oncogene*

-activated Smad3 represses MEF2-dependent transcription in myogenic
edline »

and Massague, J , Integration of Smad and forkhead pathways in the control of
ation. *Cell* **117**,211-23 (2004). Medline »

r. *Nature* **432**,298-306 (2004). Medline »

Mishra, B., and Mishra, L., Orofacial and gastrointestinal hyperplasia and
utant mice. *J Oral Pathol Med* **34**,23-9 (2005). Medline »

Byers, S W , The role of TGF-beta and Wnt signaling in gastrointestinal stem
5). Medline »

Literature

Houssaint, E., Differentiation of the mouse hepatic primordium. II. Extrinsic origin of the haemopoietic cell line. *Cell Differ* **10**,243-52 (1981). Medline »

Sporn, M.B., and Roberts, A.B., Peptide growth factors are multifunctional. *Nature* **332**,217-9 (1988). Medline »

Sporn, M.B., and Roberts, A.B., The transforming growth factor-betas: past, present, and future. *Ann N Y Acad Sci* **593**,1-6 (1990). Medline »

Moses, H.L , Yang, E.Y., and Pietenpol, J A., Regulation of epithelial proliferation by TGF-beta *Ciba Found Symp* **157**,66-74; discussion 75-80 (1991). Medline »

Slack, J.M., Developmental biology of the pancreas. *Development* **121**,1569-80 (1995). Medline »

Markowitz, S., Wang, J., Myeroff, L., Parsons, R , Sun, L , Lutterbaugh, J , Fan, R.S , Zborowska, E., Kinzler, K.W., Vogelstein, B., and et al., Inactivation of the type II TGF-beta receptor in colon cancer cells with microsatellite instability. *Science* **268**,1336-8 (1995). Medline »

Zaret, K.S., Molecular genetics of early liver development. *Annu Rev Physiol* **58**,231-51 (1996). Medline »

Hahn, S.A., Schutte, M., Hoque, A.T., Moskaluk, C.A., da Costa, L.T., Rozenblum, E., Weinstein, C.L., Fischer, A., Yeo, C.J., Hruban, R.H., and Kern, S.E., DPC4, a candidate tumor suppressor gene at human chromosome 18q21.1. *Science* **271**,350-3 (1996). Medline »

Takenoshita, S., Tani, M., Nagashima, M , Hagiwara, K., Bennett, W.P., Yokota, J., and Harris, C.C., Mutation analysis of coding sequences of the entire transforming growth factor beta type II receptor gene in sporadic human colon cancer using genomic DNA and intron primers. *Oncogene* **14**,1255-8 (1997). Medline »

Mishra, L., Cai, T., Levine, A., Weng, D., Mezey, E , Mishra, B , and Gearhart, J , Identification of elf1, a beta-spectrin, in early mouse liver development. *Int J Dev Biol* **42**,221-4 (1998). Medline »

Goggins, M., Shekher, M., Turnacioglu, K., Yeo, C.J , Hruban, R.H., and Kern, S.E., Genetic alterations of the transforming growth factor beta receptor genes in pancreatic and biliary adenocarcinomas *Cancer Res* **58**,5329-32 (1998). Medline »

Grady, W.M., Myeroff, L.L., Swinler, S.E., Rajput, A., Thiagalingam, S., Lutterbaugh, J.D., Neumann, A., Brattain, M.G., Chang, J., Kim, S.J., Kinzler, K.W., Vogelstein, B., Willson, J.K., and Markowitz, S., Mutational inactivation of transforming growth factor beta receptor type II in microsatellite stable colon cancers. *Cancer Res* **59**,320-4 (1999). Medline »

Miyaki, M., Iijima, T., Konishi, M., Sakai, K., Ishii, A., Yasuno, M., Hishima, T., Koike, M., Shitara, N., Iwama, T., Utsunomiya, J., Kuroki, T., and Mori, T., Higher frequency of Smad4 gene mutation in human colorectal cancer with distant metastasis. *Oncogene* **18**,3098-103 (1999). Medline »

Roberts, A.B., TGF-beta signaling from receptors to the nucleus. *Microbes Infect* **1**,1265-73 (1999). Medline »

Miyazono, K., ten Dijke, P., and Heldin, C.H., TGF-beta signaling by Smad proteins. *Adv Immunol* **75**,115-57 (2000). Medline

Figure 1-12: Multiple published articles are aggregated to formulate a single biological pathway diagram from the Science STKE repository (Ray, 2003).

In summary, the development of each and every biological pathway is a highly collaborative effort, requiring the aggregation and integration of numerous experimental results, derived from the fundamental understanding of genes, proteins, and protein-protein interactions. Moreover, such a development effort, as will be discussed in forthcoming sections, is not linear, but cyclic, involving constant updates and refinements to the biological pathway, based on new experiments.

1.3 Systems Biology

Systems biology is a new field of biology; however, building systems-level understanding of biology is not a new phenomenon. Over 6,000 years ago, many traditional systems of medicine including Siddha, Unani, Ayurveda and Traditional Chinese Medicine (TCM) proposed systems approaches to describing the whole human physiome (Patwardhan, 2005; Subbarayappa, 1997). During modern times, starting in 1930's, with the concept of homeostasis (Cannon, 1933) and biological cybernetics (Wiener, 1948) attempts were made to understand biology from a systems level using the modern language of physics and control systems theory.

The discovery of the structure of DNA in 1953 (Watson, 1953) combined with recent high-throughput measurement techniques for imaging and quantifying molecular level interactions has enabled a completely new field of biology: systems biology. Systems biology aims to develop system-level understanding by connecting knowledge at the molecular level to higher level biological functions (Kitano, 2000). Such a goal was not possible before. Previous attempts at system-level approaches to biology, whether ancient or modern, were primarily focused on the description and analysis of biological systems, limited to the physiological level. Since these approaches had little to no knowledge of how molecular interactions were linked to biological functions, a *systems biology* of connecting molecular interactions to biological functions was not previously possible. Systems biology, therefore, is a new field of biology as it offers the opportunity, as never before in human history, to link the behaviors of molecules to the characteristics of biological systems. This new field will enable us to eventually describe cells, tissues, organs and human beings within a consistent framework governed by the basic principles of physics (Kitano, 2001). Systems biologists aim to link molecular-level interactions with cellular-level functions through quantitative modeling.

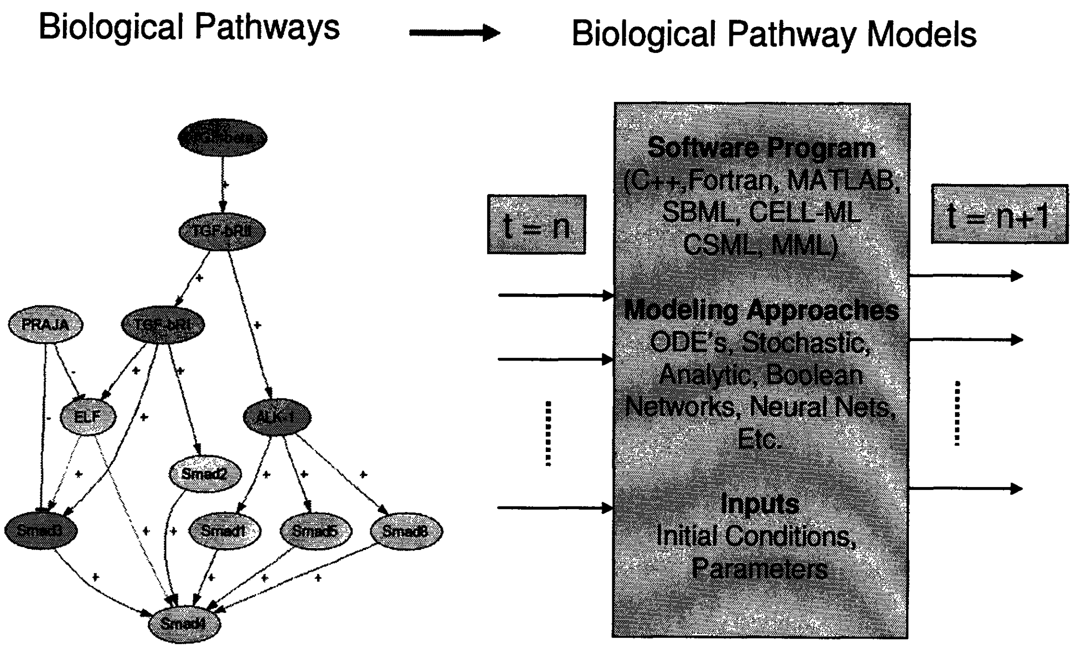
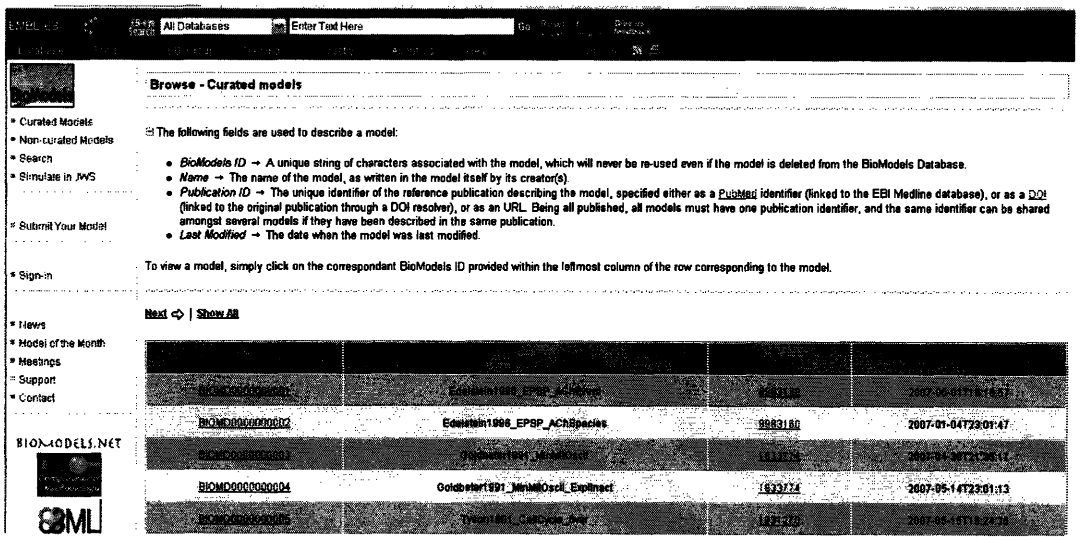

Biological Pathways
Biological Pathway Models
TGF-bRII
PRAJA
TGF-bRI
ELF
ALK-1
Smad2
Smad3
Smad1
Smad5
Smad8
Smad4
t = n
Software Program
(C++,Fortran, MATLAB, SBML, CELL-ML CSML, MML)
Modeling Approaches
ODE's, Stochastic, Analytic, Boolean Networks, Neural Nets, Etc.
Inputs
Initial Conditions, Parameters
t = n+1

Figure 1-13: Systems biologists work to convert biological pathway diagrams to biological pathway models.

Over the past decade, new measurement techniques are enabling biologists to quantify the molecular concentrations and rates of molecular interactions within biological pathways. Such techniques are being used to transform diagrammatic representations of biological pathways to build biological pathway models. Systems biologists convert "ball and stick" diagrams to *biological pathway models,* mathematical representations expressed in software program code along with the inputs and data needed to run the model.

Figure 1-13 illustrates the high-level process by which a biological pathway is converted to a biological pathway model. There are approximately 300 published biological pathway models today. The software program source code of these models may be represented in different formats: MATLAB, C++, C, SBML, CellML, etc. Different mathematical representations including ordinary differential equations (ODE's), Boolean Networks, Stochastic approaches, analytical functions, etc. are used to specify these models. The internal parameters for these models, such as kinetic rate constants, for example, are also determined through experimentation. Maintaining just one biological pathway model is a complicated task since the biological pathway models and the internal parameters are constantly updated based on changes to the biological pathway diagrams (e.g. based on new experiments). There are different emerging repositories for hosting biological pathway models including BioModels.Net (www.Biomodels.net) and CellML.Org (www.cellml.org) (Cuellar, 2003; Le Novere, et al., 2006). A page from the BioModels.net web site is shown in Figure 1-14.



Browse - Curated models

The following fields are used to describe a model:

- *BioModels ID* → A unique string of characters associated with the model, which will never be re-used even if the model is deleted from the BioModels Database.
- *Name* → The name of the model, as written in the model itself by its creator(s).
- *Publication ID* → The unique identifier of the reference publication describing the model, specified either as a PubMed identifier (linked to the EBI Medline database), or as a DOI (linked to the original publication through a DOI resolver), or as an URL. Being all published, all models must have one publication identifier, and the same identifier can be shared amongst several models if they have been described in the same publication.
- *Last Modified* → The date when the model was last modified.

To view a model, simply click on the correspondant BioModels ID provided within the leftmost column of the row corresponding to the model.

Next ⇨ | Show All

BIOMD0000000001	Edelstein1996_EPSP_AChEvent	8983160	[illegible]
BIOMD0000000002	Edelstein1996_EPSP_AChSpecies	8983160	2007-01-04T23:01:47
BIOMD0000000003	Goldbeter1991_MinMitOscil	1833774	[illegible]
BIOMD0000000004	Goldbeter1991_MinMitOscil_Explinact	1833774	2007-05-14T23:01:13
BIOMD0000000005	Tyson1991_CellCycle_6var	1831270	[illegible]

Figure 1-14: BioModels.Net is one repository of biological pathway models (Le Novere, et al., 2006).

From BioModels.Net, one can download a biological pathway model and execute it on a local computer. In Figure 1-15 is a portion of the biological pathway diagram of the EGFR model of Kholodenko (Kholodenko, et al., 1999) fro BioModels.Net.

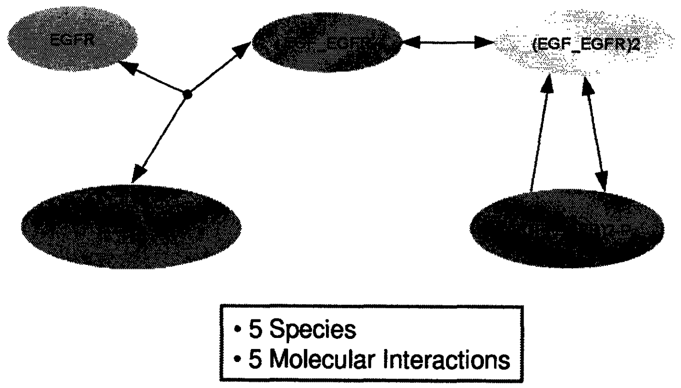


Figure 1-15: Example of a portion of the Kholodenko EGFR model's biological pathway diagram from BioModels.Net (Kholodenko, et al., 1999).

This biological pathway has 5 molecular species and 5 molecular interactions. Figure 1-16 illustrates the transformation of the biological pathway diagram, on the left hand side, to a *black box* biological pathway model, on right hand side. This black box has inputs, being the species concentrations of the molecular species at time t=n, and outputs being the species concentrations of the molecular species at time t=n+1. The internals of the black box contain the software code and the mathematical representation. In this case, the mathematical representation is an ODE and the software code is in SBML. However, the mathematical representation and the software code could be in any

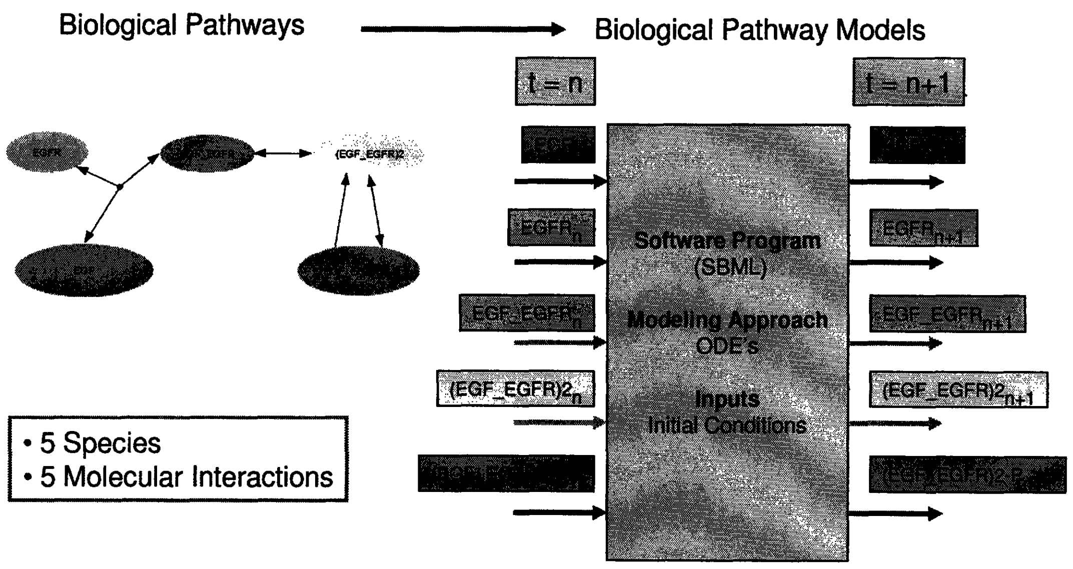


Figure 1-16: Representation of the biological pathway model as a "black box".

format as described earlier. Execution of this biological pathway model will yield results as shown in Figure 1-17.

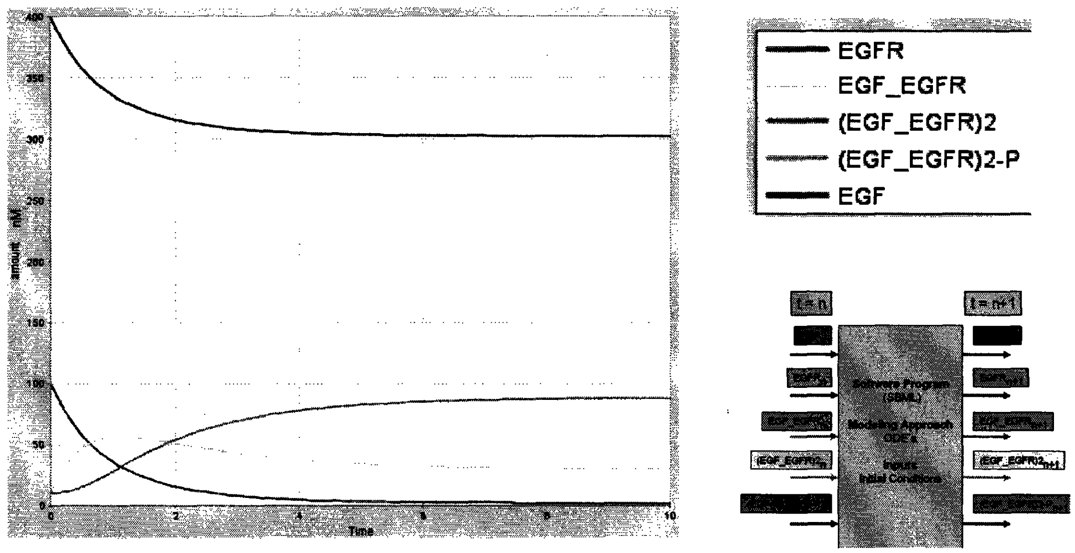


Figure 1-17: Results from the execution of the biological pathway model.

The results in Figure 1-17 are the time varying changes in species concentrations. Along the x-axis is time and along the y-axis is the species concentration in nM (nano-molar). Such a biological pathway model serves to provide a quantitative and predictive capability to describe the interactions of 5 molecular species. Each biological pathway model is treated as a black box, having a defined set of input species and the same defined set of outputs, the values of which are the species concentrations. The construction and validation of such biological pathway models remains a tedious and time-consuming process mainly due to the experimental effort required to determine the many internal parameter values. Most biological pathway models are developed, used for a single application by a single developer, and then forgotten. One can only imagine how many biological pathway models, for example built in MATLAB, and never published are located in some file folder, in some unknown computer, developed by some graduate student. Therefore, considering all the hard work that goes into developing a model, it is rather surprising that so little attention is paid to the presentation and conservation of existing models (Snoep, et al., 2006).

Systems biologists are attempting to create reusable components of biological pathway models by constructing online repositories like BioModels.Net, which offer an archive and curated repository. In addition to supporting the conversion of the diagrammatic representation of biological pathways to biological pathway models, systems biologists also aim to build *larger* biological pathway models by

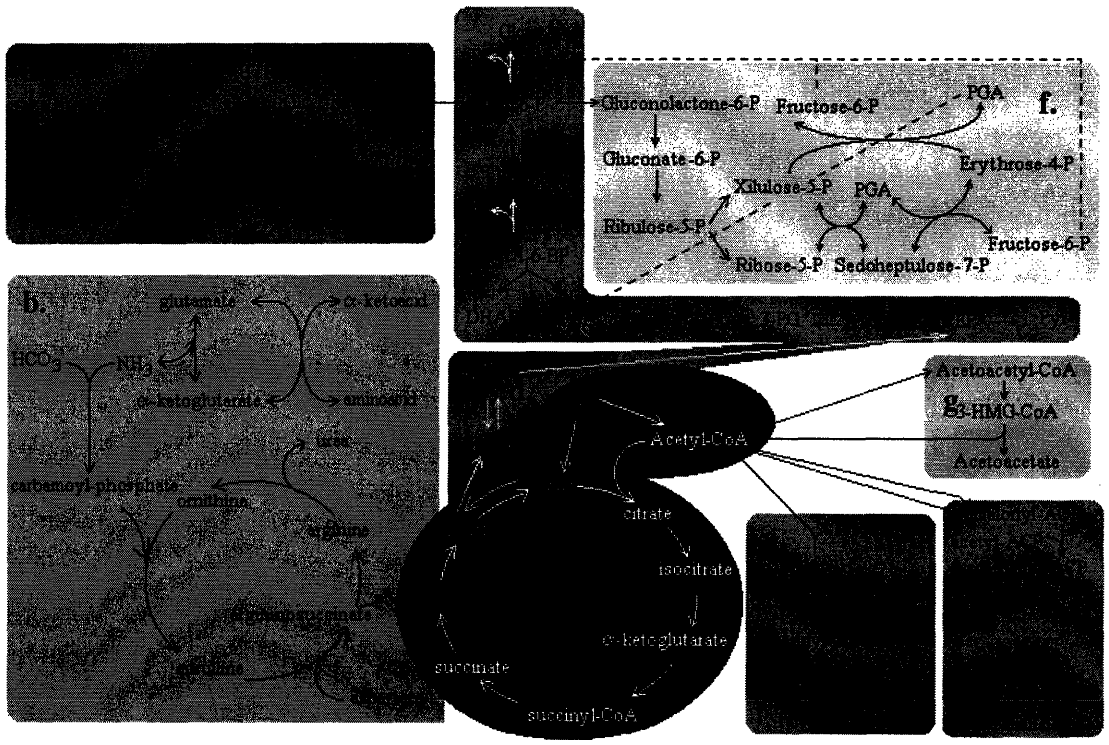


Figure 1-18: An example of how biological pathways interact to support a cellular function. Eight (8) biological pathways including **a.** glycogen metabolism, **b.** amino acid degradation and urea cycle, **c.** glycolysis, **d.** gluconeogenesis, **e.** citric acid cycle, **f.** pentose phosphate, **g.** ketogenesis, **h.** fatty acids beta-oxidation, and **i.** fatty acids synthesis involved in the cellular function of metabolism (Silva, 2002).

integrating *smaller* biological pathway models. The purpose of such effort is to gain new insights on cellular functions not possible from experimental research. For example, Figure 1-18 illustrates the integration of 8 different biological pathways that are part of the cellular function of metabolism. In this example, only the integration of the diagrams of the biological pathways is shown, *not* the integration of the mathematical models for each biological pathway. Figure 1-19 is another example of an integrated biological pathway, represented also

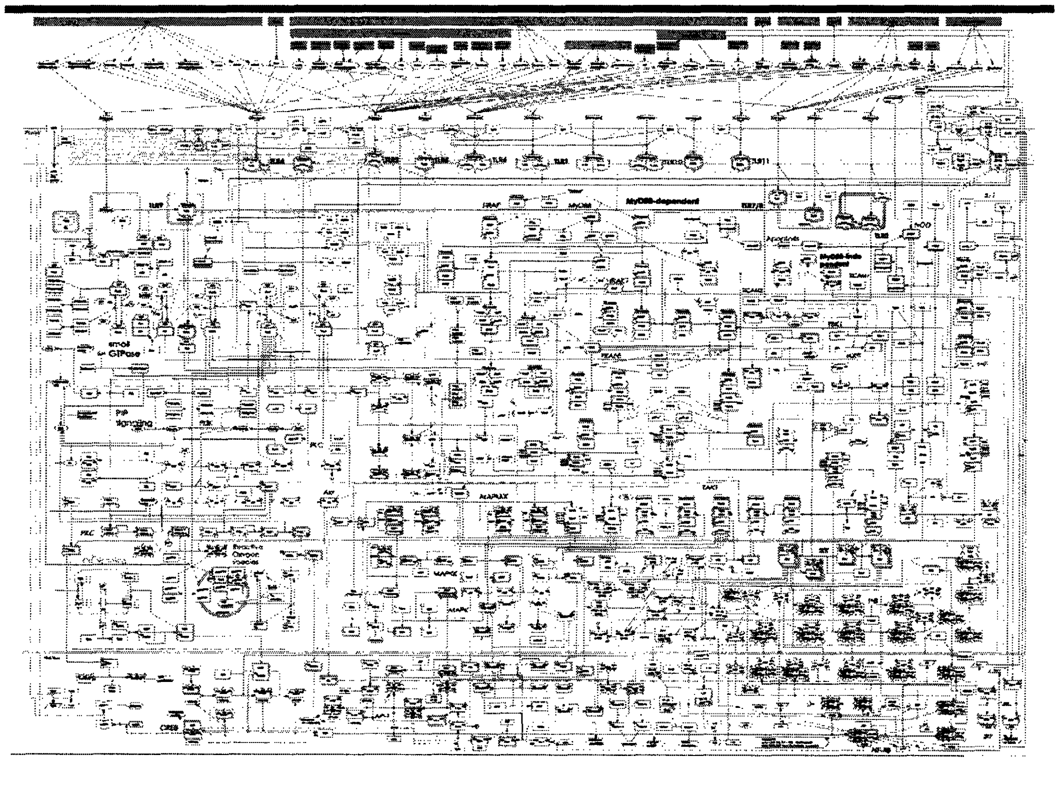


Figure 1-19: Example of a integrated biological pathway constructed from aggregating knowledge from approximately 200 published papers on the TLR signaling network (Oda, 2006).

currently only as a diagram, constructed from the aggregation of multiple biological pathways from across approximately 200 papers (Oda, 2006). These two examples illustrate the integration of *just* smaller "ball and stick" biological pathway diagrams to create larger "ball and stick" biological pathway diagrams. Recently, systems biologists are moving beyond just integrating the diagrams of biological pathways to integrating biological pathway models. Such integration efforts, currently few and less than a handful, are providing higher level system understanding not possible by experiments alone. One such example is shown in Figure 1-20.

The integrated biological pathway model, shown in Figure 1-20, serves to provide an integrated model of the cellular function of osmoregulation. The model is predictive since it suggests previously unrecognized features as confirmed with experiments and serves as a valuable tool for future studies on

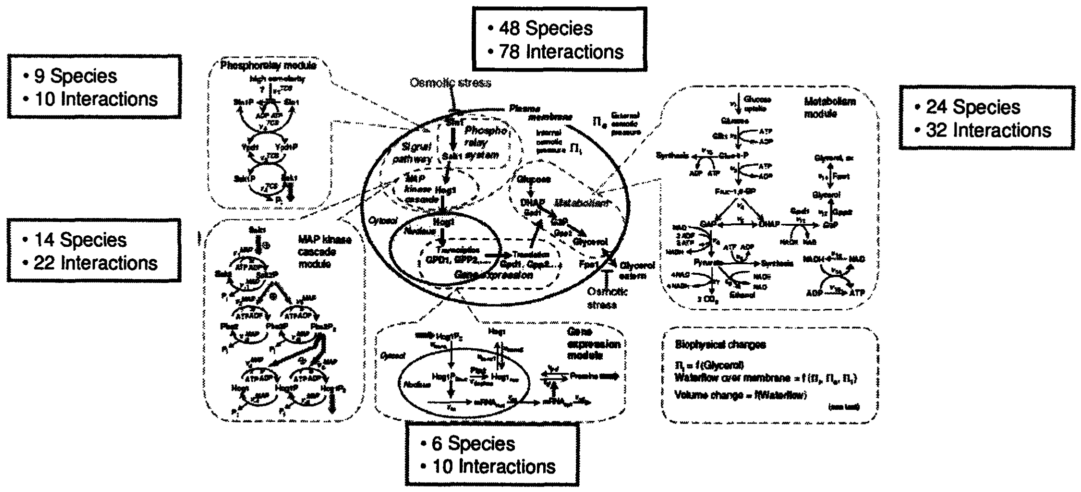


Figure 1-20: Example of an integrated biological pathway model constructed by integrating four different biological pathway models to provide new understanding of the cellular function of osmoregulation (Klipp, et al., 2005).

osmoregulation (Klipp, et al., 2005). Currently, there are approximately 5 to 10 such integrated biological pathway models. There are three main reasons why there is such a low number of integrated biological pathway models. First, the individual biological pathway models are in different formats. Second, understanding any one model requires a great deal of domain specific knowledge and expertise. Third, the primary method of integration involves merging the source codes of each biological pathway model into one large source code. Because of these reasons, it is very time-consuming and expensive to integrate biological pathway models. Maintenance of the resulting integrated model is also very difficult since the integrated model can become invalid as it has a "half-life." New proteins, protein-protein interactions and new parameters (e.g. rate constants) in any one of the individual biological pathway models are being discovered and/or updated constantly. Integrating models can also become especially difficult, within the current method, if the source code for any one particular biological pathway model is not publicly available. This would require one to recode that entire model's source code from scratch.

1.4 Research Motivation

Figure 1-21 illustrates the path to modeling the whole cell and summarizes the concept from the previous sections.

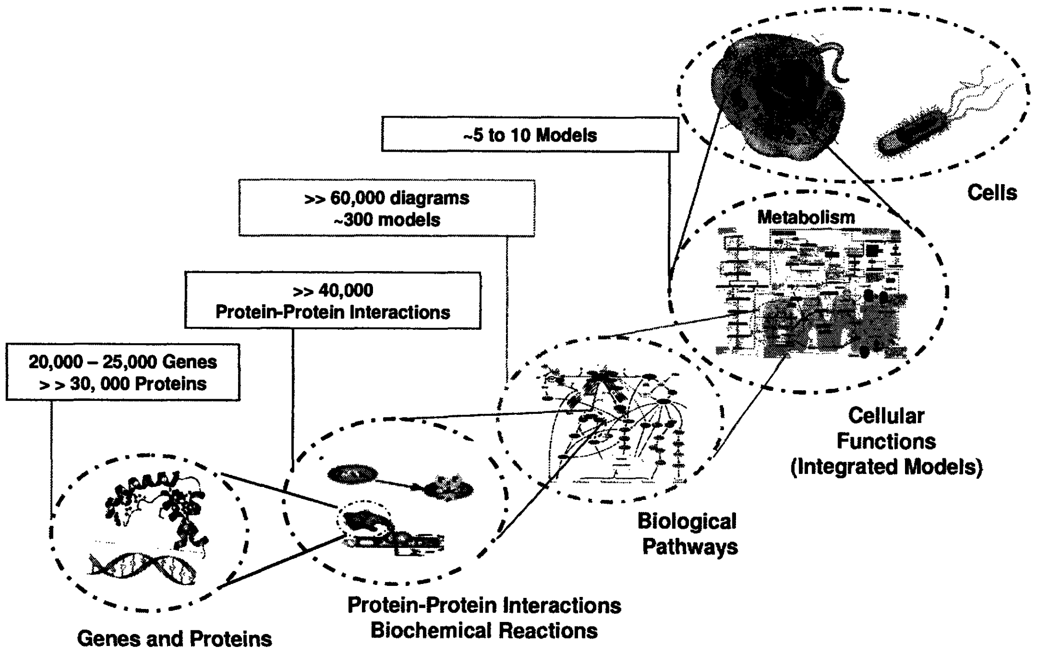


Figure 1-21: Summary of the development path towards whole cell modeling.

This figure presents four major steps to modeling the cell. First, the understanding of genes and proteins are used to build an understanding of protein-protein interactions. Second, these protein-protein interactions are networked to create biological pathways. Third, the integration of biological pathways serves to describe cellular functions. Fourth, and finally, the integration of cellular functions serves to model the whole cell. Currently, as shown in Figure 1-21, there are only 5 to 10 such integrated models of cellular function. It is expected that the number of biological pathway models, currently numbering approximately 300, will grow; however, the step in developing larger integrated models is severely limited by the time consuming and expensive effort, for the three reasons highlighted earlier. The discussion below provides greater insights on the efforts needed to build biological pathway models.

Development of Biological Pathway Models

The process to create and maintain a particular biological pathway model is an iterative process of manipulating, measuring, mining and modeling as shown in Figure 1-22. The two major areas are experimentation and modeling.

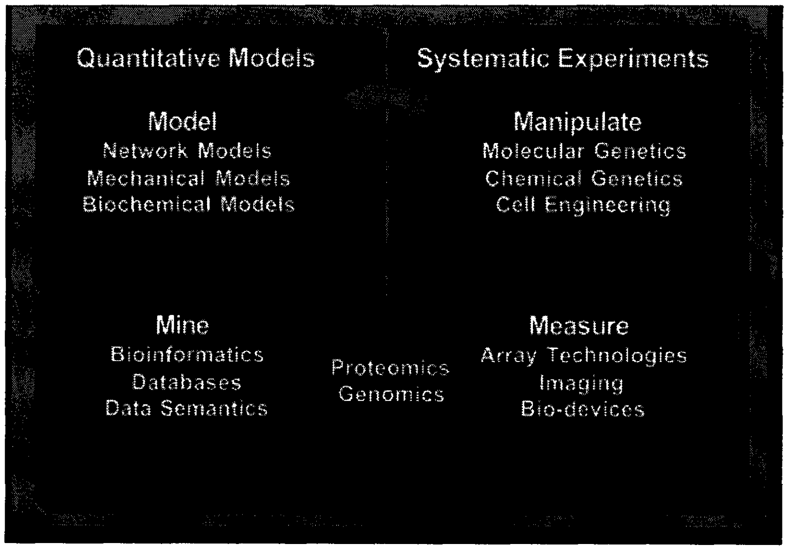


Figure 1-22: The four M's of systems biology (Lauffenburger, 2003).

Systematic experiments involve manipulation and measurement. Manipulation involves modifying an existing biological system. Measurement involves collection of data from that manipulated biological system. Quantitative modeling involves both mining and modeling. Mining enables the identification of underlying relationships in large datasets. These relationships can be used to create predictive mathematical models. Biologists, working in highly domain specific areas, perform systematic experiments, using a range of advanced measurement devices quantify the molecular concentrations and dynamics of molecular interactions. Data mining and modeling efforts are used to refine their conclusions. Biological pathway models are developed and refined through this constant and arduous iterative process. The World Wide Web (WWW) offers a vehicle for scientists to more easily share and publish their biological pathway models. There are thousands of such biological pathway models being published and refined each day by teams of biologists world wide. Figure 1-23, for example, illustrates three different research teams, spread across the globe, performing the iterative process of systematic experiments and modeling to produce biological pathways which are made available and published over the WWW.

Given the decentralized nature of these efforts, the source code of any one biological pathway model may be written and stored in a variety of software programming languages, may be publicly accessible or proprietary. A particular source code is typically built and tested on a particular computer hardware platform, and multiple teams may be involved in maintaining that one source code.

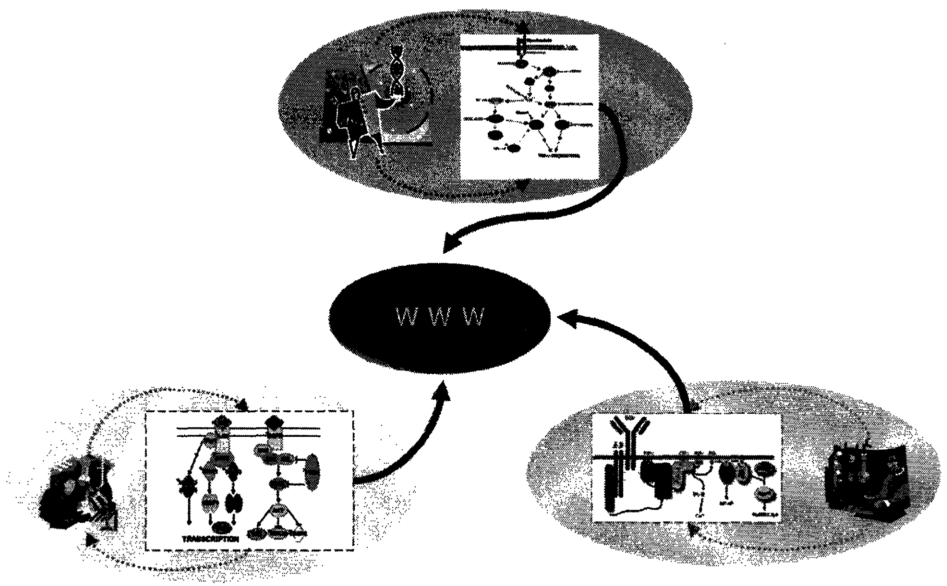


Figure 1-23: Scenario of three research teams performing systematic experiments to produce biological pathway models which are published and made available over the WWW.

Complexity of Integrating Multiple Biological Pathway Models

As the number of biological pathway models and our ability to accurately model any one biological pathway model increases, the challenge becomes how to integrate an ensemble of biological pathway models to build more *complex* models of cellular function. As an aside, this term complex needs to be discussed prior to proceeding. Any one biological pathway model within an integrated model may contain hundreds of species and a set of hundreds of resulting mathematical equations describing those interactions; however, this does not mean the model is necessarily complex. For example, on a personal computer, super computer or even powerful handheld devices, hundreds of simultaneous differential equations can be solved; however, this does not mean that the model we solve is complex just because it has many equations. A complex system, on the other hand, may be complex even if the number of equations is small and apparently simple if the individual elements of the system have their own unique dynamic behavior. Such a system is said to be complex if it has multiple elements which reveal different dynamic properties. This may occur, for example, when all system elements are continuous with concentrated parameters, but the model includes very fast and very slow parts (Raczynski, 1996). Another example is a system where discrete parts interact with continuous sub-models of different speed and different kind such as an electronic circuit that contains integrated circuits as well as electro-mechanical parts such as relays and motors (Bulatwicz, 2006). In other words, the model complexity has little to do with the model size.

Let us now consider a complex biological system: the interferon (IFN) response to virus infection. This integrated system involves various biological pathways, each of which is a unique domain of knowledge and effort for modeling. Figure 1-24 illustrates the four key biological pathways involved in this complex integrated system. Each pathway is developed by different research teams world wide. At the lower left of this figure is the *virus infection pathway*. This biological pathway model simulates the virus infection of a cell and results initially in the up regulation of IFN-Beta, a critical signaling protein; and later on, results in the up regulation of IFN-Alpha, another signaling protein.

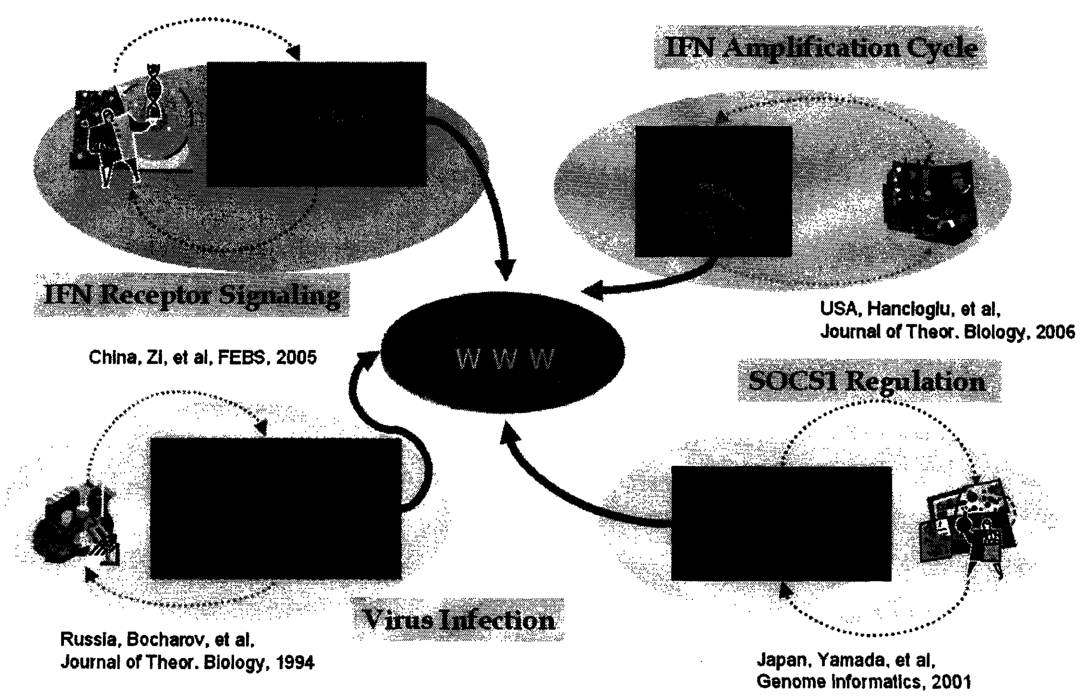


Figure 1-24: Scenario of scientists performing systematic experiments to produce biological pathway models which are published and made available over the WWW.

A second biological pathway model is *IFN receptor signaling,* as shown in the upper left. This biological pathway model represents the interactions of IFN proteins, either IFN-Alpha or IFN-Beta, landing on cell receptors to trigger the activation of other proteins within the cell's cytoplasm to up regulate IRF-7, an interferon regulatory factor. A third pathway is the IFN *amplification cycle,* as shown in the upper right. This biological pathway model simulates the production of increased amounts of both IFN-Alpha and IFN-Beta, which results from the by-product of virus infection with IRF-7. A fourth pathway is SOCS1 *regulation*, shown in the lower right. This pathway serves to regulate the production of IFNs by inhibiting the IFN receptor signaling pathway.

The ensemble of all of the biological pathways depicted in Figure 1-24, if integrated, can provide an integrative model of IFN response to virus infection. Each biological pathway is a contribution of different research teams across three continents of North America, Asia, Europe, and four countries: China, Russia, United States and Japan. Any individual biological pathway model within this ensemble does not have thousands of equations but the activity to integrate these four models to create one new model is unequivocally a complex problem for a number of reasons. First, based on the literature, not all of the biological pathway models depicted in Figure 1-24 have source codes for their models. Second, the biological pathway models were built using different software programming languages. Third, each team developed their biological pathway models on different hardware platforms. Fourth, each of the biological pathway model exhibits different dynamic properties (e.g. different time scales). To integrate these four models is a complex problem. Such a problem is representative of most cellular functions that involve multiple biological pathways which need to be connected to build a larger model.

Complexity of Maintaining an Integrated Model

The above discussion outlined the complexity of creating an integrative model of cellular function from combining various biological pathway models. This complexity is only one part of the problem. Another critical function is: the *maintenance* of the resulting integrated model. In the scenario described in Figure 1-24, each biological pathway model is developed by different teams worldwide. Each team provides a particular domain of knowledge. As discussed in a previous section, the development of any one particular biological pathway model is an iterative process of systematic experimentation combined with quantitative modeling, both supporting each other.

The reality is this: any one biological pathway model is constantly undergoing refinement.

This means that the maintenance of a combined set of biological pathway models can be nearly impossible if there is no easy mechanism to receive and incorporate the updates from each biological pathway model; otherwise, the integrated model's accuracy is only good as its latest update. Since each model is developed in different mathematical and software representations, the integration and maintenance is made even more complicated.

Integration of Biological Pathway Models

Thus far, we have used the term *integrated model* without formally defining it. Moving forward in our discussion, an integrated model refers to a group of biological pathway models executing together that have the ability to affect each other's computations. Based on the previous discussion, the question arises as to how does one effectively integrate an ensemble of biological pathway models and maintain them to ensure reliability.

One way is to avoid the problem entirely and take a completely different approach: develop from scratch an entirely new biological pathway model that encompasses the multiple phenomena across each individual biological pathway model. In essence, create one large biological pathway model. The time and expense, however, involved in developing such a model is prohibitive. The design and implementation of a model requires a combination of software engineering skill and domain expertise. In addition, it involves an extensive amount of verification and validation, requiring the iterative process of systematic experimentation and quantitative modeling as previously discussed.

Another way is to acquire and reuse the source codes from each biological pathway model, and merge them through some mechanism to create one large biological pathway model. While this process may be appearing easier than the previous one, it may not be so. The reuse of software is a key principle of software engineering and is usually achieved by developing a set of simple components, or modules that can be combined in different ways to create more complex components. Ideally scientists would connect their biological pathway models by integrating the existing source codes, treating them as modular pieces that can be easily and quickly plugged together. This, however, can be a very difficult task when the legacy source codes themselves may be poorly understood, and more than likely, were not originally designed to be integrated, and may be written in different software programming languages for different hardware computing platforms.

Despite the potential benefit of building new models from existing ones, integrating pathway models is not a common practice in systems biology community because of the difficulties inherent in working with source codes. Reusing source code in general is difficult for many reasons. Not only are programs difficult to comprehend (a necessary part of any software reuse) but the task of identifying useful source code fragments and integrating these source code artifacts that were not designed for reuse is challenging (Bulatwicz, 2006; Krueger, 1992; Rajlich, 2002). This is especially true for source codes written in unstructured languages and languages that make extensive use of global data (e.g. Fortran) (Bulatwicz, 2006). Reusing source code is particularly difficult because biological pathway models are a unique class of computer programs whose design and use is intertwined with a great deal of domain-level theory outside the model code itself (Robinson, 2004).

In short, the amount of time and expense required to understand the legacy source codes of each individual biological pathway model, prior to merging them may be more costly than starting from scratch. Moreover, once the integrated model is created, the next problem becomes the complexity of maintaining the resulting integrated model, since changes will no doubt take place in the originating biological pathway model's source codes, as they are refined and enhanced, through systematic experiments and modeling.

Perhaps a better way is to integrate biological pathway models in a decentralized manner such that the integrated model functions as one whole, while any of its component biological pathway models can continue to be owned and maintained by its original authors. If this approach is taken, effort will be required to build a new messaging architecture that enables disparately produced biological pathway models to interface, obviating the need to explicitly integrate the source codes. Such an infrastructure would allow scientists to quickly prototype integrated models. This thesis is motivated to create such a computational infrastructure to integrate biological pathway models.

Research Question

The above discussion should have clarified to the reader that biology is fundamentally an experimental science. The development and understanding of genes, proteins, and protein-protein interactions, or just any one element along the path to modeling the whole cell (as shown in Figure 1-21) , is difficult, time-consuming and complex, requiring the collaborative effort of multiple teams of scientists world wide. . This thesis, therefore, poses the following research question:

How can we build larger models from smaller models in a scalable framework to support whole cell modeling given the reality of biology – an experimental science?

1.5 Original Contributions

This thesis summarizes a body of research performed at MIT over the past four years. Original contributions were made in five areas: (1) General Principles and Literature Review, (2) Distributed Computing, (3) Systems Biology, (4) Scientific Visualization, and (5) Computing Architectures

General Principles and Literature Review

- A clear formulation of requirements necessary for a computing architecture to support the building and computing of large scale complex models of the cell and complex cellular functions
- One of the most comprehensive literature reviews to date of the methods used to build integrated models from ensembles of biological pathway models
- A detailed presentation of *why* biology is a complex systems problem along with an exposition of why *ab initio* and monolithic approaches to build whole cell models are intractable and not scalable, respectively.

Distributed Computing

- A novel distributed computing architecture for integrating ensembles of distributed biological pathway models
- Complete implementation of a scalable computational architecture for performing integration of distributed biological pathway models. This initial prototype was implemented using publicly available software tools.
- A unique Controller program that performs the computational integration of an ensemble of individual biological pathway models without requiring any access to the source codes

- An optimal mechanism for orchestrating the activation and parallel processing of calculations across a system of biological pathway models to evaluate an accurate solution
- Creation of the *pathway interface descriptor (PID)* which serves to enable any biological pathway models to be used and accessed by the architecture requiring only minimal information
- An *algorithm for distributed mass balance* which performs the real-time calculation and synchronization of species concentrations in real-time across an ensemble of biological pathway models
- A distributed solver architecture for integrating biological pathway models written in SBML, CellML, or in any other format

Systems Biology

- The first distributed implementation of the EGFR model of Kholodenko which yields results that are consistent with published literature
- Identification of a set of biological pathway models that form the components of the interferon (IFN) response to viral infection
- Integrative modeling and solution of the IFN response to virus infection with both positive and negative feedback

Scientific Visualization

- A detailed choreography of the IFN response dynamics including four three-dimensional (3-D) animation scenes
- A complete 3-D computer graphics video of the IFN response to virus infection. This has also been used as a teaching tool in BE440, one of the core graduate classes in Systems Biology at MIT.

Computing Architectures

- A categorization existing architectures for integrating multiple models
- A quantitative methodology for evaluating architectures for integrating biological pathway models that takes into account the particular needs of different stakeholders
- The development of architectural notation to describe computational architectures for modeling the cell
- A quantitative proof of why a distributed parallel architecture with model reuse is superior to a monolithic approach that relies on merging the source codes of multiple models.

1.6 Organization of Thesis

This thesis contains nine chapters and three Appendices.

Chapter 2: Prior Work

A comprehensive review of the existing methods for integrating biological pathway models is provided in this chapter.

Chapter 3: Methodology

A description of the step-by-step approach to designing, implementing and testing the proposed architecture for integrating biological pathway models is provided in this chapter.

Chapter 4: Architecture

A detailed description of Cytosolve, the scalable architecture for integrating multiple biological pathway models is provided in this chapter. Thee details of the architecture are presented along with details of implementation including initial tests to understand computation time.

Chapter 5: EGFR Model of Kholodenko

The EGFR model of Kholodenko is used to validate and compare the Cytosolve architecture with existing monolithic approaches. Both computation time and accuracy are compared.

Chapter 6: Integrative Model of IFN Response to Virus Infection

In this chapter, we present the results of using Cytosolve to develop the integrative model of the interferon (IFN) response to virus infection using the Cytosolve architecture.

Chapter 7: Quantitative Methodology for Evaluating Architectures for Integrating Biological Pathway Models

This chapter provides a new methodology for evaluating architectures for integrating biological pathway models. This methodology is applied to define various types of architectures, then to evaluate quantitatively their efficacy based on different stakeholder needs.

Chapter 8: Video of the IFN Response to Virus Infection

The storyboards of the four major scenes of the 3-D animation of IFN response to virus infection along with the actual DVD format of the video are provided in this chapter. The scenes include virus infection, up regulation of IFN-Beta, then IFN-Alpha and finally IFN-Gamma.

Chapter 9: Conclusions

A summary of the thesis along with the key findings and future areas of research are provided.

Chapter 2

Prior Work

2.0 Introduction

Integrating biological pathway models is becoming an important area of research for advancing the field of systems biology. In the next section, we review the key factors that are driving this need and some recent efforts to create integrative models of biological systems. In the third section, we survey general computational architectures for integrating models. The fourth section specifically surveys the current approaches for integrating biological pathway models in the field of systems biology. We conclude this chapter by summarizing the current approaches in tabular form. We also provide a discussion on the critical weakness limiting the integration of biological pathway models.

2.1 Movement to Integrate Biological Pathway Models

There is a worldwide movement in the computational systems biology community to find powerful ways to integrate the growing number of biological pathway models. This movement is being driven by a transition from diagrammatic representation of pathways to quantitative and predictive mathematical models, which span time-scales, knowledge domains and spatial-scales (Gianchandani, et al., 2006; Palsson, 2004; Papin, et al., 2005). This transition is being accelerated by high-throughput experimentation which isolates reactions and their corresponding rate constants (Hood, et al., 2004). Vast amounts of information is now available at the level of genes, proteins, cells, tissues and organs, which requires the development of mathematical models that can define the relationship between structure and function at all levels of biological organization (Hunter, 2003).

Systems biology aims to provide a comprehensive quantitative analysis of the manner in which all the components of a biological system interact functionally over time (Hood and Perlmutter, 2004). Such an objective is pursued by an interdisciplinary team of investigators (Aderem, 2005). A significant computational challenge is how we can integrate such sub-cellular models running on different types of algorithms to construct higher order models (Takahashi, et al., 2004).

Biological pathways, including metabolic pathways, protein interaction networks, signal transduction pathways, and gene regulatory networks, are currently represented in over 220 diverse databases. These data are crucial for the study of specific biological processes, including human diseases. Standard exchange formats for pathway information, such as BioPAX, CellML, SBML and

PSI-MI, enable convenient collection of this data for biological research, but mechanisms for common storage and communication are required (Cerami, et al., 2006). However, one the greatest challenges in establishing this systems approach are not biological but computational and organizational (Liu, 2005). The critical need across all domains of molecular and cell biology is to effectively integrate large and disparate data sets (Hwang, et al., 2005).

Vigorous interest in understanding the dynamic aspects of cellular networks is also another driver in the development of integrative techniques for biological pathway models (Endy and Brent, 2001; Sauro, et al., 2003). Such explorations could provide insight into the mechanisms of healthy and diseased cells, as well as a better understanding of how system-level or whole-cell properties emerge from intracellular interactions of molecular components (Lindon, et al., 2006). Moreover, understanding dynamics at the global network level seems to be now a reachable goal which has motivated the growth of systems biology. Also, it is commonly admitted that the study of the network dynamics is able to enlighten the function of genes and groups of genes (Pecou, 2005).

A central question now confronting virtually all fields of biology is whether scientists can deduce from this torrent of molecular data how systems and whole organisms work. All this information needs to be sifted, organized, compiled, and—most importantly— connected in a way that enables researchers to make predictions based on general principles (Pennisi, 2005). Mapping protein interactions and transactions (such as phosphorylation, ubiquitination, and degradation) within a cell or organism is essential to developing a molecular understanding of physiology. Over the past decade, protein interaction mapping has evolved from low throughput manual screens to systematic interrogations of entire proteomes (Bader and Chant, 2006). Reconstitution of biochemical and biophysical processes from 'minimal systems' of proteins has built confidence that top-down and bottom-up approaches to biology meet somewhere in the middle.

Systems biology has sought to integrate these results and data to reverse-engineer an understanding of biological network function and dynamics. The infrastructure for storing and disseminating information on biological systems, and for modeling them, has grown concurrently. In turn, this allows the rapid access and cross-comparison of information that is critical to establishing data quality and creating interoperability standards that will enable biologists to leverage their efforts and build scalable systems (Arkin and Fletcher, 2006).

2.2 Existing Methods

Two critical elements need to be carefully assessed when selecting a modeling approach for any dynamic system: 1. the *level of abstraction* and 2. the *methodology* *of implementation.* In determining which level of abstraction and which methodology of implementation to use, the notions of *tractability, scalability* and *accuracy* are some of the important selection criteria, among others. Tractability is measured by the time and expense needed to design, implement, and test and assess the viability of the modeling approach. Scalability is determined by the ease with which the modeling approach can integrate new components at a particular level of abstraction. Accuracy is determined by the ability of the modeling approach to yield results which match that observed in nature. Different users of the modeling approach will have these and other qualitative criteria in determining which modeling approach are the most optimal for their particular needs.

Currently there are two existing methods towards building whole cell models. The first method proposes to use first principles *(ab initio)* and large-scale computing, as has been applied in other fields such as climatology, particle dynamics, etc. to build a whole cell model. The second method involves downloading and accessing existing models and manually integrating their source codes together by hand to create one monolithic software program: *the monolithic approach.* A variation on this approach is to use semi-automation tools that help one to automatically read and integrate source codes together to create one monolithic software program

First Principles - *Ab Initio*

There are various choices for which level of abstraction to use in modeling the cell. In Figure 2-1 four potential abstractions are illustrated: quantum, atomic, biological pathways, and organelles. In the first principles approach, one could

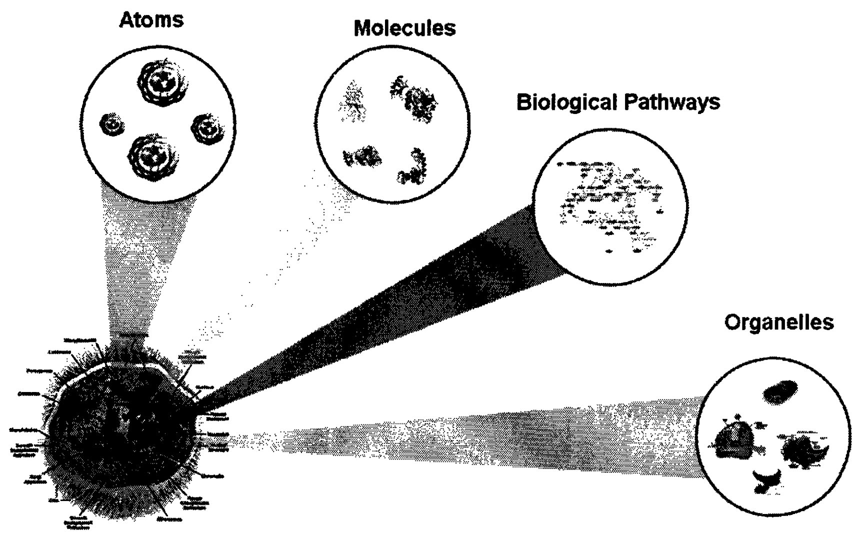


Figure 2-1: Various levels of abstraction in modeling the whole cell.

start at the atomic level of abstraction and solve the time-dependent Schroedinger equation (quantum mechanics, QM) to quantify the dynamics of the whole cell (Vaidehi, 2001). Such an approach would lead to a detailed understanding of the role that atomic level interactions play in determining the fundamental biochemistry of the whole cell. The difficulty in using QM, for example, is that the vast range of length and time scales, from a nitrous oxide molecule to an organelle, makes the QM solution both impractical and useless (Vaidehi, 2001). It is impractical since there are too many degrees of freedom describing the motions of the electrons and atoms, whereas in the functioning of a cell it may only be the rate of transfer across some membrane. The complexity of QM limits its applications to systems with only 10 to 200 atoms (depending on the accuracy), leading to distance scales of less than 20 Angstroms and time scales of femtoseconds. Even the simplest protein in the cell contains over 1000 atoms. While the atomic level abstraction offers high level of accuracy, the level of abstraction is not scalable as the addition of each new atom increases the computational needs exponentially. This potential solution is also impossible, today, as the computing power needed to model the cell using this level of abstraction does not exist.

The first principle method therefore attempts to leap frog some of the steps in modeling the cell as shown in Figure 2-2 by using the laws of physics to model the cell versus experiments, which are the basis of biology. Another choice of abstraction in using the first principles method is at the molecular level, where Newton's equations are used rather than Schroedinger's to model molecular dynamics (or MD). Where in QM the solution is determined by averaging over the scale of electrons to describe the forces on atoms, in molecular dynamics (MD), one averages over the dynamics of atoms to describe the motion of large molecules. While MD provides the ability to predict the dynamic interaction of molecules *ab initio* in an accurate manner, this level of abstraction is neither tractable nor scalable for modeling the whole cell since biological molecules such as proteins have far more atoms, degrees of freedom and numbers of states not encountered in other engineering fields where the species and interactions are well-defined (White, 2007). We consider the simple problem of modeling the interaction of two proteins to demonstrate the intractability of using MD for whole cell modeling.

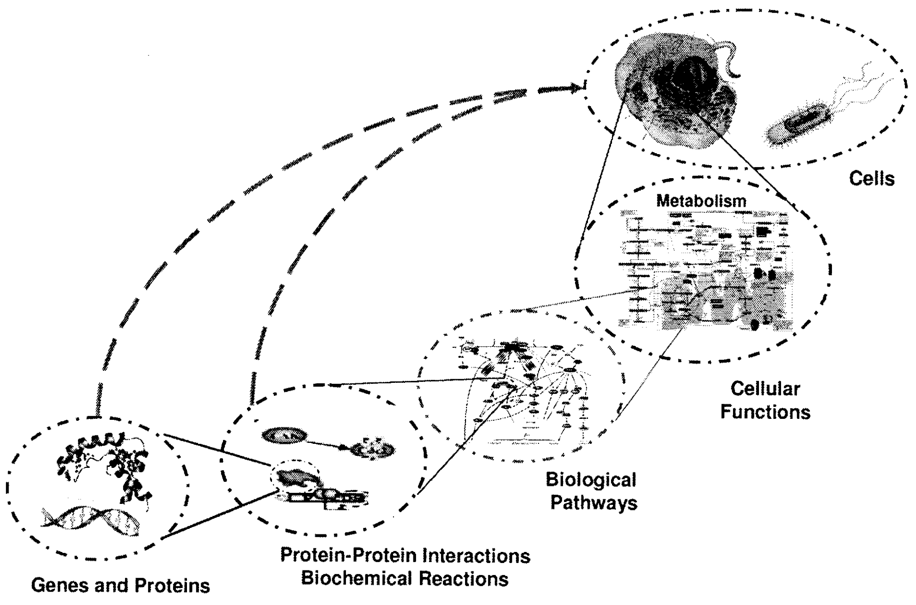


Figure 2-2: First principles approach to modeling the cell by using the laws of physics.

Consider the interaction of two proteins A and B to form the complex AB. We assume that each protein has 100 amino acid residues and each residue has three states (e.g. alpha, beta and other). In MD to model this simple interaction, two key calculations need to take place: 1. Thermodynamic and 2. Kinetic in order to find the most likely transition state of protein A and protein B combining to produce complex AB (Stultz, 2007). For the thermodynamic calculations, MD requires the calculation of thermodynamic properties such as entropy which requires the need to evaluate all the possible states and associated probabilities of protein A, protein B and the complex AB. Protein A will have 3^{100} possible states (100 residues, each of which can be in 3 possible states), protein B will have 3^{100} possible states, and the complex AB can have up to 3^{200} (since AB is a combination of A and B) possible states. Just performing this calculation to determine the states and associated probabilities using modern computers is impossible and therefore intractable.

The kinetic calculation requires the identification of an appropriate reaction coordinate by computing the relative energies (or probabilities) for all the conformations along this reaction coordinate. In this case, it will require determining all the possible conformations of A and B that are at the energy of the activated complex, denoted by A′ and B′, (a higher energy than the energy of A and B); then determining all the possible conformations of A and B within complex AB stage, denoted by A″ and B″, (at an energy lower than that of A and B); and then finally determining all the conformations of AB complex. The kinetic calculation then attempts to link the most probable conformations starting with A and B, then A′ and B′, then A″ and B″, and finally the AB complex to calculate the reaction coordinate. These sets of multiple calculations using atom-by-atom MD to determine the reaction coordinate, as shown in Figure 2-3, to solve even the simple molecular interaction of two proteins is intractable using modern day computers (Stultz, 2007). Moreover, this level of abstraction is not scalable as the number of interactions, number of proteins, and number of atoms

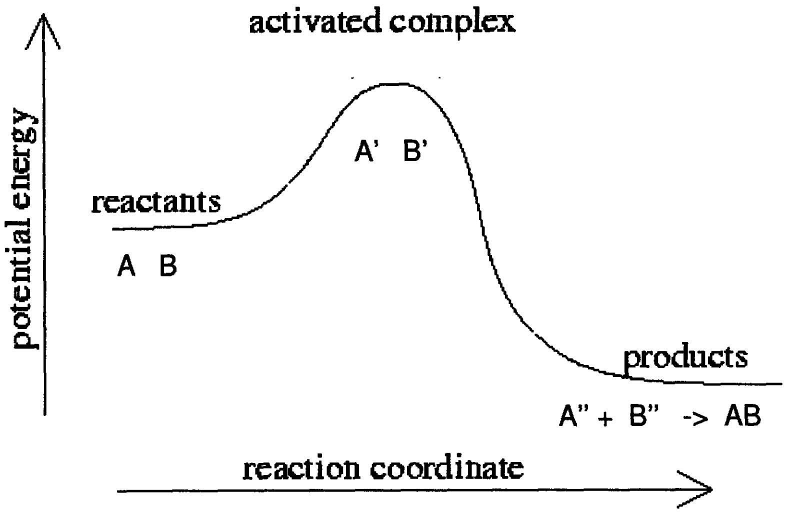


Figure 2-3: Reaction coordinate of A and B reactants going to activated complex A′ and B′, and then to products A″ and B″ which then form the complex AB.

per protein increases. In summary, while MD has powerful applications for determining protein conformations, it is not viable for whole cell modeling where hundreds of thousands of proteins are involved in millions of molecular interactions. Therefore, neither QM nor MD, using the laws of physics, offers tractable approaches to modeling the whole cell.

Biological Pathways as Modules

Another approach towards modeling the cell is to consider biological pathways as being the elemental modules from which complex cellular functions and the whole cell can be modeled. In this section, we present various viewpoints in the existing literature that supports such an approach. Biological systems are thought to have large number of parts almost all of which are related in complex ways (Keller, 2007). Functionality emerges as the result of interactions between many proteins relating to each other in multiple cascades and in interaction with the cellular environment. By computing these interactions, it can be used to determine the logic of healthy and diseased states (Noble, 2006). One way to model the whole cell is through bottom up reconstruction. Such bottom up reconstruction, for example, of the human metabolic network was done primarily through a manual process of integrating databases and pathway models (Duarte, et al., 2007).

It is possible, for example, to regard signaling networks as systems that decode complex inputs in time, space and chemistry into combinatorial output patterns of signaling activity (Bhalla, 2003). By treating biological pathways as modules our minds can still deal with the complexity. In this way, accurate experimentation and detailed modeling of network behavior in terms of molecular properties can reinforce each other (Hornberg, et al., 2006). The goal then becomes that of linking kinetic models on small parts to build larger models to form detailed kinetic models of larger chunks of metabolism, and ultimately of the entire living cell (Snoep, et al., 2006). The value for integrating pathways is that it was found that the integrated network shows emergent properties that the individual pathways do not possess, like extended signal duration, activation of feedback loops, thresholds for biological effects, or a multitude of signal outputs (Klipp and Liebermeister, 2006). In this sense, a cell can be seen as an adaptive autonomous agent or as a society of such agents, where each can exhibit a particular behavior depending on its cognitive capabilities.

Unique mathematical frameworks will be needed to obtain an integrated perspective on these complex systems, which operate over wide length and time scales. These may involve a two-level hierarchical approach wherein the overall signaling network is modeled in terms of effective "circuit" or "algorithm" modules, and then each module is correspondingly modeled with more detailed incorporation of its actual underlying biochemical/biophysical molecular interactions (Asthagiri and Lauffenburger, 2000). The mammalian cell may be considered as a central signaling network connected to various cellular machines that are responsible for phenotypic functions. Cellular machines such as transcriptional, translational, motility, and secretory machinery can be represented as sets of interacting components that form functional local networks (Ma'ayan, et al., 2005).

As biology begins to move into the "postgenomic" era, a key emerging question is how to approach the understanding of how complex biological pathways function as dynamical systems. Prominent examples include multi-molecular protein "machines," intracellular signal transduction cascades, and cell-cell communication mechanisms. As the proportion of identified components involved in any of these pathways continues to increase, in certain instances already asymptotically, the daunting challenge of developing useful models—mathematical as well as conceptual—for how they work is drawing interest (Lauffenburger, 2000).

Multi-scale modeling is essential to integrating knowledge of human physiology starting from genomics, molecular biology, and the environment through the levels of cells, tissues, and organs all the way to integrated systems behavior. The lowest levels concern biophysical and biochemical events. The higher levels of organization in tissues, organs, and organism are complex, representing the dynamically varying behavior of billions of cells interacting together (Bassingthwaighte, et al., 2005). Biological pathways can be seen to share structural principles with engineered networks, along with three of the most important shared principles, modularity, robustness to component tolerances, and use of recurring circuit elements. (Alon, 2003).

An important attribute of the complexity pyramid is the gradual transition from the particular (at the bottom level) to the universal (at the apex) (Kitney, 2007; Oltvai and Barabasi, 2002). Others have recognized that one can build cellular-like structures from a bottom up approach (Seeman, 2002). Integrated models would represent the most compact, unambiguous and unified form of biological hypotheses, and as such they could be used to quantitatively explore interrelationships at both the molecular and cellular levels. (Morgan, et al., 2004). At this time, for instance, the computational function of many of the signaling networks is poorly understood. However, it is clear that it is possible to construct a huge variety of control and computational circuits, both analog and digital from combinations of the cascade cycle (Sauro and Kholodenko, 2004).

2.3 Integrative Modeling Efforts

As discussed in the previous section, systems biology is determined to find new ways to integrate biological pathway models to build larger systems. Neuroscience, for example, seeks such integration of computational models for better understanding of different signaling pathways in neurons (Mishra and Bhalla, 2002).

In the area of metabolism, researchers have created comprehensive mathematical descriptions of the cellular response of yeast to hyperosmotic shock. Their model integrates a biochemical reaction network comprising receptor stimulation, mitogen-activated protein kinase cascade dynamics, activation of gene expression and adaptation of cellular metabolism with a thermodynamic description of volume regulation and osmotic pressure (Klipp, et al., 2005).

The IUPS Physiome Project is an international collaborative open source project intended to provide a public domain framework for computational physiology, including the development of modeling standards, computational tools and web-accessible databases of models of structure and function at all spatial scales and across all organ systems (Hunter, et al., 2005).

For the first time, kinetic information from the literature was collected and used to construct integrative dynamical mathematical models of sphingolipid metabolism (Alvarez-Vasquez, et al., 2004). In another example, a model of 545 components (nodes) and 1259 interactions representing signaling pathways and cellular machines in the hippocampal CA1 neuron were combined. Using graph theory methods, this effort analyzed ligand-induced signal flow through the system. Specification of input and output nodes allowed them to identify functional modules. Networking resulted in the emergence of regulatory motifs, such as positive and negative feedback and feed-forward loops, that process information (Ma'ayan, et al., 2005).

(Oda, 2004) have developed a complete map of the macrophage pathway. In this example, 234 published manuscripts were reviewed and 506 reactions were integrated within the single centralized software framework of Cell Designer (Kitano, et al., 2005) . No models were integrated in this case, rather the work produced a large and complex monolithic diagram interconnecting the various biological pathways.

These examples demonstrate current efforts to integrate models to gain greater insight into a particular area of biology. The results seem promising, and such efforts are only growing. There is an also an equally growing need for foundational tools and architectures that support the continued development of such integrated models in a far more scalable manner. This has led to the development of many new tools such as Cell Designer which aim to offer an easy-to-use interface for linking biological pathway models. In the next sections, we first survey general techniques that are used for integrating such models in order to gain a perspective for reviewing the more specific techniques in computational systems biology.

2.4 Generalized Architectures for Integrating Models

There are computationally two broad approaches to integrating multiple models: monolithic and messaging. The *monolithic approach* involves the creation of one monolithic source code resulting from the merger of the source code of the individual models. The *messaging* approach involves the need for no such merger, but creates a mechanism by which the necessary input and output data streams common across all models can be shared and transferred either statically or dynamically. Some have referred to this messaging approach as a *communications* approach (Bulatwicz, 2006).

Monolithic Approach

There are three broad types of monolithic approaches for integrating models: *manual, semi-automated* and *module-based.*

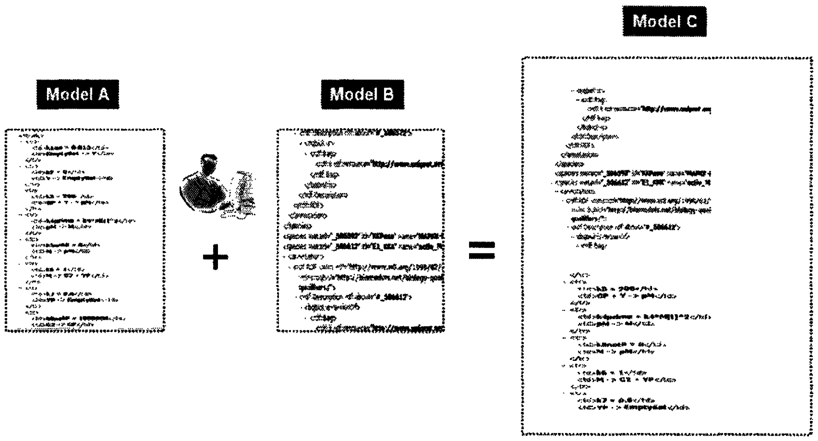


Figure 2-4a: Monolithic approach of cutting and pasting source codes of two models: Model A and Model B to produce a new source code of Model C.

Manual Monolithic Approach

The *manual monolithic* approach is process where the model integrator manually creates a single program or "file" by "cutting and pasting" the source codes or "wiring together" the pathway diagrams of individual models. Figure 2-4a illustrates the cutting and pasting of source codes from two biological pathway models Model A and Model B to produce Model C. Figure 2-4b similarly shows

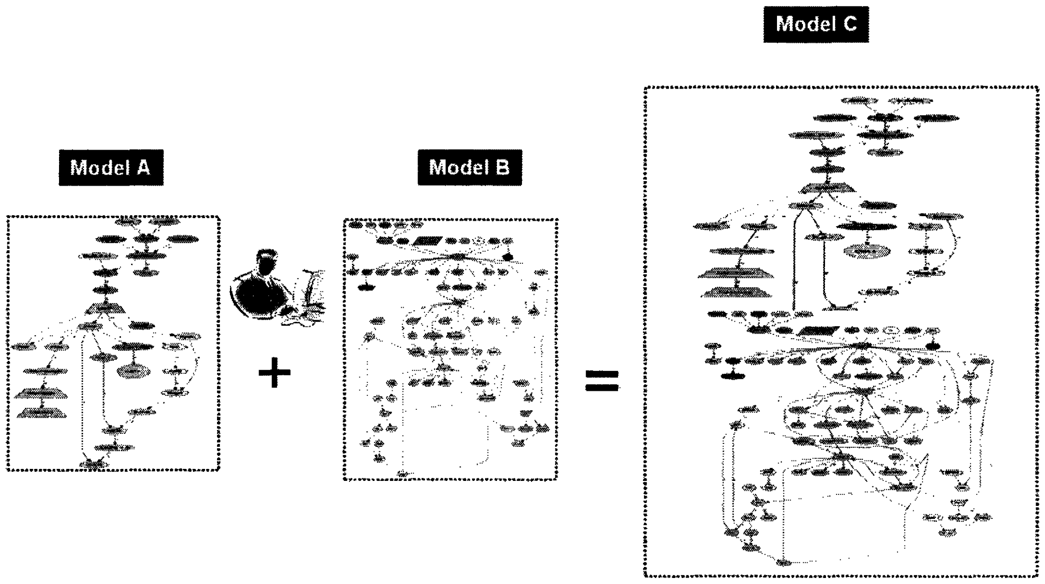


Figure 2-4b: Monolithic approach of wiring the pathway diagrams of two models: Model A and Model B to produce a new pathway diagram of Model C.

how a model integrator may alternatively wire together the pathway diagrams of two pathway models to produce a single pathway diagram using a visual design tool. Many find the manual monolithic approach easy to use. It offers full control to the model integrator of the source codes or pathway diagrams. The model integrator has control of all the coding details (control structure, memory allocation, data types, input/output file formats, etc.) (Bulatwicz, 2006). Although this approach works, it has significant drawbacks. The individual performing the integration needs a complete and detailed understanding of the constituent models. In many cases, the source code is often difficult to obtain since legacy model codes are frequently complex, uncommented, and poorly documented (Bulatwicz, 2006). The single integrated model's source code is also difficult to work with from a software engineering point of view (testing, debugging, verifying, updating, etc.) since it is much larger than its constituent model source codes, and improvements made to the original model source codes must be repeatedly made to each integrated model's source code as well. Examples of recent published papers in using this monolithic approach are shown in Figure 2-5a and Figure 2-5b. In Figure 2-5a, four models are integrated to produce on monolithic model for modeling the cellular function of osmoregulation (Klipp, et al., 2005). In Figure 2-5b, three existing kinetic models are linked with one focusing on yeast glycolysis, a second extending this glycolytic pathway to the glycerol branch, and a third model introducing the glyoxalase pathway (Snoep, et al., 2006).

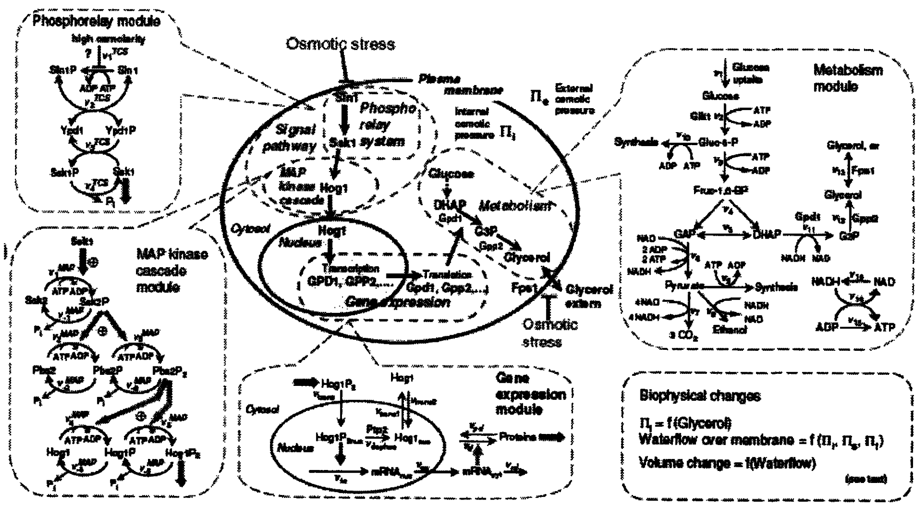


Figure 2-5a: An example of integrating models using the monolithic approach to integrate a model for the cellular function of osmoregulation (Klipp, et al., 2005).

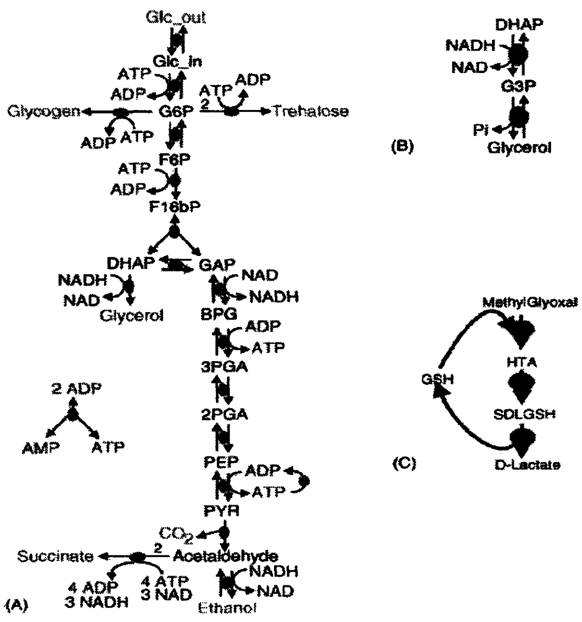


Figure 2-5b: An example integrating three pathways in a monolithic approach (Snoep, et al., 2006).

Semi-Automated Monolithic Approach

The semi-automated approach is a slight variation of the manual monolithic approach. In the semi-automated monolithic approach, software tools are used to accelerate the development of a single program from the individual model's source codes. In these approaches, a software tool, as illustrated in Figure 2-6a, combines together source codes or diagrams using some additional information to produce a single source code or diagram. Rarely do these tools produce the right source code, the first time. There is always manual intervention to review the initially produced output and then perform manual manipulations to produce the final output.

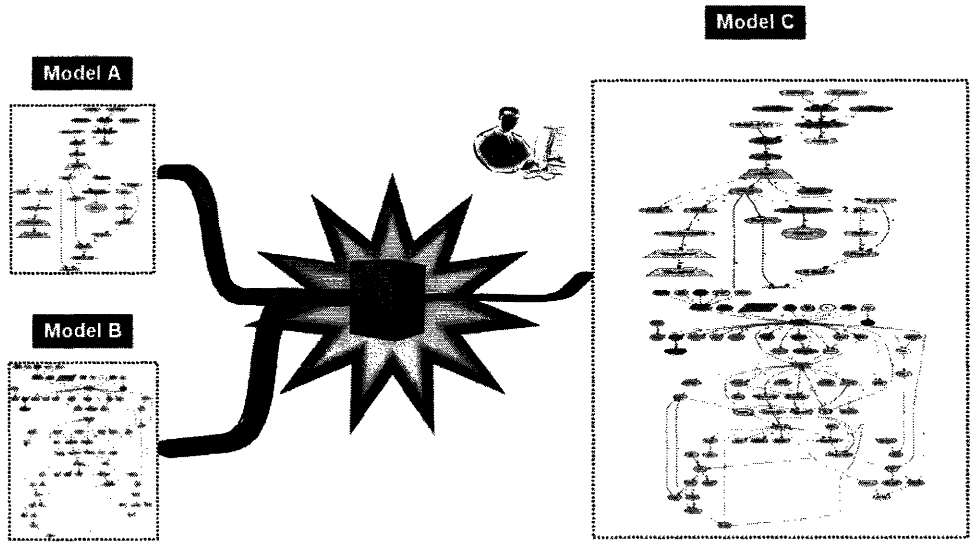


Figure 2-6a: Semi-automated monolithic approach of wiring the pathway diagrams of two models: Model A and Model B to produce a new pathway diagram of Model C.

This semi-automated monolithic approach has the advantage of speeding up the initial merger process of source codes; however, the result varies based on the kind of semi-automation tool being used. In some cases, more effort is spent

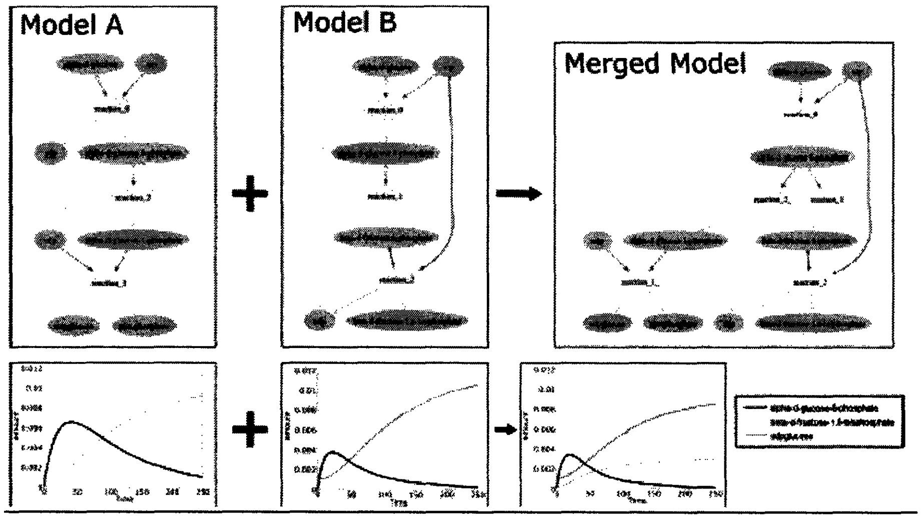


Figure 2-6b: SBMLMerge is an example of a tool that takes two biological pathway models and produces a merged model in a monolithic format (Schulz, et al., 2006).

trying to get the tool itself working properly. Although this approach works, it too has the same significant drawbacks as the manual monolithic approach. One example of such a tool, SBMLMerge, is shown in Figure 2-6b (Schulz, et al., 2006). This tool merges two biological pathway models to produce one biological pathway model. For this tool to operate, both biological pathway models need to be in SBML format and the tool produces one monolithic source code of the merged model in SBML format.

While these semi-automated tools may speed up the process, one disadvantage is that the model integrator does not have as much control of the resulting integrated source code, as they had in the manual approach. Some semi-automated tools insert new proprietary code and data structures along with a new format which may take time to understand and debug. The model integrator, in that event, needs to not only have a complete and detailed understanding of the constituent models and their source code, as in the monolithic approach, but also now needs to understand how the tool for semi-automation itself works.

Module-Based Monolithic Approach

The module-based monolithic approach addresses some of the limitations and drawbacks of the manual and semi-automated monolithic approaches. This approach offers the ability to employ reusable techniques for integrating models.

The module-based monolithic approach results in the creation of single set of source code, but differs in that rather than decompose each model's source codes into blocks of source code designed for integration into another specific model's source code, the scientist decomposes each model's source code into software modules. Modules are subroutines that are reusable and can be written with little knowledge of the other modules, and they can be replaced independently without significant changes to the rest of the program (Bulatwicz, 2006). We use the term reusable to mean that a module can be used in a variety of different situations without any changes made to it. Each module possesses a standard interface for invoking and passing parameters. The modules are then recomposed by connecting their interfaces.

The interfaces for each module are generally simple and consist of a set of input data that must be supplied before the module can be executed, and a set of output data that is available upon completion (Bulatwicz, 2006). The computation that a module performs is encapsulated and hidden within the module. These modules can be classified and organized into searchable libraries and the strict interfaces allow for automatic compatibility checking.

The specification of the module connections can either be through a visual environment or textual configuration files. Modules are connected by specifying which module outputs map to which module inputs. This configurable dependence is a key feature of modules (Bulatwicz, 2006). The module-based monolithic approach is advantageous because simpler modules are easier to test, debug, update, compare, and verify. Once the modules are created, they can be easily assembled, reassembled in different ways, and can be reused to create new compositions. In addition, there are architectures that supply pre-made general purpose modules for common operations. Some architectures standardize the argument data types of the modules while others allow no arguments and require put/get calls (from a custom library) within the component for data input/output, and some use a combination (Bulatwicz, 2006).

Architectures can be general purpose or apply to specific domains such as climatology, hydrology (Bulatwicz, 2006) or systems biology . Those that apply to specific domains typically support the transfer and transformation of domain-specific data types (grids, flux, etc.). Although the module-based monolithic approach addresses the issues of the integrated model's source code complexity (by breaking the computations down into simple components) and reuse (by allowing easy reuse via standard interfaces), it still requires the scientist to have a complete and detailed understanding of the underlying model codes in order to rewrite them into modules. Although the integrated models would be easier to create from modules than from scratch, such an approach to integrating models requires substantial reprogramming.

Messaging Approach

Due to the limitations of the monolithic approaches, scientists turned to approaches that obviate the need to produce either manually, semi-automated manually or even programmatically, one source code from the combined models and approaches that require a substantial programming. This led to the messaging approach. There are two types of messaging approaches: static and dynamic.

Static Messaging Approach

In the static messaging approach, the models remain independent programs and do not affect each other as they are executing. Any one model accepts as input a dataset, which may reside in any variety of formats, and executes to completion to generate an output dataset. That output dataset is then given to another model (perhaps after some transformation) which that model uses it as input and also executes through to completion. This process can then be continued with other models, and they can be executed concurrently if there are no dependencies between their datasets.

Architectures for supporting static messaging architectures offer tools that provide automated ways for the user to select models and datasets, and then specify the distribution of datasets (Bulatwicz, 2006). For example, in one such architecture, called Le Select, a database-oriented approach is used in which both models and datasets are stored in geographically distributed databases and the user specifies the execution and data distribution through textual queries in a standard database query language (Bulatwicz, 2006). Other architectures provide a visual interface to specify the order of execution of the models (Akarsu, 1998; Whelan, 1997) . These architectures typically have different user interfaces and different input/output data formats making them difficult to use, especially for non-specialists, requiring a significant investment of time to learn.

Some architectures provide a standard user interface to each model (Neteler, 2004). This requires that the original user interface source code be removed from each model and replaced with the common user interface source code. To address the issue of non-uniform data input and output formats, some architectures require the user to perform this data transformation manually between model runs while others require the user to change the model codes so that they use a standard data format (Akarsu, 1998; Whelan, 1997). This requires that all the input and output source code be removed and replaced with source code to access the common database and use its data types.

Dynamic Messaging Approach

Using the dynamic messaging approach, the underlying model's source codes remain independently executing programs that interact only by exchanging data via message passing during execution. Architectures that use this approach can be classified by whether or not they include an independent application (a controller) that mediates the execution and messaging between the models.

The primary role of the controller is to transform exchanged data, which typically involves data type conversions, but they also sometimes control the startup of the models or track the global state of the integrated model as well. Architectures that do not include a controller are essentially libraries of data transfer and transformation routines customized for the data types and messaging styles needed by models. Architectures that do include a controller have messaging libraries that support direct model-to-model messaging as well as model-to controller messaging. In either case, these libraries often require the scientist to convert the model data into a standard data type, which is then communicated. All of these architectures require the scientist to perform an initial exercise of creating some mechanism of interfacing with each model through library calls in order to send or receive data.

The user then writes configuration files that specify which models are to execute, and the data that are to be sent and received. The messaging approach avoids the substantial model code rewriting required by the monolithic approaches, but the user still needs a complete and detailed understanding of the model codes in order to properly set up the correct interfacing for them (Bulatwicz, 2006).

In systems biology, for example, if one biological pathway model is written in MATLAB, and other model is in SBML, with each sharing four common variables that need to be communicated to solve the integrated problem, then the interface code is developed for the MATLAB model and for the SBML model. Once this interface code is written, both can transfer data. This interfacing process is often specific to a specific type of integration, so the interfacing process has to be to be repeated for each different type of interfacing.

These architectures also suffer from the problem that the interfacing of the model code (e.g. adding a send() call which sends a value to another model) is separate from the specification of the integration which is done in configuration files (e.g. specifying that one model will be sending a value to the controller and another will be receiving a value from the controller) (Bulatwicz, 2006). This can make setting up an interfacing error-prone because there is no way to perform consistency checks to ensure that the model's source codes are interfacing in a way consistent with the controller's configuration files. This typically isn't a serious problem when the projects that use these integrated architecture are large (often climate-related) and specialists familiar with the models are available to interface with them, but this does become a problem for scientists in other domains seeking to create prototype integrations quickly and easily (Bulatwicz, 2006).

2.5 Approaches to Integrating Biological Pathway Models

In this section, we will review current approaches to integrating biological pathway models within the field of systems biology. Two developments are predominant in addressing this integration problem: 1. the development of new software systems that allow the integration of multiple biological pathway models within a single centralized software framework, and 2. the development of common ***standards*** to define and code biological pathway models.

Software Systems for Integrating Biological Pathways

As discussed in the previous section on *Generalized Approaches to Integrating Multiple Models*, there are two broad classifications for approaches to performing such model integration: monolithic and messaging.

In systems biology, there is another dimension of classification: *informational* and *computational*. Informational approaches are those that provide a way for integrating multiple sources of biological pathway information but do no computing with that integrated information. Computational approaches are those that are a superset of informational architectures by also provide the ability to perform integrated computations across the biological pathways. In the diagram below, we review the predominant software systems for integrating biological pathways and characterize them in this two-dimensional context of monolithic versus messaging and informational versus computational.

Popular Monolithic Tools for Pathway Modeling

There are a set of architectures including Virtual Cell, Kinetikit/Genesis, Cell Designer, Jarnac/JDesigner, JSim, XPAUT, E-CELL, Gepasi, Jarnac, StochSim which are *manual monolithic* and use a *computational* approach. These tools force one to bring all biological pathway models into their particular tool and place into one format prior to integration. They offer easy-to-use user interfaces that makes it convenient for cell biologists to facilitate the construction of models and the generation of predictive simulations from them (Slepchenko, et al., 2003). Some of them use programming markup languages such as SBML (Hucka, et al., 2003). Beyond the ones stated here, there are many other graphical tools and mathematical solvers to construct and solve biological pathway models using a manual monolithic approach. Currently, 136 such software systems exist for constructing biological pathway models. These monolithic systems, such as Cell Designer (Kitano, et al., 2005), allow for the integration of multiple biological pathway models; however, each individual pathway model needs to be loaded and combined within this one centralized framework, and typically each pathway model is assumed to exist in one of the several standard formats such as SBML or CellML. If individual pathway models were developed in other computing systems or other formats, these systems either do not support such integration or make such integration extremely onerous.

MATLAB

MATLAB uses a *module-based monolithic* and provides *computational* approach for integrating biological pathway models. MATLAB offers programming interfaces that can be used to communicate with different models stored in different programs. All models, for all practical purposes, need to be within the MATLAB framework.

PathSys

PathSys uses a *static messaging* and *informational* approach for integrating biological pathway model information. It provides a way for creating a combined database of biological pathways for generating an integrated view of biological mechanisms. It does not offer a way to compute solutions as it is an informational, not computational, mechanism. PathSys has been used to integrate over 14 curated and publicly contributed data sources for the budding yeast (S. cerevisiae) and Gene Ontology. It serves as a general-purpose, scalable, graph-data warehouse of biological information, complete with a graph manipulation and a query language, a storage mechanism and a generic data-importing mechanism through schema-mapping (Baitaluk, et al., 2006). While PathSys provides robust functions for biological pathway model archival and storage including the ability to integrate data from other models source, it is not useful for any computational modeling.

<u>SBMLMerge</u>

SBMLMerge uses a *semi-automated monolithic* and *computational* approach for supporting the integration of biological pathway models written in SBML format only. It provides a tools for combining biological pathway models that must be in the SBML format (Schulz, et al., 2006). By its nature, this approach attempts to force all models into SBML. In fact, ancillary tools are provide as a part of this approach which convert models expressed in CellML into SBML without significant loss of information (Schilstra, et al., 2006).

<u>CellAK</u>

CellAK (for Cell Assembly Kit) is such an agent-based method and follows a *static messaging* and *computational approach.* Systems biology is seeing the emergence of agent-based modeling methods. These methods treat each biological pathway model as a single entity (or agent) obeying its own pre-defined rules and reacting to its environment and neighboring agents accordingly (Pogson, et al., 2006). CellAK has been used to model an abstracted cell, consisting of membrane-bounded compartments with chemical reactions and internal organelles (Webb and White, 2005). It produces models that are similar in structure and functionality to those that can be specified using the systems biology markup language (SBML), and CellML, and implemented using E-CELL (Tomita, et al., 1999), Gepasi, Jarnac, StochSim, Virtual Cell, and other tools currently available to the biology community. This tool does not use differential equations to determine the time evolution of cellular behavior, as is the case with most of the cell modeling systems, since differential equations find it difficult to model directed or local diffusion processes and sub-cellular compartmentalization and they lack the ability to deal with non-equilibrium solutions. This approach offers many positive ways for integrating biological pathway models; however, as with other static messaging approaches for computing, a non-specialist has very high learning curve in preparing a set of biological pathway models for use with this approach.

Cellulat

Cellulat (Gonzalez, et al., 2003) is another *static messaging* and *computational* approach in which a collection of autonomous agents (our active objects—enzymes, transport proteins, lipid bilayers) act in parallel on elements of a set of shared data structures called blackboards (our compartments with small molecule data structures). The dynamics of a Cellulat model result from messages passing between active objects through the blackborad. Agent-based modeling of cells is becoming an area of increasing research interest owing in no small measure to the desire to understand cellular processes at an increasing level of detail.

Cellware

Cellware offers a *module-based monolithic* and *computational* approach. Recent advancements in systems biology have encouraged researchers to move from 'few-reactions-based' to the 'whole-cell-based' and 'whole-organ-based' models. Though the initial efforts have been fairly successful, the task of finding a closest mathematical equivalent of cellular processes has been truly daunting. Cellware, a multi-algorithmic modeling and simulation software, has been developed to precisely address this requirement of the modeling community (Dhar, et al., 2004). The Cellware approach is based on the view that a uniform modeling approach cannot capture the immense diversity of a cell since in the past a number of tools have been developed for cell modeling and simulation which are based on just one mathematical technique (Kierzek, 2002; Le Novere and Shimizu, 2001; Loew, 2002; Mendes and Kell, 2001; Tomita, et al., 1999; You, et al., 2003). Cellware argues that an integrated modeling environment is warranted to model diverse cellular processes using distinct mathematical descriptions. For example, for modeling gene expression a stochastic method is more appropriate while for modeling metabolic pathways a deterministic approach is more suitable. The philosophy behind Cellware is correct for the need for an integrated environment; however, in execution, they are still a module-based monolithic approach computationally.

SigPath

SigPath is a *static messaging* and *informational* approach that connects qualitative information stored in biological databases with the quantitative data required for biochemical modeling approaches (Campagne, et al., 2004). SigPath does not offer a way to compute and integrate biological pathway models.

Gaggle

Gaggle follows *dynamic messaging* and *informational* approach. It attempts through a loose coupling of diverse software and databases to integrate disparate systems to enable simultaneous exploration of experimental data (mRNA and protein abundance, protein-protein and protein-DNA interactions), functional associations (operon, chromosomal proximity, phylogenetic pattern), metabolic pathways (KEGG) and Pubmed abstracts (STRING web resource), creating an exploratory environment useful to 'web browser and spreadsheet biologists', to statistically savvy computational biologists, and those in between (Shannon, et al., 2006). Again, it is important to note that Gaggle does not compute integrated models as it is an informational approach.

Standards for Integrating Biological Pathways

An ancillary approach to support the integration of biological pathways is to promote and enforce standards; however, standards are only as good as their adoption. In the systems biology community, two competing standards are currently being promoted: Systems Biology Markup Language (SBML); and Cell Markup Language (CellML) for coding and storing mathematical models of biomolecular pathways. Approximately 100 biological pathway models are coded in SBML. While the CellML standard receives less mention in the literature than SBML, over 150 models are coded in CellML. CellML models can be converted by sub-sampling the information they contain into SBML, but SBML models are difficult to convert to CellML. In spite of the promotion of these two standards, most mathematical models are coded in programming languages such as FORTRAN, C, MATLAB, etc. and not in any one particular standard format. The development of standards, while valuable, has not resulted in widespread adoption of any one particular standard; however, in biological pathway modeling, the use of SBML and CellML is growing.

Some have attempted to use standards in combination with a manual monolithic approach to build larger biological pathways. One such effort is that of Oda et al. In this work, the diagrams of multiple biological pathways that were coded in SBML were connected together using Cell Designer, which was used as the standard and monolithic software system. This effort served to build a complete map of the macrophage pathway. In this example, 234 published manuscripts were reviewed and 506 reactions were integrated within the single centralized software framework of Cell Designer. No mathematical modeling was executed in this case, but the integration of many pathway diagrams was accomplished.

2.6 Summary

In the previous sections, we have surveyed a number of different architectures that can be used for integrating biological pathway models. In Table 2-1, we summarize these extant approaches based on our two-dimensional classification

Architecture	Monolithic/Messaging	Informational/Computational
Virtual Cell	*Monolithic (Manual)*	*Computational*
Kinetikit/Genesis	*Monolithic (Manual)*	*Computational*
Cell Designer	*Monolithic (Manual)*	*Computational*
Jarnac/JDesigner	*Monolithic (Manual)*	*Computational*
JSim	*Monolithic (Manual)*	*Computational*
E-CELL	*Monolithic (Manual)*	*Computational*
Gepasi	*Monolithic (Manual)*	*Computational*
Jarnac	*Monolithic (Manual)*	*Computational*
StochSim	*Monolithic (Manual)*	*Computational*
PathSys	*Messaging (Static)*	*Informational*
SBMLMerge	*Monolithic (Semi-Automated)*	*Computational*
CellAK	*Messaging (Module-Based)*	*Computational*
Cellulat	*Messaging (Module-Based)*	*Computational*
Cellware	*Monolithic (Module-Based)*	*Computational*
SigPath	*Messaging (Static)*	*Informational*
Gaggle	*Messaging (Dynamic)*	*Informational*
XPAUT	*Monolithic (Manual)*	*Computational*
MATLAB	*Monolithic (Module-Based)*	*Computational*

Table 2-1: Summary of architectures for integrating biological pathways.

methodology from the previous section. The 18 architectures summarized in Table 2-1 represent those architectures that support the integration of multiple biological pathway models. As noted earlier, some of these architectures support the integration of biological pathway model *information* but do not support the computing of the integrated models.

Most of the approaches are monolithic. Clearly within the monolithic approach, the module-based mechanism is an ideal way to construct new models if modules are available, but the approach is impractical for integrating existing models. Cellware and MATLAB offer a module-based monolithic approach; however, they are very difficult to perform the actual programming for this kind of application for a non-specialist, have a significant learning curve, and cannot produce code that is scalable. What is interesting to note; however, is that the most widely use architectures in the systems biology community is the monolithic approach.

The messaging approach allows existing models to be integrated with minimal changes to the model's source codes. Since we are interested in model reuse and scalability, we will focus on the messaging approach in this work. Moreover, we will ignore those architecture that are only *informational* for obvious reasons, since the do not support computation.

Based on Table 2-1, there are only two messaging-style architectures in the current systems biology community: CellAK and Cellulat. These two architectures offer an approach that lets biological pathway models be integrated without the need to explicitly integrate the source codes as in the monolithic approach. Neither of these approaches, however, performs dynamic messaging among the integrated models, which would be even more advantageous.

Weakness of the Current Approaches

As can be seen from the previous discussion, the predominant method for performing such integration is a *monolithic approach*, which involves creating one large biological pathway model through either a manual, semi-automated or module-based method that executes on a single computer. There are many weaknesses which make this approach not scalable.

- First, scaling to, for example, approximately 1,000 pathways - the level required to describe a single cell - would require a massive effort beyond the research and development expended to obtain the original individual pathways (Dewey, 2006).
- Second, each pathway represents a knowledge domain, and it would be essentially impossible to have one person sufficiently knowledgeable in all the scientific areas to understand each of these domains well enough to manually construct a single monolithic program.
- Third, the monolithic approach does not provide a means for pathways from proprietary models to be used with other models that are open

source. The monolithic approach wrongly assumes that the owners of any one model will be freely willing to share their models directly and will not protect proprietary information. The reality is that some models may be public and others, say ones owned by a pharmaceutical company may be private.

- Fourth, most monolithic approaches support only one standard format. This means all other models need to be converted to that standard format.
- Fifth, and related to the previous weaknesses, the monolithic approach wrongly assumes that all models within an ensemble to be integrated are in the same format. The reality is that, while standards efforts are underway, models are coded in software platforms convenient to the author; and more practically any one platform is not capable of modeling all biological pathways. In the (Oda, 2004) example, all pathways had to be constructed in SBML and the integration had to be performed within the centralized framework of Cell Designer. Thus, this monolithic approach demands that in order to model the whole cell, all pathways would have to be placed into SBML or one common standard and then integrated within the framework of a centralized software system such as Cell Designer. Given the reality of standards adoption as aforementioned and the existence of 136 different software systems such as Cell Designer, a monolithic approach does not provide a scalable method to integrate multiple biological pathways to model the whole cell.

- Sixth, the monolithic approach also wrongly assumes that all the models will run on the same computer and/or the same hardware platform. Different models may run on only certain hardware platforms, and more than likely were optimized and tested to run on particular hardware systems. This will become more important as the size and complexity increase , and special coding and computational accelerators are used to control the computational time.
- Seventh, the monolithic approach assumes that all models reside in the same geographical location and ignores that biologists, even in a particular domain area of research such as immune response or cell motility, are distributed across laboratories worldwide. While they may build models within the same software platform and on the same hardware environment, the models themselves may be resident at a completely different location.
- Eighth, the monolithic approach offers no real viable or practical mechanism to *maintain* the single large body of software code emerging from the merger of the software codes of the individual models. Consider four individual models that have been merged using the monolithic approach, and consider what process the author of the monolithic model will need to employ to track and maintain updates and changes to any of the four individual models. To any experienced software development manager, this is known as a *change management nightmare*. The reality is that

any model may change constantly due to any one of several reasons including new advances in measurement, corrections to rate constants or identification of new species in a particular pathway.

- Ninth, the monolithic approach, since it is centrally managed and maintained, places a new burden on the authors of the integrated model: that they become experts in the multiple domains represented by the individual pathway domains that were merged. This is nearly impossible.
- Tenth and finally, many of the monolithic approaches attempt to enforce "standards" as a way to further reinforce a "monotheism". This centralized approach that would not fare well compared with more democratic, community-based approaches that understand and include research-driven development efforts. Creating a rigid standard before a field has matured can result in a failed and unused standard, in the best of circumstances, and, in the worst, can have the effect of stifling innovation (Quackenbush, et al., 2006).

What is Necessary

What is needed is a method that directly addresses this integration and scalability problem by providing a parallel and distributed architecture allowing any individual pathway model to exist in any format and across different computers. Such a method would obviate the need to manually load, understand and interconnect each individual pathway, as is required in monolithic systems. It is clear from the above literature review, the common approach today is a monolithic approach in which computational biologists seeking to create an integrate model "cut and paste" source codes to create on monolithic model. More specifically, of the two recent papers that provide examples of integrating biological pathway models, both involve the merging of SBML models using a manual monolithic approach (Klipp, et al., 2005; Snoep, et al., 2006). Thus, from a short-term perspective, even an architecture which allows the integration of distributed SBML models would be of huge benefit. SBMLMerge offers a tool that is a semi-automated monolithic approach, while valuable for those seeking to "cut and paste" models faster, it is not a scalable solution, since the resulting code base still cannot be maintained as the models change overtime, requiring re-integration each time every time the elemental models change.

A more advantageous solution would be an architecture that could, with minimal effort and no programming, connect or couple the codes of multiple SMBL models, linking their computations. Such a tool would be of immense benefit. If this same architecture could allow for the integration of models also not written in SBML, but in any other format, it will accelerate the development of complex models, versus waiting for all extant models to be converted and curated in SBML. This too will be particularly useful, given the reality that most of the encoded models available in scientific publications or on the Internet are not in a standard format. Of those that are encoded in a standard format, it turns out that most actually fail compliance tests developed for these standards (Le Novere, et al., 2005). In fact markup languages for model encoding such as SBML may not be always the only way for modeling biological pathway models, there are others that provide a range of descriptive and analytical powers. As the field matures, there will be a wider uptake of these alternative approaches for several reasons, including the need to take into account the great complexity of cellular organization (Gilbert, et al., 2006).

From a software engineering perspective, each biological pathway model represents an individual software program, each with different inputs and outputs, written in different programming languages, by different developers, potentially distributed worldwide. Modeling the whole cell, therefore, can be likened to a large-scale systems integration problem.

The research goal of this thesis, therefore, is to develop a new approach to integrate biological pathway models that overcomes the intractability and lack of scalability of existing approaches. In his seminal work, (Brooks, 1975) has demonstrated that the amount of effort to develop such a large-scale system increases exponentially with the amount of additional personnel communications required to coordinate development personnel. The thesis will explore a method that resolves this bottleneck in personnel communications by allowing individual teams to own and manage their own pathway models without being involved in a massive project management effort to coordinate the integration.

Chapter 3

Methodology

3.0 Introduction

This chapter of the thesis is an important transition point. The previous two chapters have provided an introduction to the nature of biological pathway models, general approaches to integrating models, the specific approaches used in the systems biology field, along with a discussion of their weaknesses. At this point, it should be clear to the reader that there is a need for solution for integrating biological pathway models in a scalable manner, and that no extant solution exists for supporting such scalability. Since the goal of this thesis is to create a solution to meet this growing need, this chapter provides a disciplined step-by-step methodology to meet this goal.

3.1 Approach

There are eight steps that will be followed.

Step 1 - Specify Requirements

A solution is only as good as the specified requirements. Prior to designing or implementing the solution, our first step will be to create an itemized list of requirements that the ideal solution will satisfy. This list will serve as the basis for guiding the design of the architecture. In Chapter 4, the section entitled *Requirements Specification,* contains such an itemized list.

Step 2 - Design Architecture

In this step, we develop the design of the computational architecture design. This design will offer diagrams and descriptions to identify the key components of the architecture along with their roles and responsibilities. In Chapter 4, the section entitled *Architectural Design,* contains the details of the architecture's design.

Step 3 - Select Tools

In this step, based on the architectural design, particular tools are selected to implement the architecture. The right tools selection can greatly affect the implementation of the architecture, particularly time and expense. In Chapter 4, the section entitled *Architecture Implementation,* contains the details of the tools that were selected for the implementation.

Step 4 - Implement the Architecture to Produce Initial Prototype

This step involves setting up the hardware, installing the tools, programming the software to implement the architecture based on the requirements specification and architectural design. In Chapter 4, the section entitled *Architecture Implementation* contains the details of the implementation of the initial prototype.

Step 4 - Validate the Architecture by Solving a Known Problem

This step involves validating the architecture by first selecting a biological problem with known solutions, executing that same biological problem using the implemented architecture, and then comparing the results with a monolithic approach. In Chapter 5, the section entitled *EGFR Model of Kholodenko* contains the results of validating the architecture using a known problem.

Step 6 - Identify and Solve a New Problem Using the Architecture

This step involves using the architecture to integrate biological pathway models to solve a complex problem. This step of our methodology is critical to proving the value of the architecture to integrate and solve new problems in a scalable manner. Chapter 6 presents the solution of a complex cellular function, the interferon (IFN) response to virus infection, using the architecture. This Chapter also compares the solution to a monolithic approach of doing the same problem.

Step 7 - Quantitatively Evaluate and Compare Architecture with Other Approaches

Based on the experience of using the initial prototype and with the understanding of how biologists work, it will now be important to assess and analyze the conditions in which a particular architecture is most optimal. In Chapter 7, we develop a methodology to assess architectures for integrating biological pathway models and evaluate the conditions when the distributed and parallel architecture of Cytosolve is best used over the monolithic approach

Step 8 - Propose Enhancements to Architecture and Future Areas of Research

This step involves outlining ways to improve the architecture. The output of Step 4 and Step 5 can be used to enhance the architecture's design and implementation. In Chapter 9, we present the key findings of this research effort and summarize areas of future research.

3.2 Summary

In this section we have presented the methodology that will be used in designing, developing, testing, evaluating the proposed architecture for integrating biological pathway models. This methodology serves to provide a clear path for supporting the research goals of this thesis.

Chapter 4

Architecture

4.0 Introduction

This chapter presents a new scalable architecture for integrating biological pathway models. This architecture is called Cytosolve. The next section specifies the requirements for such an architecture. The third section defines the design of the architecture to meet the requirements. The fourth section describes the set of tools that were selected to meet the design and the requirements to implement the architecture. The fifth section details the implementation effort. The sixth section provides the reader with a simple example to demonstrate how the architecture performs in real-time. The final section summarizes the chapter.

4.1 Architecture Requirements Specification

This section outlines the requirements along with the key decisions that are necessary fro developing a scalable solution for integrating biological pathway models. Ten key requirements are identified below to address the weaknesses of previous approaches.

1. Scalability

If the goal is to model complex cellular systems and eventually the whole cell, the architecture must be able to integrate new pathway models with the same ease as it is to integrate the first one. Scalability is measured by the ease in which additional models can be integrated. Recall that complexity of integration, from our earlier discussion, has little to do with the number of equations in any one model. Two models with numerous equations can be relatively easy to integrate if they are written in the same program, same time scales, in the same domain and developed on the same hardware platform.

2. Opacity of Multiple Knowledge Domains

The architecture must express the property of *opacity*. Opacity means that when one individual is integrating a particular pathway model into a pre-existing ensemble of integrated models, one should not have to know the details and inner workings of all other pathway models. Each pathway may represent a unique knowledge domain, and it would be essentially impossible to have one person sufficiently knowledgeable in all the scientific areas to understand each of the domains.

3. Support for both Public and Proprietary Models

The architecture must support both Public models (models where the source code is readable and available to all) and Proprietary models (models where the source code is inaccessible). The monolithic approach does not provide a means for pathways from proprietary models to be used with other models that are open source. Many pharmaceutical companies, for example, will not want to share the inner source code of their particular proprietary models; however, they are interested in coupling their models with other models to gain better understanding of a larger cellular process. Alternatively, researchers in the academic environment may wish to integrate their Public models with existing Proprietary models to learn some new aspect of science, but cannot currently due to confidentiality issues. By enabling a way to ensure protection of the source code of those Proprietary models, new industry and academic collaborations will be possible with far greater ease. Those with Proprietary models currently chose either not to follow standards to protect their models or if they do follow standards are unwilling to share their model code, which was the reason for the standard itself.

4. Support for Multiple "Standards"

The architecture must support any pathway model code in any format or "standard." While the architecture should support the integration of pathway models constructed in a standard such as SBML, for example, it should be able to communicate with models in any format.

5. Heterogeneity of Integration

At any time models may be in different formats. The architecture should support the ability to integrate, in real-time, models that are in different formats. Thus, if there are 3 models, even if one model is in SBML, another in MATLAB and a third in FORTRAN, the architecture should be able to integrate them with minimal to no effort.

6. Cross Platform Support

The architecture should allow models developed on different hardware and computing environments to be integrated with ease. Different models may run on only certain hardware platforms and more than likely were optimized and tested to run on a particular hardware system. It will be far easier to keep a model resident on a hardware platform for which it was designed and tested for versus having to recode or reconfigure it in any manner to a new hardware platform, which could prove to be very expensive and time-consuming.

7. Independence of Location

The architecture should support integration of models across geographical boundaries. While each model may be on different computers, they may also be physically at different locations anywhere in the world. The model should support protocols for communicating with models anywhere without regard to geographical location.

8. Ease of Maintenance

Any model integrated within the architecture should be able to dynamically change with little to no source re-coding efforts to incorporate changes for that model into the larger integrated model. In the monolithic approach, any change to an individual model typically requires significant recoding and retesting of the integrated model. In this new architecture, we want to avoid such a process. This requirement is extremely important to create a scalable architecture. The addition of new models should not require changes to any of the other existing models.

9. Decentralized Management and Distributed Control

Decentralized management and distributed control means that each model is maintained at the "local" level, not at a central level. The current monolithic approach requires centralized model curation, such as many of the existing model repositories. We want to support local management of models.

Moreover, any creator of a model should be able to integrate their model from their local location to an ensemble of distributed models. This means that if one owner of a model has Model A and wishes to quickly test or integrate their model with a set of three other models: Model B, C and D. They should not have to download each of the other models to their local computer. With ease, the architecture should enable the owner of Model A to integrate with the other three models with little to no effort.

10. Hierarchical Support

Biological systems are systems of systems. This means that the architecture should support the ability for systems to be composed of other systems, while ensuring that each system satisfies the requirements herein.

The above ten requirements will be used as the basis for defining the architectural design.

4.2 Architecture Functional Specification

The goal of this section is to specify the functional requirements that will be used to design the architecture. We begin our functional specification process by abstracting the cell to be interconnection of biological pathways, as shown in Figure 4-1.

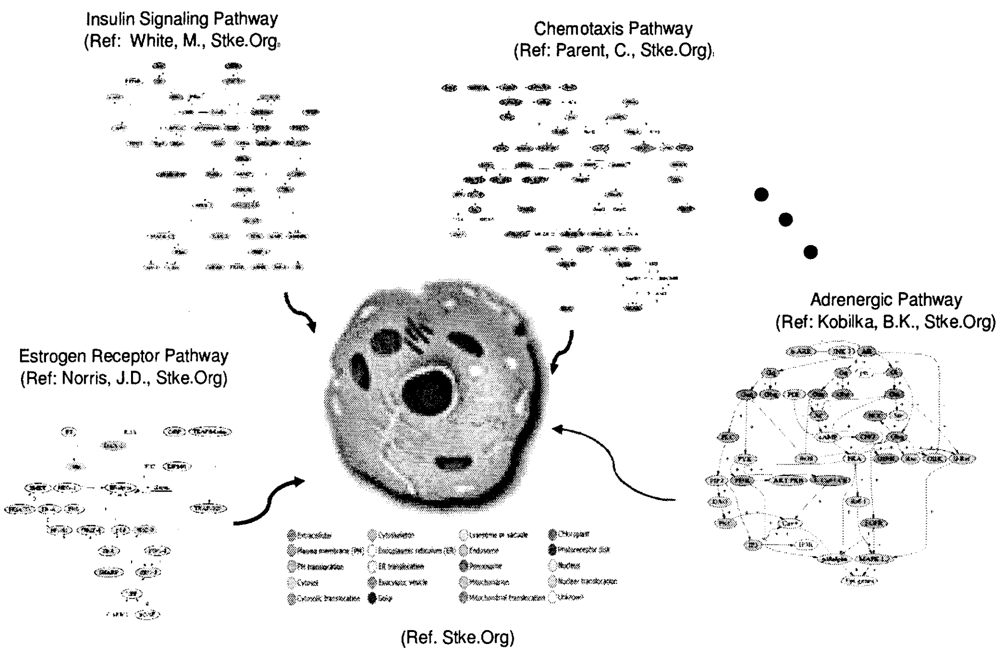

Insulin Signaling Pathway
(Ref: White, M., Stke.Org)
Chemotaxis Pathway
(Ref: Parent, C., Stke.Org)
Adrenergic Pathway
(Ref: Kobilka, B.K., Stke.Org)
Estrogen Receptor Pathway
(Ref: Norris, J.D., Stke.Org)
(Ref. Stke.Org)

Figure 4-1 - A cell as an interconnection of biological pathways.

Each biological pathway, within the cell, will need to communicate its state concentrations to other biological pathways, as shown in Figure 4-2, in the ensemble in order to simulate the function of the whole cell.

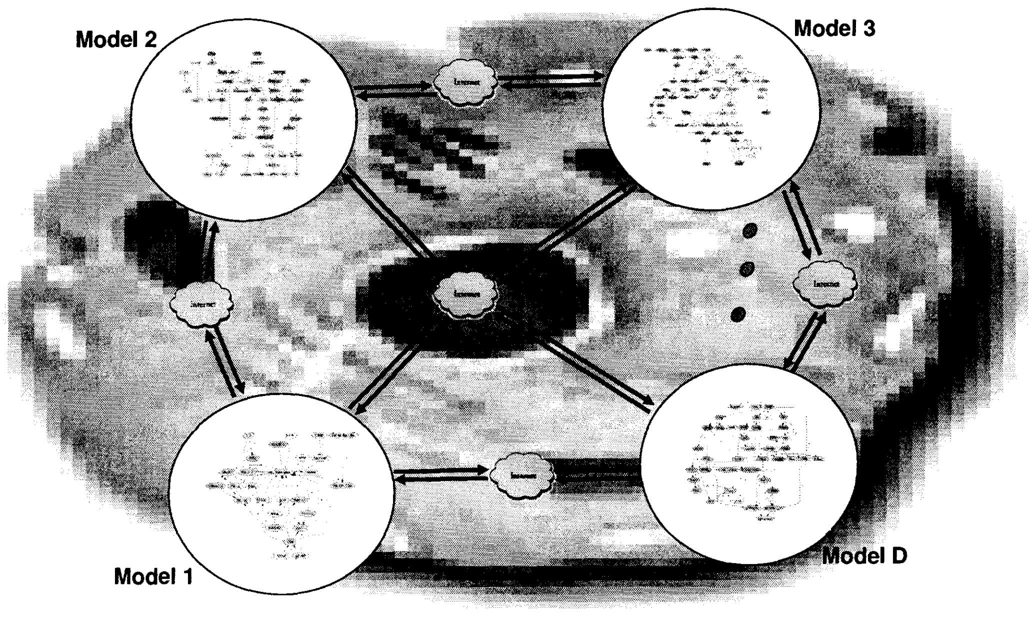


Figure 4-2: Communication among an ensemble of biological pathways from 1 to D pathways.

Formally, we present any one model in the ensemble as shown in Figure 4-3.

Let M denote the structure of a single biological pathway model in the architecture:

$M = <X_m, Y_m, Q, \delta_{int}, \lambda, ta>$

where,

X_m is a set of input values of species concentrations

Y_m is a set of output values of species concentrations

Q is a set of states

δ_{int}: $Q \rightarrow Q$ is the internal transition function

λ: $Q \rightarrow Y$ is the external transition function

ta: $Q \rightarrow \Re^{+}_{0,\inf inity}$ is the time advance function

Figure 4-3: Formal representation of a biological pathway model in the ensemble.

Also, the entire ensemble representing the whole cell model is formally represented as shown in Figure 4-4.

Let O denote the structure of a coupled or integrated model in the architecture:

O = <X, Y, {M_i}>

where,

X is a set of input values of species concentrations

Y is a set of output values of species concentrations

For each i,

M_i is a individual model

Figure 4-4: Formal representation of the total ensemble of interconnected biological pathway models.

Mathematically, if we allow Xi to denote the molecular species in the integrated model within the architecture where:

$$X_i, i = 1 \ldots N$$

With,

$$x_i, i = 1 \ldots N$$

representing the number of molecules of species x_i, then the state of the cell or any compartment within the cell (e.g. nucleus) is determined by:

$$\mathbf{x} = (x_1, x_2, \ldots x_N)^T$$

A reaction in this formalism is then represented as:

$$\mathbf{X}_r \xrightarrow{w_r(x_r)} \mathbf{X}$$

Where,

$\mathbf{X}_r$ represents the state of the system before the reaction.

and,

$w_r(x_r)$ represents the reaction *propensity* (the probability of reaction per unit time.

Based on the above formulation, this leads the Chemical Master Equation (Andersen, 1983):

$$\frac{dp(\mathbf{x},t)}{dt} = \sum_{r=1}^{R} w_r(\mathbf{x}_r)p(\mathbf{x}_r,t) - \sum_{r=1}^{R} w_r(\mathbf{x})p(\mathbf{x},t).$$

The above formalism provides us with the basis for providing a set of functional requirements.

- First, the cell or compartment is well-mixed. This means that a sufficiently long-time between reaction collisions takes place to ensure that each pair of molecules is equally likely to be the next to collide. This also means that the concentration of each species is high and transport essentially instantaneous.

- Second, the progress of the system only depends on previous state (e.g. Markov process).
- Third, between cells and compartments, transport is slower and associated with an observable rate.
- Fourth, we treat each pathway model as a *black box*. This means the following:
    - Inputs and outputs are species concentrations
    - Changes in localization are represented by *compartments*
    - Species are defined by their compartments
    - Species can move through compartments
    - Species can inhabit one or more compartments
- Fifth, the cell can be modeled as an integration of biological pathway models. Recall, biological pathways are moving from diagrammatic representations as shown in Figure 4-5 to biological pathway models, and each biological pathway model has internal parameters along with inputs and outputs which are the molecular species concentrations for the n^{th} and $(n+1)^{th}$ time step, respectively.

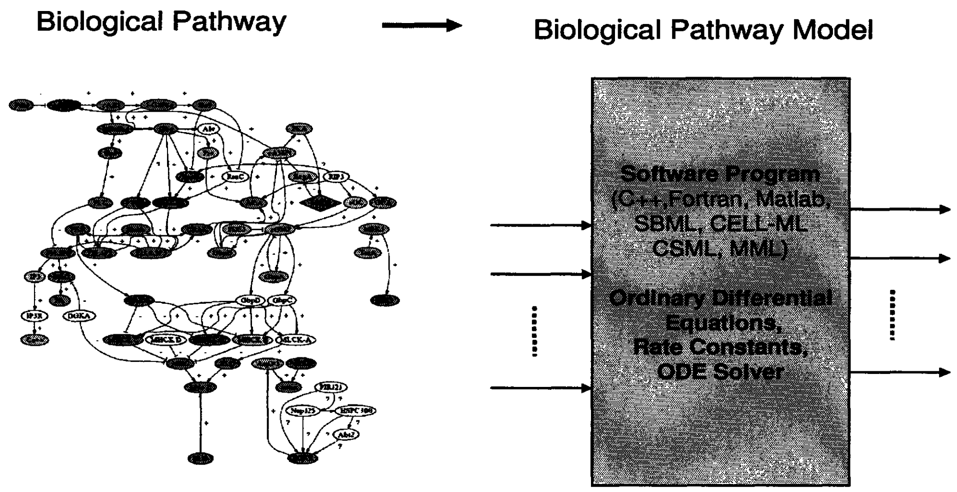

Biological Pathway
Biological Pathway Model
Software Program (C++,Fortran, Matlab, SBML, CELL-ML CSML, MML)
Ordinary Differential Equations, Rate Constants, ODE Solver

Figure 4-5: Transition of biological pathways to biological pathway models.

Thus, modeling the cell therefore can be seen as an interconnection of biological of pathway models as shown in Figure 4-6.

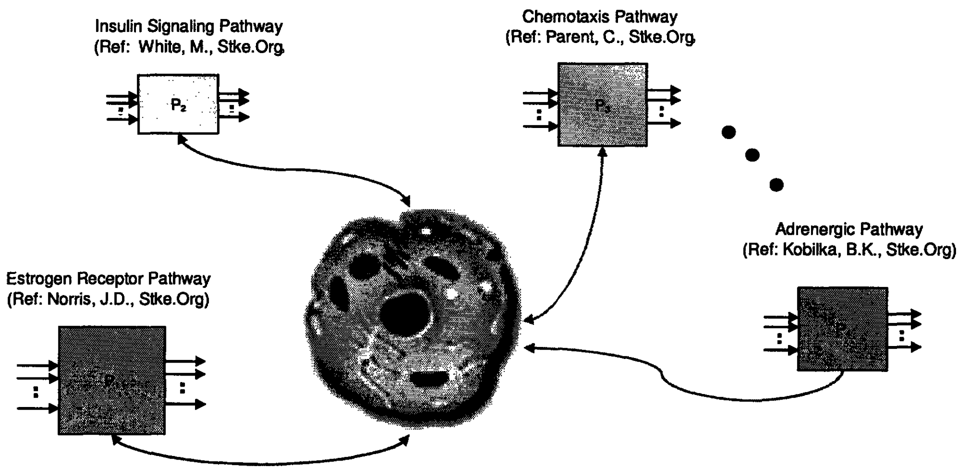

Insulin Signaling Pathway
(Ref: White, M., Stke.Org
Chemotaxis Pathway
(Ref: Parent, C., Stke.Org
Adrenergic Pathway
(Ref: Kobilka, B.K., Stke.Org)
Estrogen Receptor Pathway
(Ref: Norris, J.D., Stke.Org)

Figure 4-6: Modeling the cell as an interconnection of biological pathway models.

Recalling Figure 4-2, the cell can be represented as an ensemble of multiple biological pathway communicating inputs and outputs as shown in Figure 4-7.

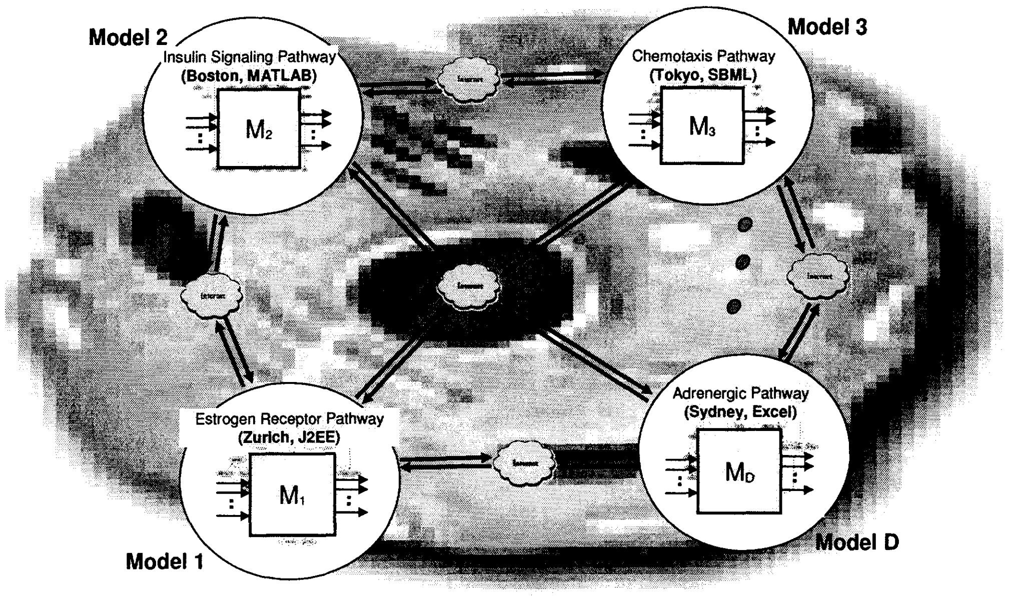


Figure 4-7: Modeling the cell as an ensemble of biological pathway models sharing and communicating input and output species concentrations.

4.3 Architecture Design

Based on the earlier discussions, it is clear that the monolithic approach, as shown in Figure 4-8, is widely used primarily because there are no other real alternatives in systems biology today. The main problems with this approach have already been discussed. The messaging approach, which has not been fully developed for systems biology, except in the two cases of Cellware and Cellulata, is in the right direction. However, both of these methods use a static messaging approach. There are no known solutions, to our knowledge, which use a dynamic messaging approach that supports scalability as well as ease of maintenance. Our first decision is to pursue a dynamic messaging approach

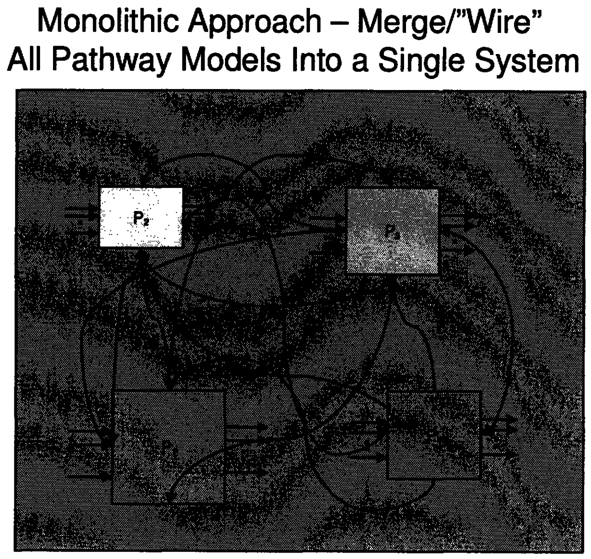

Monolithic Approach – Merge/"Wire"
All Pathway Models Into a Single System

Figure 4-8: Monolithic approach involves "wiring" all models together.

as the basis of the architecture. This dynamic messaging approach provides nearly all of the advantages to address the weaknesses of the monolithic approach and the static messaging approach, for the reasons previously stated.

Cytosolve: A Dynamic Messaging Approach

Based on the above discussion, we now introduce Cytosolve, a scalable architecture for integrating an ensemble of distributed biological pathway models. In Figure 4-9 the layout of the architecture for supporting a dynamic messaging approach is illustrated. The layout of this architecture will meet all of the requirements of the specifications outlined in the previous sections.

The elements of this architecture are:

- Biological Pathway Models - These are the boxes with the input and output arrows along the outer edges of the diagram. Each biological pathway model is a computer software program.
- Internet - This is represented by the internet "clouds". This serves to show that the biological pathway models can reside anywhere geographically on the planet and can use the internet for communication through the controller. This does not mean that we *have* to use the Internet. All models can be centralized on one computer and Cytosolve will communicate locally to each model.

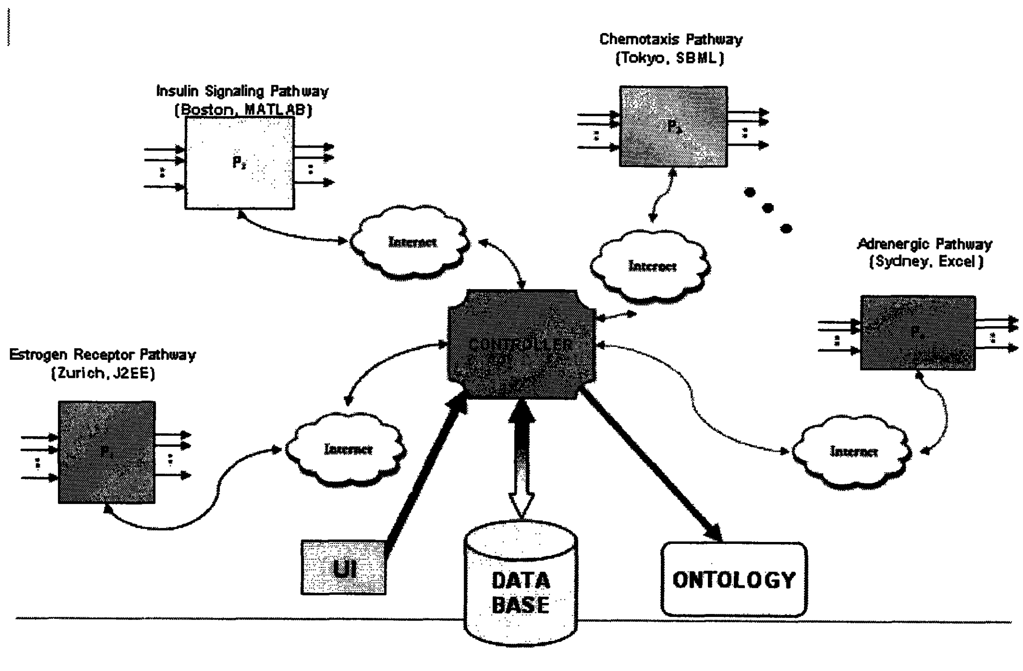


Figure 4-9: Cytosolve- A Dynamic messaging approach.

- Controller - This module serves to coordinate the computational activities across the various models. Note, that any one model itself could be another replication of this architecture as shown in Figure 4-10 to support systems of systems.

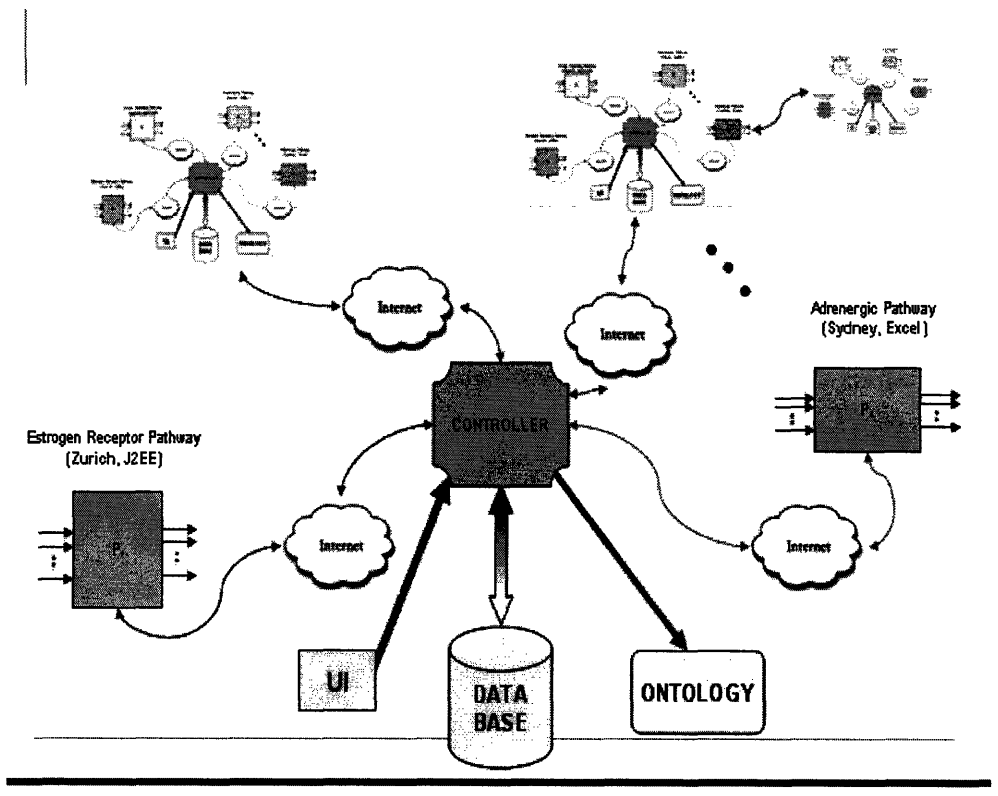

Internet
Internet
Adrenergic Pathway (Sydney, Excel)
Estrogen Receptor Pathway (Zurich, J2EE)
CONTROLLER
Internet
Internet
UI
DATA BASE
ONTOLOGY

Figure 4-10:- Dynamic messaging of systems within systems.

- Controller-to-Pathway Interface - These are the arrows in the diagram from the Controller to the Pathway, and represent the mechanism by which the controller communicates with each individual pathway model.

- User Interface – The user interface allows the user to specify which models will be included within the architecture.

- Ontology – This is the data which specifies characteristics of each model's input-output behavior to allow the Controller to effectively communicate across all models.

The key features of this architecture include an infrastructure that provides simple communications interface to each model, is distributed, Web-enabled, and automatically aggregates the models to build the integrated model. The architecture supports both distributed and parallel processing, while using a hybrid of shared memory and message passing. The shared memory is for tracking the species concentrations across all models in time. Message passing is used for remote communications between the controller and individual models. The architecture is to be built in an open environment supporting: 1. publicly available tools, and 2. emerging standards.

In the next sections of this chapter, major elements of the architecture design are discussed in detail. Prior to the end of this chapter, a *simple example* is provided so the reader can follow the exact mechanics of how the architecture integrates models to compute a solution.

4.4 Architecture Implementation

This section discusses the details of implementing the architecture. The architecture was implemented using open source tools to reduce expense and to ensure that future research could be pursued on the architecture with minimal reliance on proprietary tools.

Development Environment and Tools Selection

Various third-party open source software tools were selected to implement the initial prototype of the architecture. These are listed below.

ODE SOLVER LIBRARY

The SBML ODE Solver Library (SOSlib) is a programming library for symbolic and numerical analysis of chemical reaction network models encoded in the Systems Biology Markup Language (SBML) (Machne, et al., 2006). This Library takes as input an SBML model file and then computes the steady-state solution of species concentrations for a given number of time steps.

SOAP/WSDL

SOAP 1.2 & WSDL 1.1 - SOAP (Simple Object Access Protocol) (http://www.w3.org/TR/soap) based Web Services technology(http://www.w3.org/ws) has gained much attention as an open standard enabling interoperability among applications across heterogeneous architectures and different networks. The European Bioinformatics Institute (EBI) is using this technology to provide robust data retrieval and data analysis mechanisms to the scientific community and to enhance utilization of the biological resources it already provides (N. Harte, V. Silventoinen, E. Quevillon, S. Robinson, K. Kallio, X. Fustero, P. Patel, P. Jokinen and R. Lopez (2004) Nucleic Acids Res., 32, 3-9]. These services are available free to all users from http://www.ebi.ac.uk/ Tools/web services.

BIOMODELS

The BioModels Database, located at http://www.ebi.ac.uk/biomodels/, is part of the international initiative BioModels.net, provides access to published, peer-reviewed, quantitative models of biochemical and cellular systems, stored in the SBML and CellML formats. Each model is carefully curated to verify that it corresponds to the reference publication and gives the proper numerical results (Le Novere, et al., 2006).

C PROGRAMMING LANGUAGE

The C programming language was used to build the Controller-Pathway Interfaces. Visual C++ 8.0 is the version used.

WEB SERVER

The web server Apache Tomcat Apache 5.5.23 / ANT 1.7.0 is used

JAVA

The Controller is programmed using J2EE 1.4.

For the initial implementation, the Controller was developed on a DELL Windows Machine running Microsoft XP with Service Pak 2.2 with 1 GB of RAM on a Pentium Processor.

Communications Protocol

The Cytosolve web service is designed for remote model simulations through the use of SOAP/WSDL. During a run, the web service can be instructed by a remote computer to execute a local model and send back results. The remote computer can instruct the web service to perform two major operations. First, the web service can be instructed to simulate a local model over a single time step. After simulation, the service sends back new concentration values calculated by the model. In the second operation, the web service can be instructed to insert new species concentration values into the model simulation. This allows external control of the simulation.

Using these two operations, a centralized controller can theoretically couple multiple models together, with each model running on different computers. During operation, the WSDL web service will be continuously running, waiting for remote simulation commands. When a command is received, the WSDL web service passes the command to intermediate Java layer called Cytosolve_Java. Cytosolve_Java simply calls Cytosolve_C, which executes the simulation commands and returns the result to Cytosolve_Java. Cytosolve_Java in turn returns the results to the WSDL web service, which sends it back to the remote computer. Figure 4-11 gives a code example of the initialization using the Cytosolve_C code.

A layer called JNI is used to enable messaging of Java module Cytosolve_Java with the simulation module Cytosolve_C. Even though this step makes compiling the source code significantly more complicated, it is essential to establish the messaging.

Since web services are stateless, information is not retained during separate calls to a single web service. This is a problem since simulation results must somehow be retained between individual web service calls. Otherwise, it would be impossible to simulate an SBML model beyond a single time step since results for that single time step would then be immediately lost.

```
/* initializes the solver */
static int IntegratorInstance_initializeSolver(integratorInstance_t *engine,
                    Data_t *data,
                    Settings_t *opt, odeModel_t *om)
{
 int i;
 Solver_t *solver = engine->solver;
 Results_t *results = data->results;;

 /* irreversibly linking the engine to its input model */
 engine->om = om;

 /* joining option, data and result structures */
 engine->opt = opt;
 engine->data = data;
 engine->results = data->results;

 /* initialize the solver's time settings */

 /* set initial time, first output time and number of time steps */
 solver->t0 = opt->TimePoints[0];     /* initial time         */

 /* first output time as passed  */
 if ( opt->Indefinitely )
  solver->tout = opt->Time;
 else
  solver->tout = opt->TimePoints[1];

 solver->nout = opt->PrintStep;    /* number of output steps */
 solver->t = opt->TimePoints[0];  /*  current time, always 0,
                    when starting from odeModel */

 /* set up loop variables */
 solver->iout=1;        /* counts integration steps, start with 1 */

 /* write initial conditions to results structure */
 if ( opt->StoreResults ) {
  results->time[0] = data->currenttime;
  for ( i=0; i<data->nvalues; i++ )
   results->value[i][0] = data->value[i];
 }

 /* count integration runs with this integratorInstance */
 data->run++;

 /* initialize specific solver structures */
 return IntegratorInstance_initializeSolverStructures(engine);

}
```

Figure 4-11: Cytosolve_C code sample.

A number of approaches were explored to address this problem. One approach was to write out all of the simulation results to a file during one WSDL invocation, then read the file during the next. However, it was discovered that with this approach, accuracy was lost over time. The reason is that the Solver has a large number of internal variables that must be saved between individual time steps. Failure to save these internal variables causes slow degradation in the accuracy of the results. Saving each of these numerous internal variables is a tedious task as there are hundreds of internal variables in the Solver.

This approach is hard to implement and hence dropped because (a) identifying those internal variables that need to be saved is not practical; and (b) large amounts of code would need to be rewritten. Another approach considered was to send to the remote computer all information that is needed to reconstruct itself. However, this approach was proved to be inefficient and tedious as it would involve sending and receiving enormous amount of information. Hence this approach was also dropped.

The third approach was to use Stateful web services. Stateful web services allow variables to be saved between different WSDL invocations. Even though stateful web services appeared to be an appropriate solution to the problem, it was discovered that in order to use stateful web services, all variables that are required to be maintained between WSDL invocations must be declared as resources. This approach was also dropped because (a) declaring variables is tedious; and (b) programming environment in C is different from that of WSDL, which is Java-based which made it impossible to therefore declare C variables to keep between invocations.

The final approach used to solve this problem was to open a continuously running Java thread that waits for commands from the WSDL web service. With this approach, the internal variables are not lost between WSDL invocations because the Java thread is continuously running. Messaging between the WSDL web service and continuously running Java thread is performed using files. This approach successfully allowed the stateless WSDL web service to communicate with stateful data.

This continuously running thread was named SolverController, and is made part of the Cytosolve_Java package. Cytosolve_Java therefore performs two roles. It acts as a bridge between Java and C environments, and in addition acts as a bridge between the stateless WSDL web services to stateful data. We consider this implementation of using Java and C for supporting a "stateful" web service a significant accomplishment which allows us to use open standards.

Logical Software Architecture

The logical software architecture is developed in a three-tier layered approach as shown in Figure 4-12.

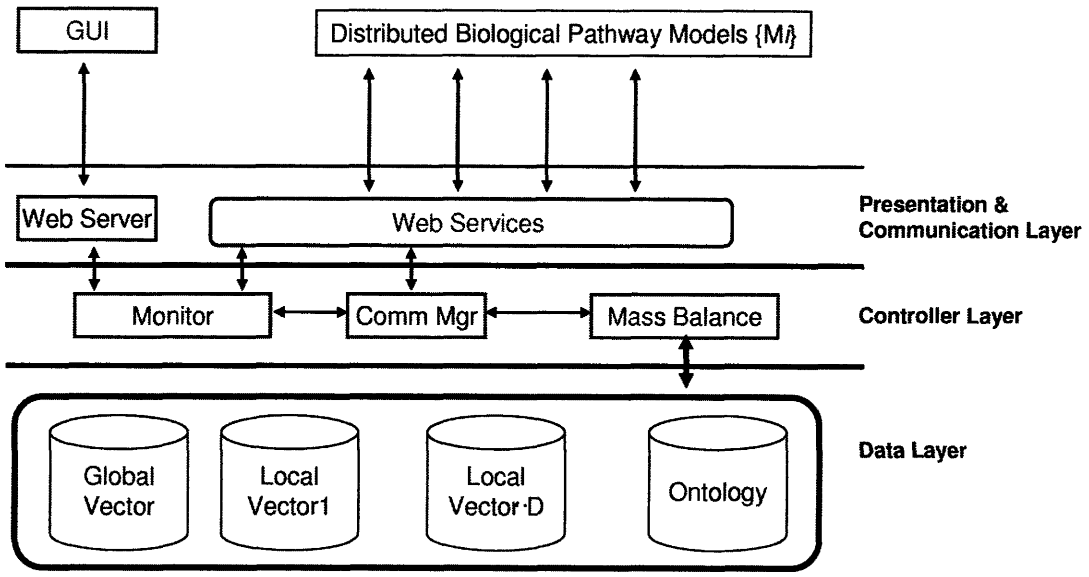


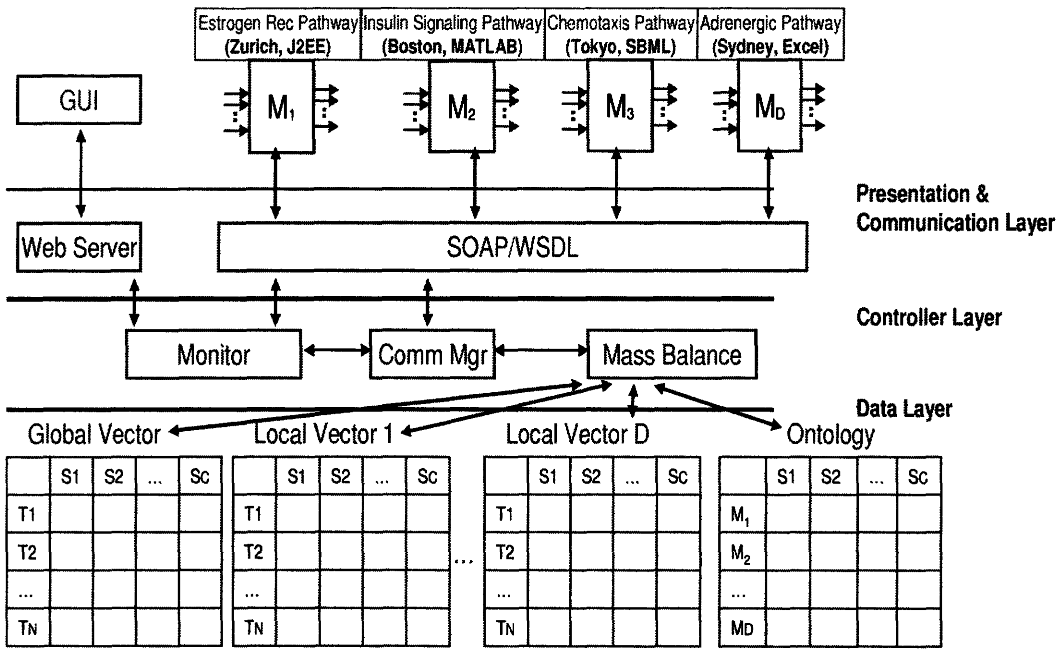
Figure 4-12:- Logical software architecture.

At the Data Layer resides the memory-resident data to track species concentrations in time within the integrated model (Global Vector) and across each individual biological pathway model (Local Vector 1 to D). The Data Layer also contains the Ontology that is used to manage nomenclature and species identification across all individual biological pathway models.

The Controller Layer consists of three main components, the Monitor, Communications Manager (Comm Mgr) and the Mass Balance algorithm. These three components work in concert to orchestrate the calculation and integration of all D biological pathway models.

The Presentation and Communications Layer provides the communications support, aforementioned, to support communication among the models as well as with the User Interface.

A detailed implementation of the Logical Software Architecture is provide in Figure 4-13. At the Data Layer, the Global Vector data structure is depicted in this diagram consisting of rows for each time step from 1 to N.



Estrogen Rec Pathway (Zurich, J2EE)
Insulin Signaling Pathway (Boston, MATLAB)
Chemotaxis Pathway (Tokyo, SBML)
Adrenergic Pathway (Sydney, Excel)
GUI
M1
M2
M3
MD
Presentation & Communication Layer
Web Server
SOAP/WSDL
Controller Layer
Monitor
Comm Mgr
Mass Balance
Data Layer
Global Vector
Local Vector 1
Local Vector D
Ontology
S1 S2 ... Sc
T1 T2 ... TN
M1 M2 ... MD

Figure 4-13:- Detailed Logical Software Architecture.

The columns are for all species from 1 to C across the entire ensemble of integrated models. The Local Vector data structure is also shown. The Ontology consists of individual entries for each model from 1 to D models, specifying the species components, which is discussed in detail below.

At the Controller Layer, only the Mass Balance component can communicate with the Data Layer. The Comm Mgr orchestrates communication between the Monitor and the Mass Balance components. The Monitor serves to modulate and monitor calculations across the ensemble of [Mi] models. Both the Comm Manager and the Monitor can communicate through to the Presentation and Communication Layer. The benefits of his logical architecture are to specialize different components for different tasks thus optimizing performance.

4.5 Pathway Model Representation

Each biological pathway model is represented or encoded into a computer software program which includes code and data. While there are many formats, these formats are the most common used to encode biological pathway models:

SBML - Systems Biology Markup Language. This is a file XML format for storing the differential equations and the data used solve a biological pathway model. SBML does not actually *solve* the model, that solving is done by another program called the *solver* which takes as input the SBML files.

CellML - Cell Markup Language. This, like SBML, is an alternative XML format for storing differential equations and describing cellular systems.

MATLAB - is software architecture for solving mathematical problems. The equations of biological pathway models can be encoded in MATLAB and solved using its own solver.

FORTRAN - is a language developed at IBM, as the name implies for FORmula TRANslation. Many biological models are encoded and can be solved in this language.

C/C++ - Like Fortran, this language is also used to encode biological models and includes external libraries which an be invoked to perform the solving

Models encoded in the above languages typically have internal parameters , while the inputs and outputs of these models are the species concentrations at a particular time n, and a particular time n+1, respectively. The input species concentration is denoted as

$$S_n^{j,i}$$

where, for any one model Mi, i is from 1 to D models, at time step n, **S** is a vector of species concentrations for species j, where j is from 1 to C species.

Similarly, the output species concentration, at time step n+1, is denoted as

$$S_{n+1}^{j,i}$$

This architecture is agnostic the format of the model. The only requirement is that the model can be configured through SOAP/WSDL protocols to receive input species concentrations for time step n and send output species concentrations for time step n+1. Beyond, this requirement, the architecture treats any *pathway model as a black box.*

4.6 Pathway Registration and Ontology

The architecture herein treats each model as an independent part of a collective or ensemble of models which are computed together (through a computational mechanism discussed later). How does a new pathway become part of this collective?

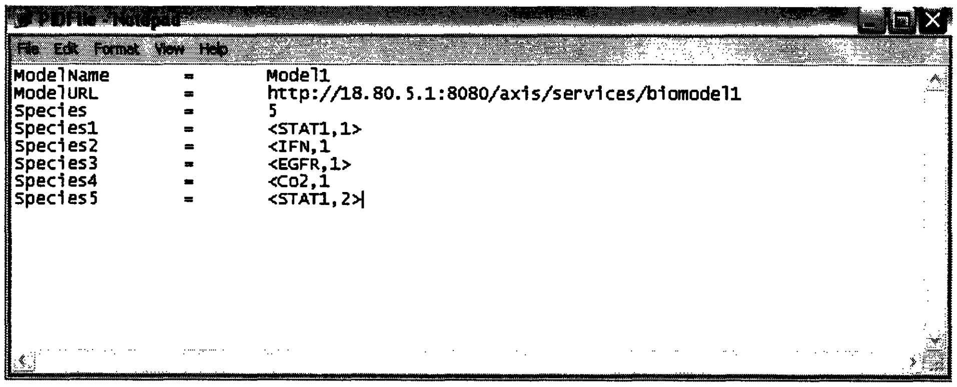
This process is enabled through *pathway registration.* The pathway registration process serves to create elements with the architecture's *Ontology* to describe the pathway's inputs and outputs to the Controller, through a *Pathway Interface*

Document (PID). In the case of biological pathway models, we are concerned primarily with coupling and *mass-balancing* (see Controller section) of a system of biological pathway models, thus our PID format needs to provide support for such integration. The format of the PID is show in Table 4-1. In Figure 4-14 is a picture of an example PID file for a model called Model 1.

Name of Variable	Meaning
ModelName	The unique name of the model
ModelURL	The location on the Internet where the model executable code resides
Species	The number of species in the model
Species1,Loc	The name of the first species (as used in the Model) and its location ID
Species 2,Loc	The name of the second species (as used in the Model) and its location ID.
...	...
Species n,Loc	The name of the nth species (as used in the Model) and its location ID

Table 4-1 - PID Format.

The Loc is the location ID denoting which compartments the species appears in the cell. For example in Figure 4-14, the species STAT1 can appear in two locations. Loc ID "1" may denote the nucleus and Loc ID "2" may denote the mitochondrion, etc. The key point is that species need to be distinguished by their location. The MIRIAM standard was published which serves to provide a framework for model developers to provide a minimal set of information for defining biochemical models (Le Novere, et al., 2007) . Our Ontology can link to take advantage of this emerging standard.



```
ModelName      =      Model1
ModelURL       =      http://18.80.5.1:8080/axis/services/biomodel1
Species        =      5
Species1       =      <STAT1,1>
Species2       =      <IFN,1
Species3       =      <EGFR,1>
Species4       =      <Co2,1
Species5       =      <STAT1,2>
```

Figure 4-14 – Representation of simple biological pathway model.

Each model when it registers itself to be part of the collective of models creates one of the above files and ensures the PID is placed into the Controller's Ontology, which organizes all the PID files for a particular Controller. There are some interesting challenges that can take place during registration of two different models. One example is if two species names are assigned the same name but mean something different or two species names are assigned different names but mean the same thing. (Schulz, et al., 2006) have identified this as a problem in the implementation of their tool SBMLMerge to support the integration of SBML model files, albeit in a monolithic manner. In our case, the solution is managed by the Ontology without requiring recoding of the individual biological pathway models. The Ontology provides a list of Unique Identifiers for mapping species names, and this list can grow as needed. In many ways it is like a *thesaurus*. An example Ontology is shown in Table 4-2.

Unique Identifier	Synonym
11231	Ca
11231	Calcium
11231	Ca+
11245	SOCS
...	...

Table 4-2 - Ontology Example.

Let us consider the case of two models where the species name "EGFR" is assigned the same name in each model. When any model registers itself with the system, the PID file for that model is compared with other existing PID files and the Ontology. If EGFR name is used in another model, than the system, through the User Interface (UI) displays a message to the user indicating that EGFR is currently being used by these other models, along with the Unique Identifier. If the developer of that model believes that EGFR in fact refers to the same species, then no changes are required. If the developer believes that the species name is different, than the onus is on the developer to create a new name and resubmit.

Let us consider the other case, where two models have two different names for the exact same species. For example, suppose one model refers to a species called "Calcium" and another model refers to a species called "Ca+". In the Ontology in Table 4-2, during registration, both of these species will point to

Unique ID 11231. The system will automatically resolve those species to be the same species internally during model integration. Thus, again the developers of the model do not have to worry about renaming their species to be the same species names.

The Ontology is public, thus, prior to registering a new model, the developer can decide to use existing names or add their own new name to the Ontology. For example, let us say a developer has a species called "Cal" and that also refers to the same species as "Calcium" or "Ca+", then the developer can update the Ontology so 11231 also has an entry for "Cal" or they can adjust their species name to one of the existing synonyms.

4.6 Controller-Pathway Interface Description

For all registered pathways the controller needs to be able to interact with each pathway to solve the integrated problem. The registration of the PID to the controller's Ontology assumes that interface protocols to the model are provided in the format the controller seeks. This section describes the necessary interface between the controller and the pathway models. There are only three interface protocols that are needed for any pathway model.

Initializing Access to the Pathway Model

The function *createPathway()* as shown below is used by the controller to setup the initial connection with the pathway model

```
public pathwayProcess uidPathway  createPathway {
            String URL          //location of the pathway model
            double totalTime,   //total time in sec for execution
            int numSteps);         //number of time steps
            }
```

The above function serves to return a unique identifier to the controller for the pathway model during initialization. For example, the call:

```
            uidPathway1 = createPathway(Model1URL, 10, 50);
```

will return a value of say 11214122 for uidPathway1. This handle will be used to refer to the pathway model in future interface calls. The first parameter in this call is the URL at which the executable code of the model resides. The next parameter denotes the time in seconds (in this case 10 seconds) how long the model should run. The final parameter denotes the number of time steps (in this case 50 time steps) for the model to run.

Processing a Time Step on Pathway Model

The function pathway *stepPathway ()* as shown below is used by the controller to request the pathway model to return the species results for one time step.

double[] results = new double[numSpecies];

results = stepPathway(uidPathway1, speciesList, newValues)

The above function call is used by the Controller to contact the pathway at uid for *uidPathway1*, with the list of species, speciesList that pathway manages, along with the input list of species to be used for the input time step *n*, which are in *newValues*. The *stepPathway()* returns a list of *results* which are the species values for that next time *n+1* given the input values of newValues.

Closing Access to a Pathway Model

The interface call *closePathway()* is used to close the session by the Controller with the pathway model.

errorCode = closePathway(uidPathway1);

This function for example closes the pathway with uid *uidPathway1.*

4.7 Controller

The Controller serves two purposes. First, it mediates communication across all pathway models. Second, it provides *computational steering* by ensuring mass conservation across all integrated models for each time step. Referring to Figure 4-13 will assist in understanding the Controller's processes described in this section. The first sub-section describes how the Controller manages communications across all pathway models. The next sub-section discusses how the Controller performs mass balance and *steers* the solution by managing this constraint across all models.

Controller Communication

The controller communication proceeds along various key steps which are outlined in detail below.

Step 1- Controller Initialization

```
//Create the controller at the start of the main program

public static void main(String [] args) throws Exception {
        Controller controller = new Controller();
        controller.process();
   }
```

Figure 4-15 - Controller startup.

When the system first starts the Controller is initiated from the main program as shown in Figure 4-15. Here, the controller creates a new object for itself as shown in Figure 4-16.

Step 2 - Memory Allocation

The controller() object is shown in Figure 4-16. Many startup activities are initiated upon creation of the controller () object. Critical ones are the memory allocation for the storage of the species concentrations for each pathway over time, as well as the global vector of species concentrations for the integrated model over time.

```
// Controller object
public Controller() {
        // Ontology is loaded along with the PID's
        loadProperties();

        // Vector of matrices, a matrix for each pathway
        localVect = new Vector<Vector>();
        for (int x=0; x<numPathways; x++)
            localVect.add(new Vector<Object>()) ;

        // Global vector for integrated model species concentrations
        globalVect = new Vector<Object>() ;

        // Create the Monitor
        monitor = new Monitor(numPathways);
    }
```
Figure 4-16 - Controller object.

Step 3 - Initialize Monitor

The Monitor is a sub-system of the Controller. It serves to track the progress of

```
public Monitor (int pc){
    processCount = pc;      //number of independent pathways
    newRow = 0;             //starting row in Global Pathway Matrix
    }
```

Figure 4-17 - Monitor initialization

each pathway's solution time. In Figure 4-16, the last event of the Controller() object is to initiate the Monitor, which initializes the number of pathways and the state of processing across all pathways as shown in Figure 4-17.

Step 4 - Comm Mgr Messages All Pathways to Wake Up

The Comm Mgr messages all of the pathway models to start up using the *createPathway()* Controller-Pathway interface call. There is quite a bit of complex

```
/**
  * For the given interface, get the stub implementation.
  * If this service has no port for the given interface,
  * then ServiceException is thrown.
  */
   public java.rmi.Remote getPort(javax.xml.namespace.QName portName, Class
   serviceEndpointInterface) throws javax.xml.rpc.ServiceException {
     if (portName == null) {
        return getPort(serviceEndpointInterface);
     }
     java.lang.String inputPortName = portName.getLocalPart();
     if ("pathwaySolver".equals(inputPortName)) {
        return getsbmlSolver();
     }
     else {
        java.rmi.Remote _stub = getPort(serviceEndpointInterface);
        ((org.apache.axis.client.Stub) _stub).setPortName(portName);
        return _stub;
     }
 }
```

Figure 4-18 - Port error processing code sample.

messaging and communication that takes place to accomplish this using the Web Services Description Language Protocol (WSDL) as shown in Figure 4-18. This code sample shows just one small piece of code for handling any port initialization errors that occurs. Many such errors are possible and need to be handled appropriately. Appendix A provides more code samples of the details of message error processing. At the end of this step all pathway models are awaiting to begin processing a time step upon initiation.

Step 4 - Initiate Monitor

The Monitor is initiated. All models are awoken and started simultaneously. Monitor continues to run observing state of calculation across all models.

Step 5 - All Models Execute a Time Step in Parallel

```
try {
        results = solverProcess.step(uid, speciesToChange, newValues, wantedSpecies);
        if(results == null)
                {
                 System.exit(1);
                }
    }
    catch (Exception e) {
        System.out.println("Module    : failed" + e);
        }
    // add data to local memory vector
    double[] currentSum = new double[resultSize];
    for (int x = 0; x < resultSize; x++){
        currentSum[x] = results[x];
    }
    //update values in local vector for time step n+1
    localVect.add(currentSum);
    for (int i = 0 ; i < currentSum.length ; i++)
        {
            System.out.print(currentSum[i] + " ");
        }
        System.out.println();
    //tells monitor it is done
    monitor.complete();
```

Figure 4-19 - A model executes, notifies monitor and rests.

Since each model is its own process, running on its own machine, each model process its input species concentrations and updates it local species vector and sends signal back to the Monitor that it has completed. After each model is completed, it goes to sleep in order to optimize CPU usage. This is shown in Figure 4-19.

Step 6 - Monitor Observes Completion of Calculation of Time Step

The Monitor observes the completion of a Time Step across all models. Once all models have completed their processing for a Time Step, the Monitor passes control to the Comm Mgr.

Step 7 - Comm Mgr Executes Mass Balance for Time Step

For a time step, the Comm Mgr now contacts the Mass Balance module sets the new values for all species across *all* models based on the mass balance calculation (which is detailed in the next section). If this is the last time step, then the Comm Mgr exits in Step 8, otherwise the Comm Mgr continues to Step 4.

Step 8 - Controller Stops

The Comm Mgr stops by sending *closePathway()* calls to each pathway model and performs a variety of cleanup functions to release resources, memory, etc. Figure 4-20 shows a small code sample of such cleanup processes. These processes include the ability to manage exceptions as well as the proper unlocking of Web Services connections to ensure proper resource and memory management.

```
public void cleanup(java.lang.String in0) throws java.rmi.RemoteException {
      if (super.cachedEndpoint == null) {
         throw new org.apache.axis.NoEndPointException();
      }
      org.apache.axis.client.Call _call = createCall();
      _call.setOperation(_operations[2]);
      _call.setUseSOAPAction(true);
      _call.setSOAPActionURI("");
      _call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANTS);
      _call.setOperationName(new javax.xml.namespace.QName("urn:pathwaySolver",
"cleanup"));

      setRequestHeaders(_call);
      setAttachments(_call);
 try {       java.lang.Object _resp = _call.invoke(new java.lang.Object[] {in0});

      if (_resp instanceof java.rmi.RemoteException) {
         throw (java.rmi.RemoteException)_resp;
      }
      extractAttachments(_call);
  } catch (org.apache.axis.AxisFault axisFaultException) {
  throw axisFaultException;
}
```

Figure 4-20 - Clean up of pathway model resources.

Controller Computational Steering of Automated Mass Balance

Computational steering (or interactive program steering, application steering, interactive steering) enables us to observe and interact with a simulation during its execution, steering it as necessary. This is formally what is taking place in Step 7 above. In the next section we provide more detail on how the architecture's controller performs this automated mass balance by *steering* each model's input for the subsequent time step based on species values calculated through mass balance.

4.8 Mass Balance

The Mass Balance component of the architecture serves to provide the calculation of species concentration for each time step n across the ensemble of models. This section describes the mathematical formalism and the implementation

Mathematical Formalism

As previously stated, we treat each model as a black box with the input and output being a vector of species concentrations denoted by the following two variables:

$$S_{n}^{j,i}$$

which denotes the species concentration at time step n, of the ith model and the jth species, and,

$$S_{n+1}^{j,i}$$

which denotes the species concentration at time step n+1, of the ith model and the jth species, respectively. Using this notation, we define a new variable

$$S_{g,n}^{j,i}$$

which denotes the species concentration of the integrated model in the global vector (denoted by subscript 'g') contributed by the ith model and the jth species.

Using the above notations, mathematically the formalism for the mass balance is represented as follows:

$$\left(S_{g,n+1}^{j}\right)=\left(S_{g,n}^{j}\right)+\left(\sum_{i=1}^{B}(S_{n}^{j,i}-S_{n+1}^{j,i})\right)$$

Algorithm and Code

```
// initialize sum vector
double[] sum = new double[globalSpecies.length];
 for ( int z=0; z < globalSpecies.length; z++)
    sum[z] = ((double [])(globalVect.lastElement()))[z];

// for each solver, add result to sum vector using local->global species mapping
 for (int y=0; y < numSolvers; y++){
    double[] rs = (double[])((localVect.get(y)).lastElement());
    int[] map = (int [])mapping.get(y);
    double[] oldrs = ((double [])(globalVect.lastElement()));

    // mass balance calculation across all species
    for ( int z=0; z < rs.length; z++)
        sum[map[z]] = sum[map[z]] + (rs[z] - oldrs[map[z]]);
 }

 // add new row or the inputs for all pathways
 globalVect.add(sum);
```

Figure 4-21 - Core code for automated mass balance.

Based on the above mathematical formalism, for each time step, the Controller maintains a store of all species concentrations across all pathway models. After each time step, by each model, the Controller calculates for each species which models consumed and which models contributed to a species concentration. This summation is used to supplant the input for the species value across all models. The Controller, therefore, manipulates the next input and steers the calculation of the integrated model through this automated mass balance process. In Figure 4-21, a small code sample shows the core of this algorithm. Algorithmically, this code is summarized in Figure 4-22.

```
Algorithm

Initialize
        Contact all Mi with {So} values from GlobalVect
        Initialize LocalVects with {So}
        Where,
                {S0} denotes the initial conditions across {Mi}

For each time step, tn
        {
        LookUpGlobalVect (Row n, S)
        SendSpeciesValues (TimeStep n, {Mi} )
        Sleep (); until all LocalVects for current time step calculated
        MassBalanceCalculation(Si,j, {Mi}) for each compartment
        UpdateGlobalVect(Sj)
        }
```

Figure 4-22 - Algorithm for automated mass balance.

Based on the logical software architecture, the Mass Balance component interacts with the Data Layer and Comm Manager to execute the algorithm through intra-process communications as shown in Figure 4-23.

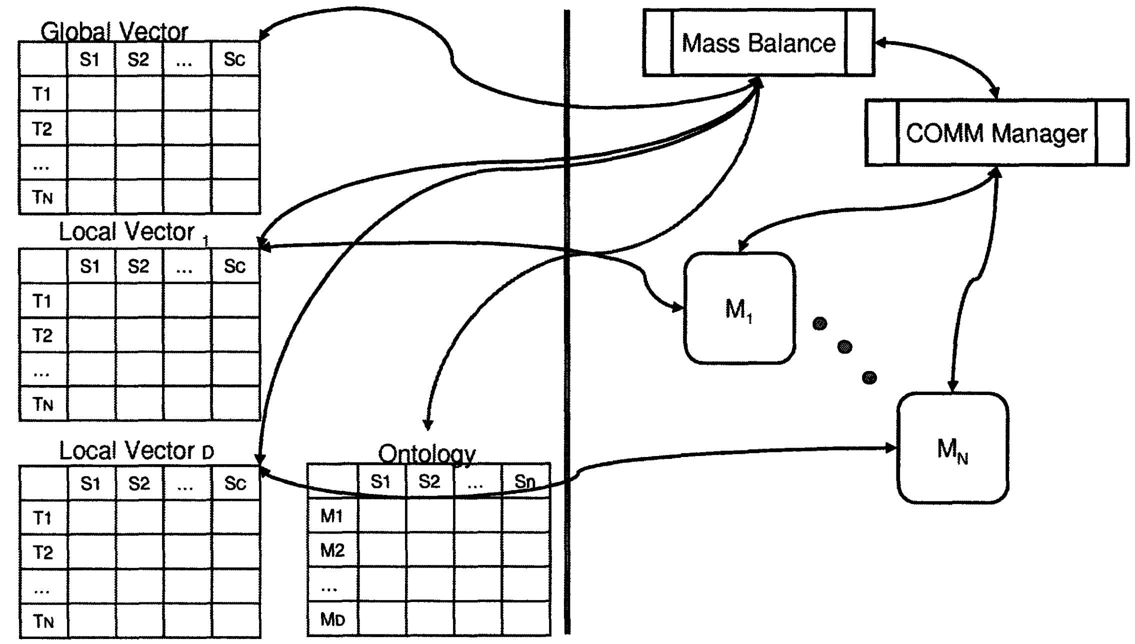


Figure 4-23 - Intra-process communications of mass balance component.

In Figure 4-23, the Comm Mgr serves to communicate and ensure that each model updates its Local Vector with the current species concentration outputs. The Mass Balance component, using the Ontology, executes its algorithm by evaluating the species concentration in the Global Vector, for the integrated model, by receiving the most up to date values for species concentrations from each model [Mi]'s Local Vectors.

4.9 Test Example

Once the architecture is implemented initial tests were run to validate the implementation as well as to understand the elements contributing to the computation time.

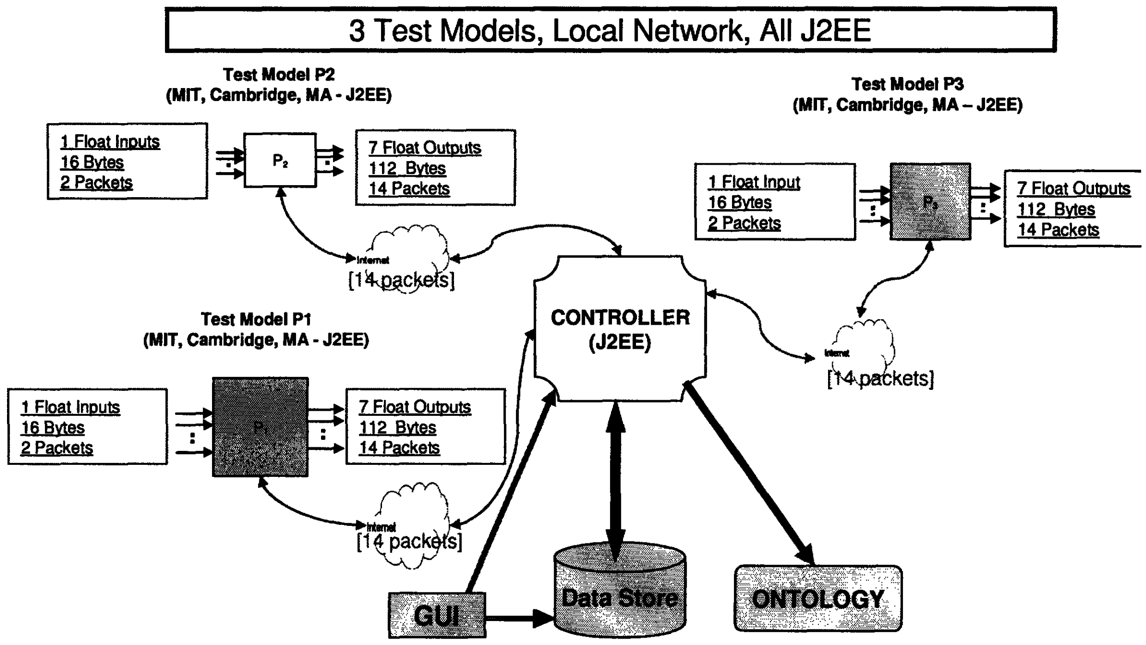


Figure 4-24 - Implementation test for Test Case 1.

Test Case 1

In the first test, three models are setup on three different computers within a local network as shown in Figure 4-24. Each model is a simple calculation which takes as its input on value and performs the same and simple mathematical calculation within each model to return seven values. Each model is in the same format. In this case, each model is written in J2EE. The results of performing this test are shown in Figure 4-25.

3 Test Models, Local Network, All J2EE

- Model Formats
  - All on J2EE, Local
- Transmission Time (ms)
  - ~ 6 ms
- Computation Time (ms)
  - Model: ~9 to 18 ms
  - Controller: ~3 ms
- Total Time: ~27 ms

*Averaging from 50 tests for each sample

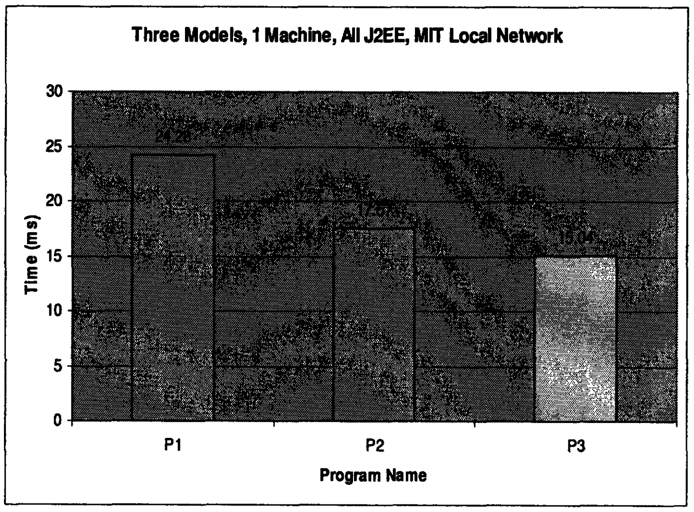


Figure 4-25 - Results of performing Test Case 1.

Test Case 2

In the second test, three models are setup on three different computers within a local network as shown in Figure 4-26. Each model is a simple calculation which takes as its input one value and performs the same and simple mathematical calculation within each model to return seven values. Two models are in the same format: J2EE and one model is in MATLAB.

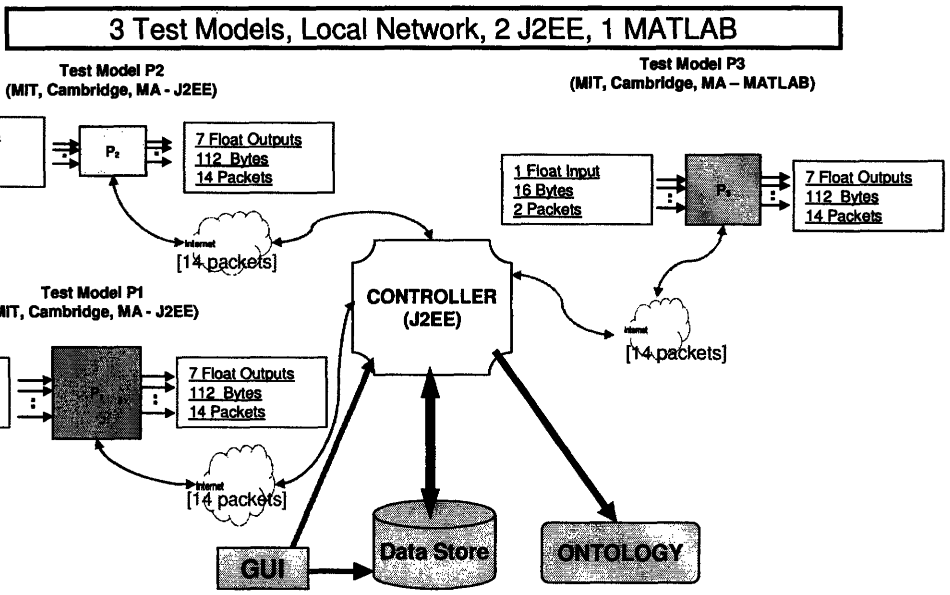


Figure 4-26 - Implementation test for Test Case 2.

The results from performing Test Case 2 are shown in Figure 4-27.

3 Test Models, Local Network, 2 J2EE, 1 MATLAB

- Model Formats
  - 2 on J2EE, Local
  - 1 on MATLAB, Local
- Transmission Time (ms)
  - ~ 6 ms
- Computation Time (ms)
  - J2EE: ~9 to 18 ms
  - MATLAB: ~251 ms
  - Controller: ~3 ms
- Total Time: ~265 ms

*Averaging from 50 tests for each sample

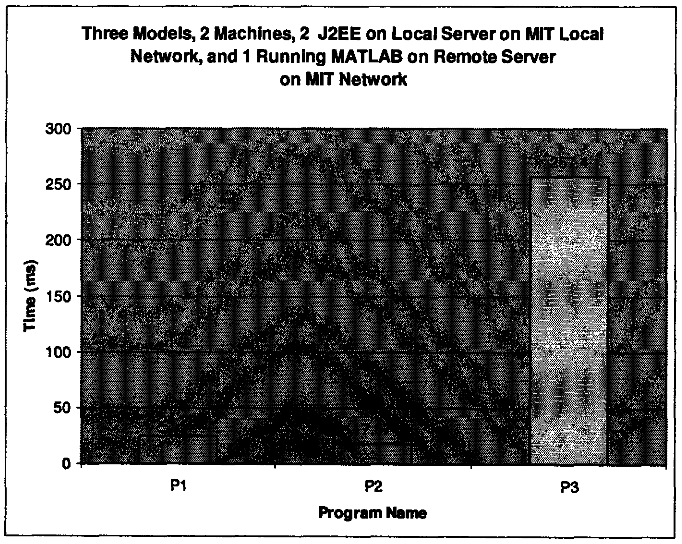


Figure 4-27 -Results from Test Case 2.

Test Case 3

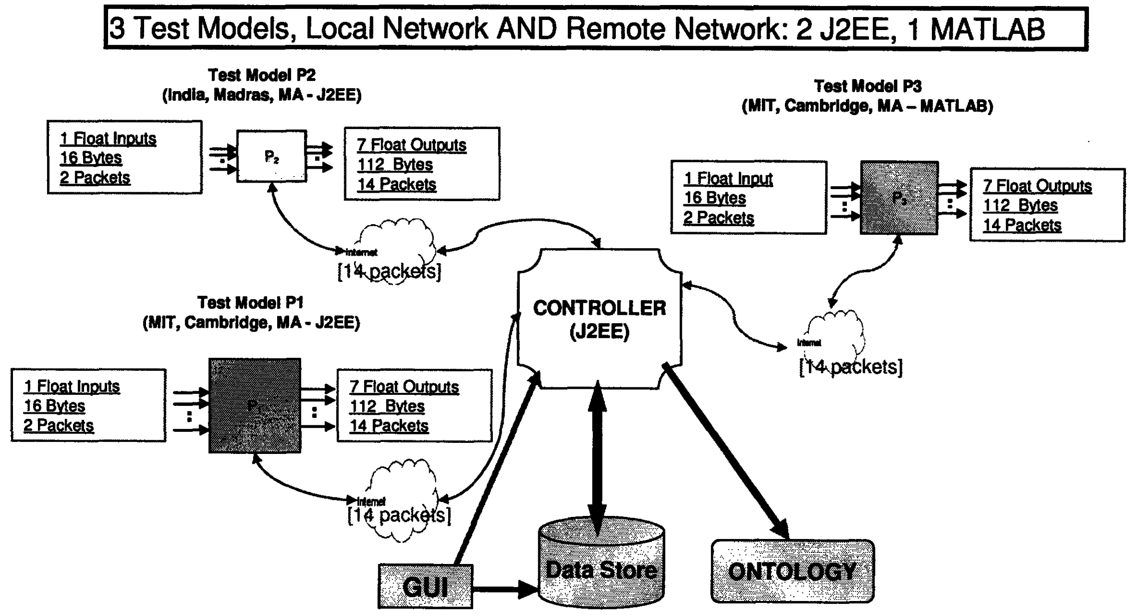


Figure 4-28 - Implementation test for Test Case 3.

In the third test, three models are setup on three different computers. Two of the models are setup within a local network and the third model is setup at a remote network, as shown in Figure 4-28. Each model is a simple calculation which takes as its input on value and performs the same and simple mathematical calculation within each model to return seven values. Two models are in the same format: J2EE and one model is in MATLAB.

The results from performing Test Case 2 are shown in Figure 4-29.

3 Test Models, Local Network AND Remote Network: 2 J2EE, 1 MATLAB

- Model Formats
  - 1 on J2EE, Local
  - 1 on J2EE, Remote
  - 1 on MATLAB, Local
- Transmission Time (ms)
  - Local: ~6 ms
  - Remote: ~654 ms
- Computation Time (ms)
  - J2EE: ~9 to 18 ms
  - MATLAB: ~257ms
  - Controller: ~3 ms
- Total Time: ~673 ms

*Averaging from 50 tests for each sample

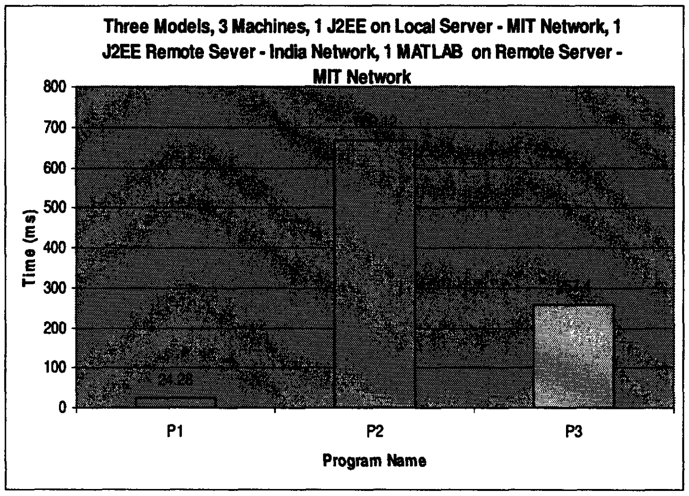


Figure 4-29 - Results from Test Case 3.

4.10 Summary

The results from the above three implementation tests provide us with an understanding of the architecture's performance. Specifically, three types of timings are involved:

1. Transmission Time
2. Model Computation Time
3. Controller Computation Time

These three times determine the Total Time for computing an integrated solution. The first and second test cases provided us with the Model Computation Time of a J2EE model to be ~9 to 18 ms per time step and a MATLAB Model Computation Time to be ~250 ms per time step. The Transmission Time across networks is summarized in summarized in Figure 4-30.

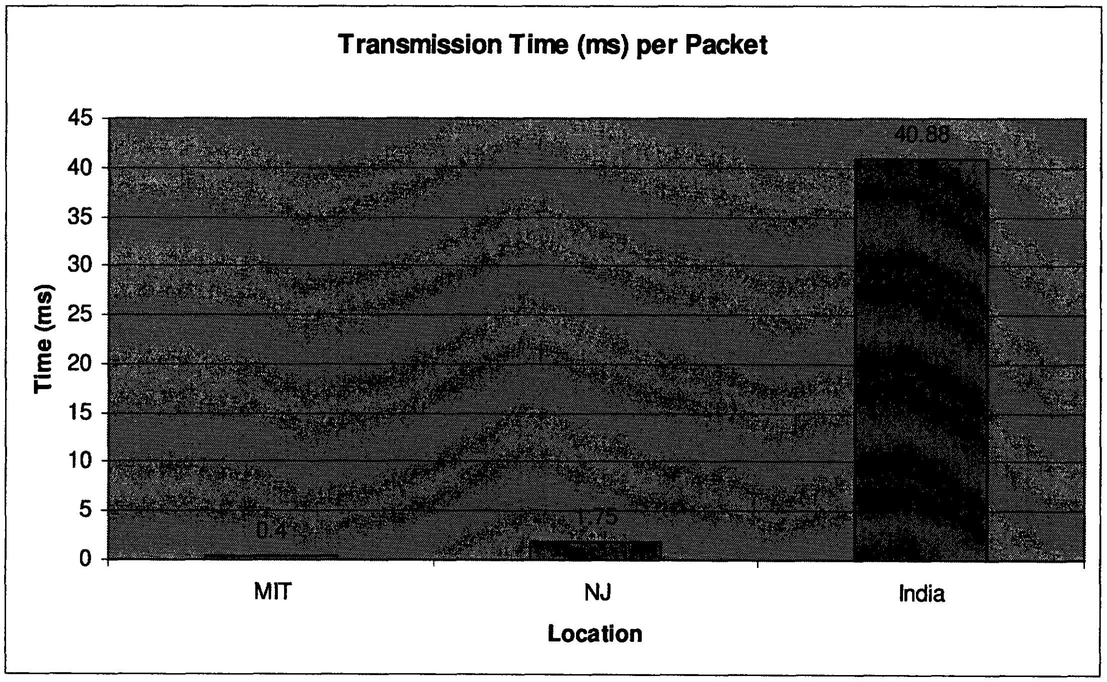


Figure 4-30 - Transmission time per packet across three different networks with Controller located at MIT.

For the total of 16 packets, which is what is transmitted per time step (2 packets outbound and 14 packets inbound), the Transmission Time for the Local Network is approximately ~6 ms per time step and for the Remote Network in

India is ~650 ms per time step. The Controller Time is approximately ~3 ms per time step. The results are summarized in Figure 4-31.

Various factors affect the Transmission Time including: Network hops, CPU-to-NIC (Network Interface Card), and Network bandwidth and traffic. The Model Computation Time is affected by: Hardware and local CPU power, Software Operating System (O/S), Software implementation (e.g. MATLAB, C, J2EE, etc.), and Mathematical representation (e.g. ODE, Stochastic, Boolean Networks, etc.). The Controller Computation Time is also affected by these same factors *and* the number of models that need to be integrated.

Chapter 5

EGFR Model of Kholodenko

5.0 Introduction

This chapter serves to validate and to test the Cytosolve architecture. In the previous Chapter, a simple problem was used to understand the elements affecting the computation time. In this Chapter, we use Cytosolve to solve a well known biological model in the systems biology community. The purpose of this effort is: 1. To validate the architecture's capability to produce known results, 2. compare our new approach with an existing monolithic approach, and 3. demonstrate the scalability and ease of use of the Cytosolve approach. In the next section, the methodology for performing this validation is presented. In the third section, the results from this effort are presented. In the fourth section, conclusions are made.

5.1 Materials and Methods

Materials

Two elements are necessary to execute this validation: 1. a known ensemble of biological pathway models along with their fully coupled system, previously solved in a monolithic approach, and 2. a popularly used monolithic approach for solving the model in order to compare the resulting solutions with Cytosolve. Relative to (1), the biological model the Epidermal Growth Factor Receptor (EGFR) model published by Kholodenko (Kholodenko, et al., 1999). The EGFR model is selected since known solutions exist for this problem thus enabling direct confirmation of the Cytosolve approach. In BioModels.Net this model has been instantiated into SBML, which can be solved using various monolithic approaches. This pathway can be viewed as an ensemble of four different biological pathways as shown in Figures 5-1 through 5-4, when coupled yields the full EGFR model shown in Figure 5-5.

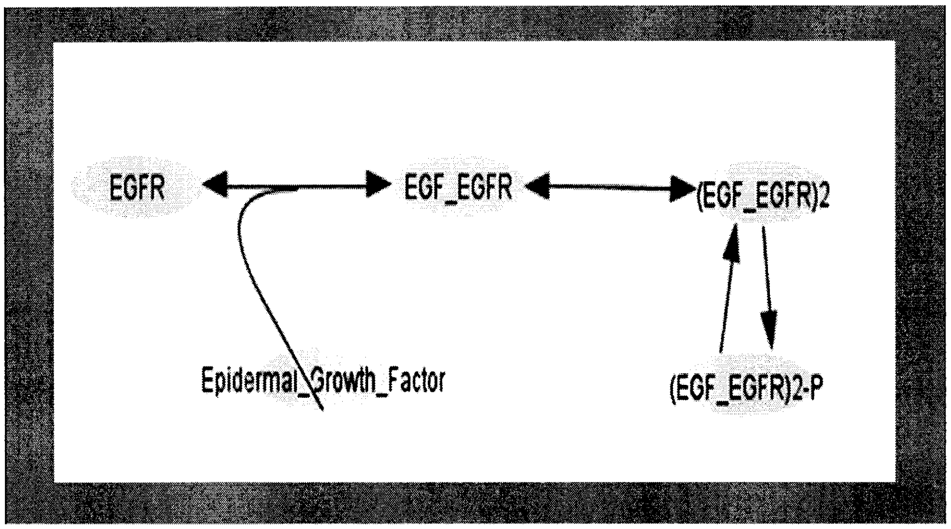


Figure 5-1 – Model 1: EGFR Dimerization Pathway.

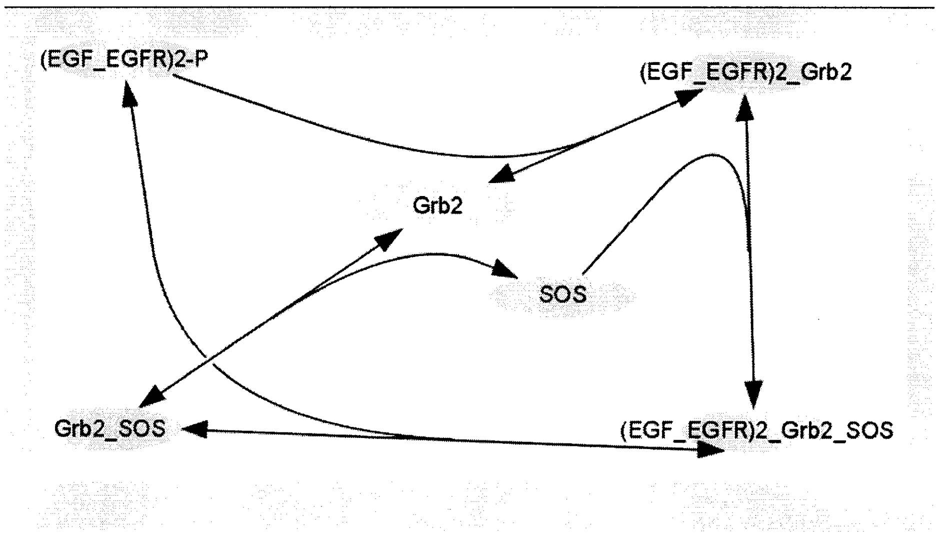


Figure 5-2 – Model 2: SOS Production Pathway.

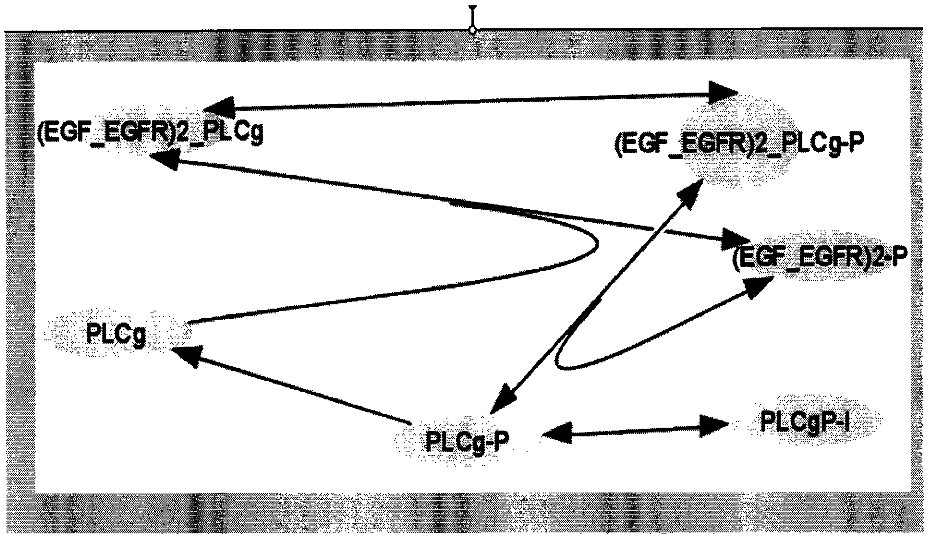


Figure 5-3 – Model 3: PLCg Production Pathway.

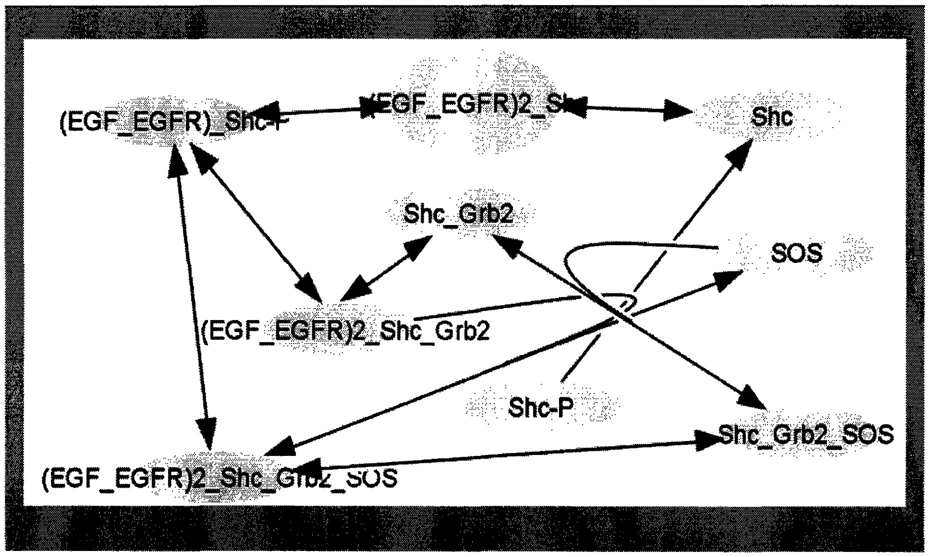


Figure 5-4 – Model 4: Shc Production Pathway

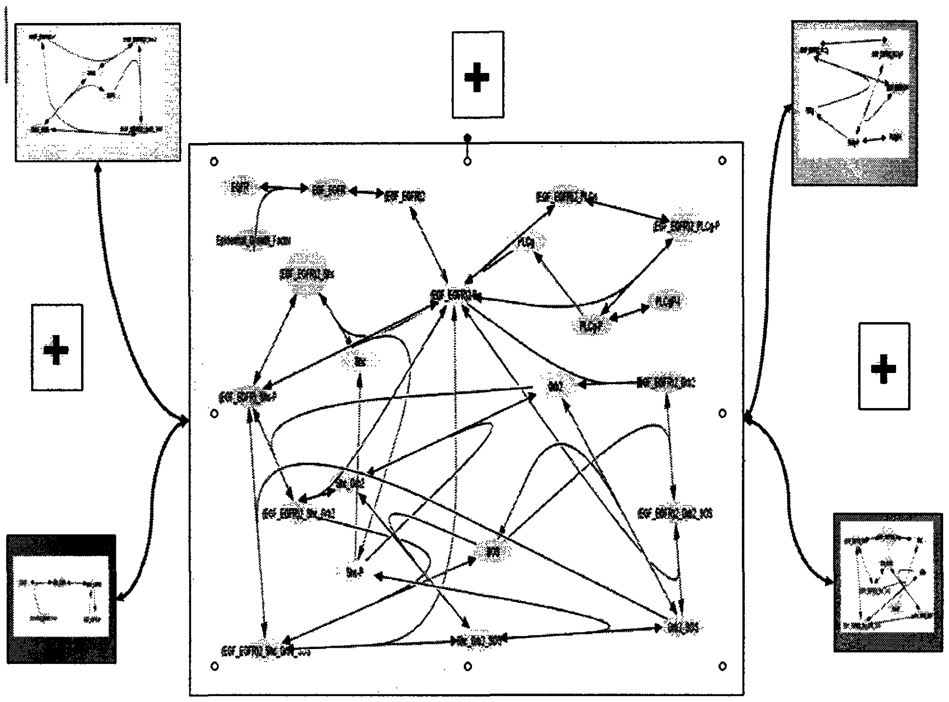


Figure 5-5 - EGFR Model

Relative to (2), for the selection of a monolithic approach, we select Cell Designer by (Kitano, et al., 2005) to compare our method. There are over 130 other systems such as Cell Designer that could have been selected; this tool was selected primarily based on its current popular use in the systems biology community and it is free. Cell Designer provides both a graphical mechanism for constructing the pathway diagram shown in Figure 5-5 as well as an ordinary differential equation (ODE) solver for calculating the various species concentrations values over time. In Figure 5-5, the creator of this pathway in Cell Designer had to "by hand" draw each and every species and then connect the species and instantiate the rate equations. Cell Designer requires the entire pathway to be coded into the Cell Designer system exclusively using the Cell Designer program prior to solving the pathway model. The total number of time steps used in the simulation is 100 or N=100, and the physical time is 10 seconds.

Method - Monolithic Approach

First, for each of the pathways shown in Figures 5-1 to 5-4, they were each loaded into Cell Designer separately and solved separately to produce baseline solutions for each pathway. Second, each of the individual pathways shown in Figures 5-1 to 5-4 were manually "hand-wired" together to produce full EGFR model shown in Figure 5-5. The resulting species concentrations over time of each of the species in the integrated EGFR model was output and documented.

Method - Cytosolve Approach

Cytosolve was then used to solve the same EGFR problem but in a distributed fashion. In Cytosolve, any *one* pathway model can exist in any format be it SBML, CellML, C++, C, FORTRAN, MATLAB, or any programming language, and there is no need to manually load, understand and interconnect each individual pathway, as is required in monolithic systems. In this case, to prove the efficacy of Cytosolve and to simulate the concept of four different teams working in four different locations world wide, each of the models was distributed on four different computers within a local area network. The total number of time steps is 100 or N=100, and the physical time is 10 seconds.

The results from Cytosolve were then compared with Cell Designer for both individual and the fully integrated models. Cell Designer and Cytosolve's central *controller* are executed on a Pentium 4 CPU 3.00 GHz Dell Workstation with 2 GB of RAM running Windows XP with Service Pack 2. In Cytosolve, each pathway model is treated as an independent entity, and is activated by messaging with the central controller, as described in Chapter 4, to ensure mass conservation. Each of the individual models, in the Cytosolve case, are also executed on a Pentium 4 CPU 3.00 GHz Dell Workstation with 2 GB of RAM running Windows XP with Service Pack 2.

5.2 Results

There are two sets of results. The first set of results provides the comparison of each individual model (Figures 5-1 to 5-4) executed in Cell Designer and in Cytosolve. The second set of results provides the entire EGFR model executed in both Cell Designer and Cytosolve.

In reviewing the four models, in Figures 5-1 to 5-4, one will recognize that the species **(EGF_EGFR)2-P** is shared by all four models; however the species **SOS** is shared only between the models in Figures 5-3 and 5-4.

Individual Model Solutions

Figure 5-6 provides the results of Cytosolve's time for solving each problem individually. The first column lists each model, the second column is the

Model	Transmission Time (ms)	Model Computation Time (ms)	Controller Time (ms)	Total Time (ms)
Model 1	822	1795	520	3137
Model 2	910	2265	476	3651
Model 3	915	2280	507	3702
Model 4	1405	2615	532	4552

Figure 5-6: Time for executing Cytosolve for each individual model.

Transmission Time involved for the Controller to communicate for 100 time steps to the model. The third column is that particular model's local Model Computation Time to execute the problem. The fourth column is the Controller Computation Time. The fifth column is the Total Time.

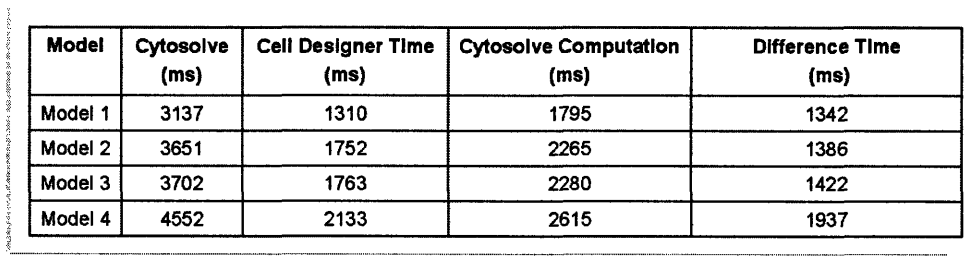
Figure 5-7 provides the results of Cell Designer's time for solving each problem individually.

Model	Transmission Time (ms)	Model Computation Time (ms)	Controller Time (ms)	Total Time (ms)
Model 1	N/A	1310	N/A	1310
Model 2	N/A	1752	N/A	1752
Model 3	N/A	1763	N/A	1763
Model 4	N/A	2133	N/A	2133

Figure 5-7: Time for executing Cell Designer for each individual model.

In this case, since Cell Designer is executed monolithically, there are no times for column 2 and column 4. Only, the individual Model Computation Time contributes to the Total Time.

Figure 5-8 compares the Cytosolve and Cell Designer compute times.



Model	Cytosolve (ms)	Cell Designer Time (ms)	Cytosolve Computation (ms)	Difference Time (ms)
Model 1	3137	1310	1795	1342
Model 2	3651	1752	2265	1386
Model 3	3702	1763	2280	1422
Model 4	4552	2133	2615	1937

Figure 5-8: Comparison of time between Cytosolve and Cell Designer.

Column 2 and column 3 are the Total Time's for Cytosolve and Cell Designer, respectively. Column 4, for Cytosolve, is the Model Computation Time on the local server. This time should be the same as Column 3; however, it is not since the local server's compute time on Cytosolve also involves read's and write's to the Global Vector and Local Vector. Thus, there is an overhead. The Difference Time in Column 5 is the difference between Column 2 and Column 4. This time offers us insights into overhead of Cytosolve's Transmission Time and Controller Time. Nearly 65% of this Difference Time is for Transmission Time and the remaining 35% for the Controller Time to perform the integration calculation.

Note, that for Cell Designer, each model was loaded in one at time and then executed. For Cytosolve, Cytosolve's central controller was implemented on one server and each model was implemented on another server. The results in above figures were calculated as the RMS average across fifty test runs for various species concentrations.

Whole EGFR Model Solution

In this case, the full integration of all four models is performed to derive the whole EGFR model in Figure 5-5. For Cell Designer, all four models were loaded into the Cell Designer system and had to be connected by hand to recreate the diagram in Figure 5-5. This process took nearly four several days to perform and to ensure consistency and accuracy of the pathway as described by Kholodenko. For Cytosolve, the central controller was run on one machine and four separate computers were setup, each running one independent model. This process took less than four hours. Recall, the goal in this exercise was to evaluate the difference in solution between Cytosolve and Cell Designer as well as computational time differences for deriving the whole EGFR model.

Model	Transmission Time (ms)	Model Computation Time (ms)	Controller Time (ms)	Total Time (ms)
Integrated	1424	2615	1893	5932

Figure 5-9: Compute time for integrated model using Cytosolve.

In Figure 5-9, the results from executing the integrated model using Cytosolve are shown. In this case, it is important to note that Column 2 and Column 3 are the maximum Transmission Time's along any Controller-Model path. This means that Column 2 is the longest Transmission Time taken by the Controller to communicate with any one of the model's over 100 time steps, and that Column 3 is the sum of the longest Computation Time across all the models. It is interesting to note that the compute time of 2615 in Column 2 is the compute time of the longest model, Model 4. Cytosolve took a total of 5932 ms to solve the integrated model.

Model	Transmission Time (ms)	Model Computation Time (ms)	Controller Time (ms)	Total Time (ms)
Integrated	N/A	3217	N/A	3217

Figure 5-10: Compute time for integrated model using Cell Designer.

In Figure 5-10, the results from executing the integrated model using Cell Designer are shown. In this case, the total time is 3217 ms to solve the integrated model. Cytosolve requires approximately two times the time to solve the integrated model; however, most of this time appears to be spent in the Transmission Time and then the Controller Time for integration

The above discussion focused on comparing the computation times of the two different approaches. Figure 5-11 and Figure 5-12 illustrate the comparison of actual solutions for the EGF and bound EGF-EGFR concentration profiles, respectively, from Cytosolve and Cell Designer.

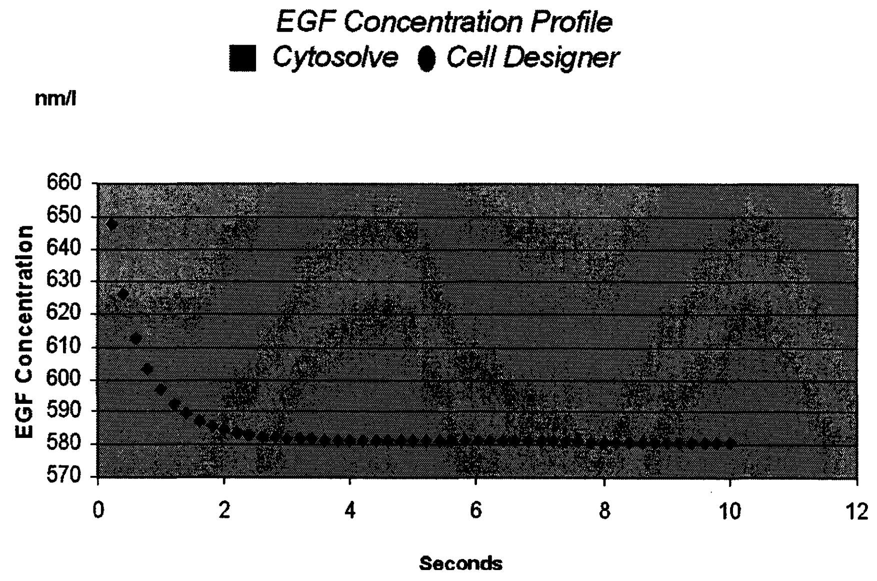


Figure 5-11 - Solving EGF using Cytosolve and Cell Designer.

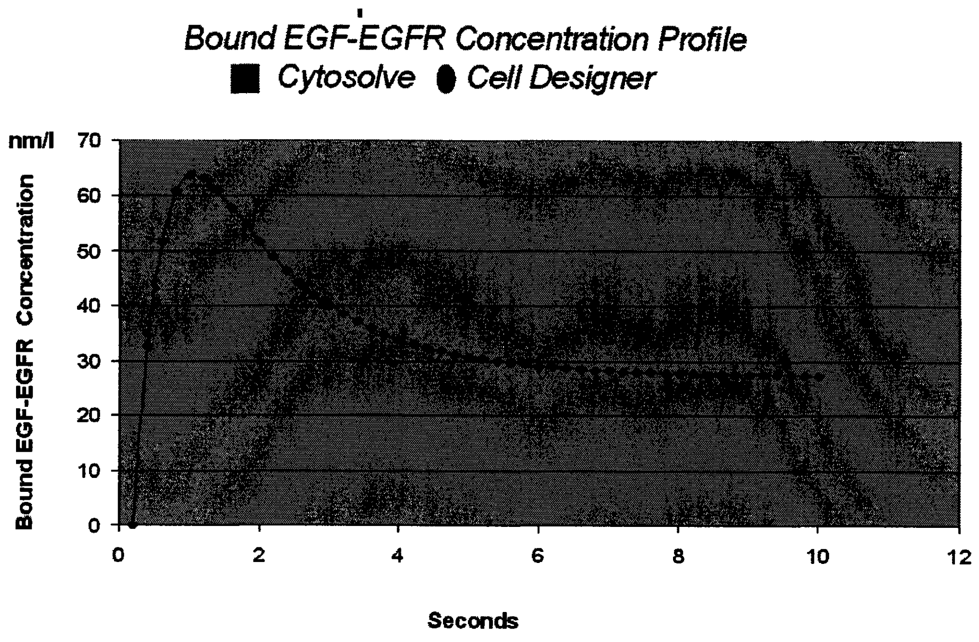


Figure 5-12 - Solving bound EGF-EGFR using Cytosolve and Cell Designer.

The above two figures show that there is little difference between the Cytosolve and Cell Designer solutions. There was less than 0.01% difference between both solutions.

5.3 Summary

The results demonstrate the viability of Cytosolve's unique distributed approach not only to solve problems that monolithic approaches are capable of solving but also to provide greater flexibility and scalability in integrating multiple biological pathway models, which monolithic approaches are incapable of doing. In Cytosolve, any *one* pathway can exist in any format on any computer, and there is no need to manually load, understand and interconnect each individual pathway, as is required in monolithic systems.

Cytosolve generated exact results to Cell Designer; more importantly, the integration of the four models in Cytosolve did not require any manual "wiring" as is needed by Cell Designer. Cytosolve's compute time was greater than Cell Designer; however, most of this compute time was due to Transmission Time. Since Cytosolve works in a distributed parallel fashion, its compute time is a direct function of the compute time of the largest pathway plus the associated Transmission Time and overhead for Controller Time to integrate. For Cell Designer, the compute time will be the compute time of the whole integrated pathway.

Initial results from the EGFR example have demonstrated that Cytosolve can serve as an alternative to the monolithic approaches for integrating and solving biomolecular pathways. Most important is Cytosolve's core feature for integrating multiple pathway models, which can be distributed across multiple computing systems, without "hand wiring" of each model. While such a manual approach may be viable for a handful of models, it will not scale to support the integration of all pathway models necessary to model the whole cell. Moreover, the monolithic approach does not provide a means for pathways from proprietary models to be used with other models that are open source. An architecture such as Cytosolve will allow individual research teams to contribute the output of their pathway models to an external dynamic network of models without revealing the details of their internal structure. There has been also no research to show that monolithic pathways can be distributed between machines for computational scalability. The Cytosolve approach parallelizes the computations from the beginning, making computational parallelization automatic.

Finally, and perhaps equally important, is that managing a monolithic model, composed of other models, is a change management nightmare. Consider a small example of a monolithic model "cut and pasted" or concatenated from the four models of EGFR, aforementioned, and each model being published and created by different authors. Now, suppose once the monolithic model has been constructed, that many months later, the authors of each of these models changes rate constants, pathway connections, etc., at that point the author of the monolithic model would have to rebuild the entire monolithic model, by instantiating changes from each author's model, which may be tenable for four models (possibly based on the complexity and domain specificity of each model). Modeling the whole cell while managing such changes across a suite of hundreds of such models will be untenable.

In summary, the results are the same as monolithic approach. Cytosolve's local Model Computation Time is approximately 30% to 40% more than the monolithic approach. Cytosolve's integrated model requires no more than 2x amount than the monolithic approach. Cytosolve's "overhead" equals the Transmission Time + Controller Time. Of the total "overhead" for Cytosolve ~65% is for Transmission Time and ~35% is for Controller Time.

Chapter 6

Integrative Model of Interferon Response to Virus Infection

6.0 Introduction

The purpose of this chapter is to solve a heretofore unknown problem by integrating multiple biological pathway models. In this chapter we begin by first giving a background on Interferons (IFNs). The next section gives an overview of the four key biological pathway models involved in the IFN response to virus infection. The third section itemizes the key molecular agents involved in IFN response. The fourth section details each biological pathway model within the IFN response mechanism. The fifth section provides the solution for each individual pathway model using the Cytosolve architecture. The sixth section provides the complete solution of the IFN response to virus infection using the Cytosolve architecture. The seventh section performs a test of replacing one of the pathway models within the integrated IFN model to demonstrate Cytosolve's ability to easily integrate updates to a given model in the ensemble. The eighth section uses the integrated model to study new biological phenomena of the IFN response to virus infection. The final section provides a summary and discussion of the results.

6.1 Background on Interferons

The immune system has many different types of cells acting together to protect the body against viruses, bacteria, and other "foreign invaders." Part of this protection includes the production of interferon (IFN), a protein that plays a special role in triggering the body's response. The following describes what interferon is and why it is so important to the immune system.

What Is Interferon

The immune system consists of a complex network of cells, tissues, and organs all working in tandem to ward off infection and keep us healthy. This includes interferon, one of the proteins called cytokines, which are diverse and potent chemical messengers that can trigger the immune system to attack invading pathogens. Interferon signals neighboring cells into action and also interferes with how foreign cells grow and multiply. Interferon is also considered essential for optimal health because it can boost the immune system's ability to recognize foreign invaders. It is because of this special role that interferon is used in drug form as an anti-viral agent to treat many different diseases. Moreover, researchers have shown that interferon, given by nasal spray in daily doses, can prevent infection and illness. However, pharmaceutical forms of interferon cause side effects such as nosebleeds, fatigue, headache and aches, and may not be useful in treating established colds (Wikipedia, 2007).

In humans, IFNs also play a roles in cell growth, differentiation and immunomodulation. IFNs are divided into two groups depending on their molecular basis; type I IFNs (IFN-alpha and IFN-beta) are produced by a variety of cells following virus infection, and type II IFN (IFN-gamma) is produced by activated T cells and natural killer (NK) cells (Sato, 2001). There are three classes of Interferon, alpha, beta and gamma. Interferon alpha and beta are produced by many cell types, including the infection-fighting T-cells and B-cells in the blood, and are an important component of the anti-viral response. In contrast, interferon gamma is involved in the regulation of the immune and inflammatory responses and is produced by activated T-cells (Sato, 2001).

The History of Interferon

Since more than half of the communicable diseases affecting human beings are caused by viruses, scientists in the 1950's began searching for clues into how the body protects itself against viruses, leading to the discovery of interferon.

During studies on virus replication, two groups of researchers in different parts of the world separately discovered interferon. The first discovery occurred in Japan in 1954 when researchers at Tokyo University were studying viruses in rabbits and found that a natural protein made the rabbits resistant to subsequent viral infection. Then, in 1957, Scottish virologist Alick Issacs and Swiss scientist Jean Lindenmann found that when chick embryos were injected with influenza virus, the protein produced by the cells destroyed the virus and also inhibited the growth of any other viruses in the embryos. Isaacs and Lindenmann named the protein interferon because of its ability to interfere with virus replication (Isaacs, 1957).

Further research showed that interferon was produced within hours of a viral invasion (antibodies take several days to form) and that most living things, including plants, can make the protective protein. Interferon was seen as the cell's first line of defense against viral infections, but because the body produces interferon in small amounts and the protein was thought to be species-specific – meaning only human interferon will work in human beings – research on the use of interferon in drug form inched forward at a snail's pace. Then, in the late 1960's, Ion Gresser, an American researcher working in Paris, and the Finnish virologist Kari Cantell developed a way to make interferon in useful amounts from human blood cells. Monoclonal antibodies, first produced in 1975, made large-scale purification of interferon possible, and the mid-1980s saw the advent of genetically engineered interferon (Glick, 2006). During the same period, scientists learned that there are three classes of interferon and that these Interferons are not species-specific but can produce a response in other species .

While these developments were occurring, Japanese researchers were focusing on interferon-inducing activities in Chinese herbal medicines. This led extensive research on ways to boost the body's ability to produce interferon through the interaction of botanicals. After screening, testing and evaluating over 200 different herbs, the research successfully isolated four botanicals that, in combination, naturally increase the body's production of its own interferon. As a result of this extensive research, interferon is being used today in drug form to treat viral diseases like rabies, hepatitis, and herpes infections. At the same time, new research now makes it possible for healthy adults to boost their immune system through a dietary supplement that naturally increases the production of interferon in the body.

How Interferon is Used To Treat Diseases and Boost Immunity

In drug form, several different types of interferon are now approved for use in humans, and are usually administered as an intramuscular injection. Interferon alpha is used as a cancer therapy and a treatment for Hepatitis C, the AIDS-related Kaposi's sarcoma and genital warts. Interferon beta is used in the treatment and to control the neurological disorder multiple sclerosis (Glick, 2006).

In therapeutic doses, interferon can be hard to tolerate because of its side effects, which include fatigue, headache and aches, and less frequently, low thyroid activity, low platelet count and depression. It is because of these side effects that researchers have not pursued the use of interferon alpha for the common cold and flu, even though studies find that interferon, given in daily doses by nasal spray, can prevent infection and illness. However, new research now makes it possible for healthy adults to boost their immune system through a dietary supplement that naturally increases the production of interferon in the body.

In summary, IFNs perform the following key functions:

- Inhibit virus replication (combat viral infections)
- Inhibit cell growth (used in anti-cancer therapy)
- Activate monocyte/macrophages
- Inhibit non-viral intracellular pathogens
- Produce pyrogenic activity (the fever you get with a cold)

6.2 Key Molecular Components of IFN Activity

The IFN response activity is evident across the extra cellular matrix (ECM), the cell membrane, the cytoplasm and the nucleus as shown in Figure 6-1 (Taniguchi, 2001).

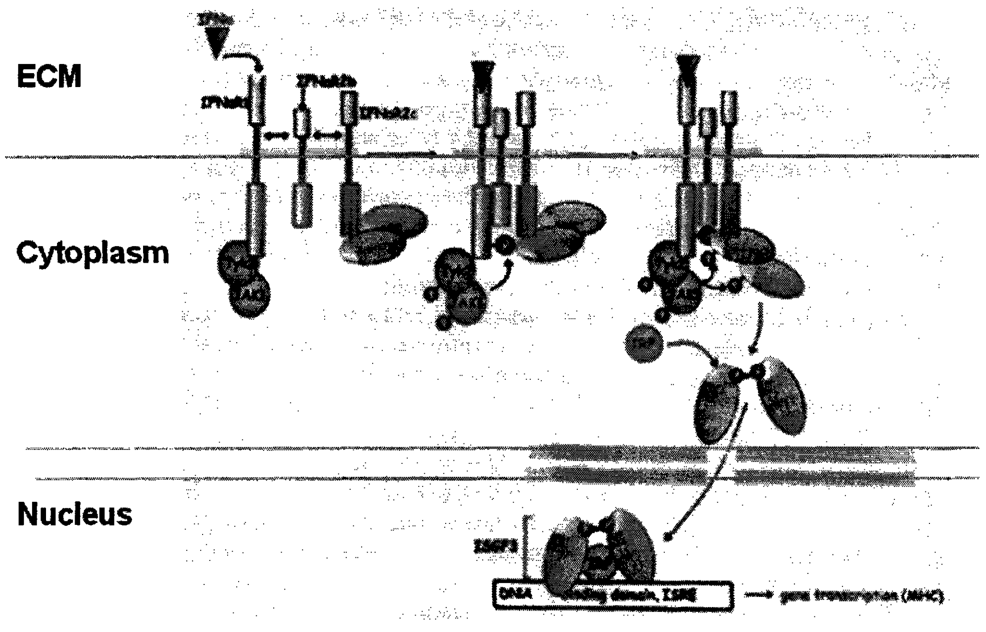


Figure 6-1: IFN signaling affects various cellular components (Taniguchi, 2001).

IFN Receptors

IFNs actions are exerted through specific cell surface receptors.

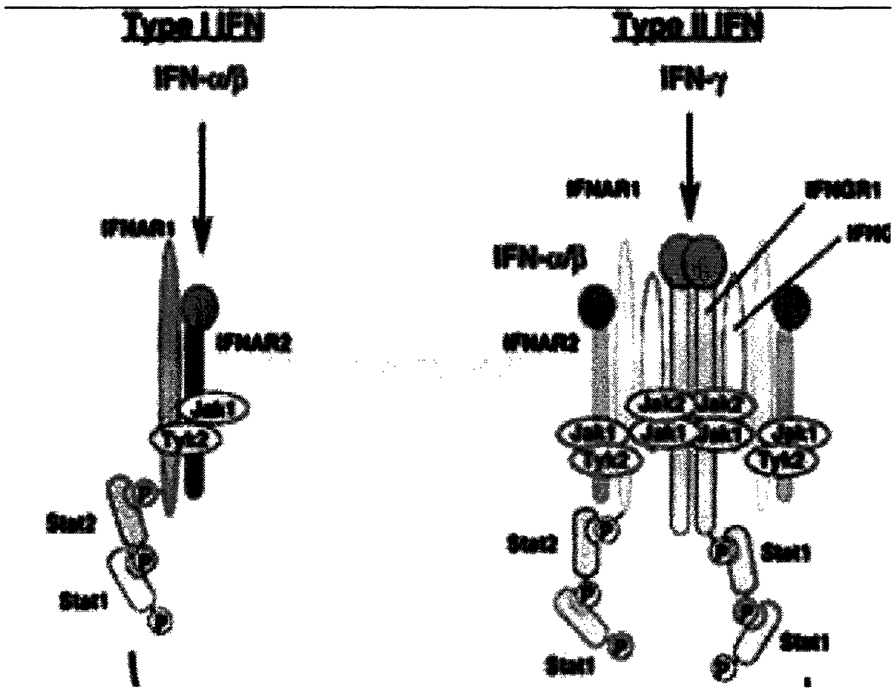


Figure 6-2: IFN receptors (Taniguchi, 2001).

While new ones, no doubt, will be discovered, it is currently known that IFNα, IFNβ and IFNω appear to have a common receptor and that IFNγ binds to a different receptor that has 2 subunits, as shown in Figure 6-2. IFN signaling involves an IFN-mediated hetero-dimerization of the cell surface receptor subunits as shown in the figure.

Signal Transducers and Activators of Transcription (STATs)

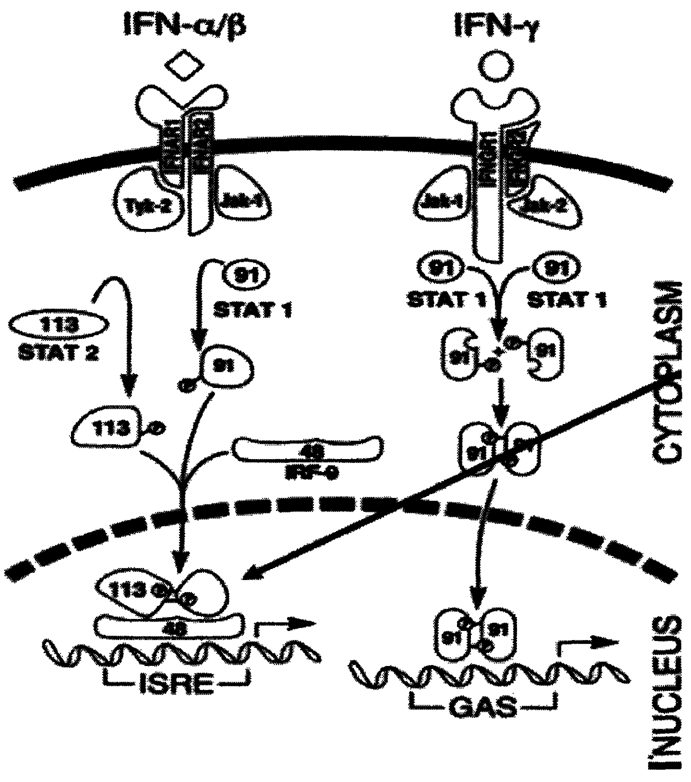


Figure 6-3: STATs in action (Taniguchi, 2001).

STATs are latent transcription factors in the cytoplasm. They are activated and then translocated to nucleus. Their activation is supported through the phosphorylation by the receptor-associated Janus family of tyrosine kinase (JAK) enzymes in response to cytokine stimulation, as shown in Figure 6-3. There are different members of the JAK and STAT families have distinct functions in cytokine signaling. For example, Jak-1, Jak-2, Tyk-2, STAT-1 and STAT-2 play central roles in mediating IFN-dependent antiviral activities.

Interferon Regulatory Factors (IRFs)

IRFs are transcriptional regulators important in regulating the interferon response. Some IRFs are induced by IFN signaling. Historically, there are nine known family members: IRF-1 to IRF-9 . Recently tenth one was discovered. IRFs and STATs function together to induce and control the expression of proteins that constitute the antiviral state.

6.3 Elements of the IFN Response

The IFN response mechanism of the cell to virus infection is a core cellular function. There are four key biological pathways which are involved to elicit IFN response to virus infection:

- Up regulation of IFN-Beta
- IFN receptor signaling to produce IRF-7
- Virus amplification cycle to produce more IFN-Beta and IFN-Alpha
- Regulation and balancing by SOCS-1

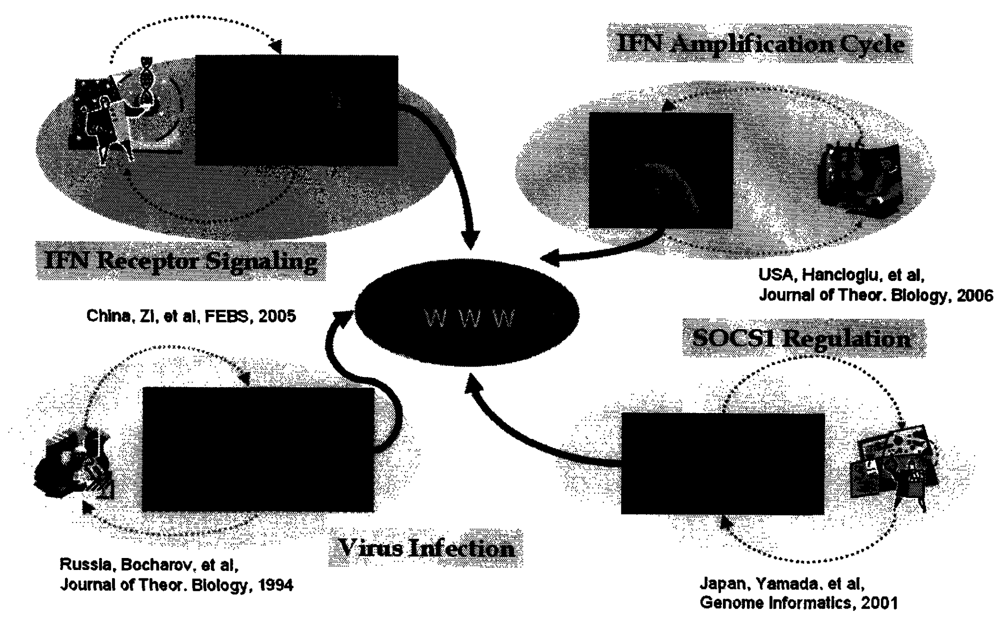


Figure 6-4 – Integration of the efforts of multiple research teams is required to develop an integrated model of IFN response to viral infection.

Figure 6-4 illustrates how the entire model of the IFN response involves the integration of work efforts from four countries and three continents. Each research paper used in this integration effort involves the work of multiple research groups.

Virus Infection Model

The high level biological pathway of virus infection is depicted in Figure 6-5 below. This pathway creates IFN-Beta as an initial response to virus infection. Scientists in Moscow, Russia in 1994 modeled this pathway in the *Journal of Theoretical Biology* (Bocharaov, 1994). The original code was written in MATLAB.

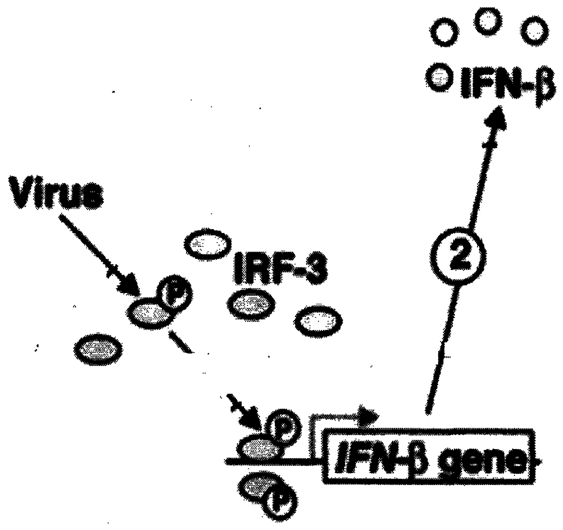


Figure 6-5 – Virus infection pathway (Bocharaov, 1994).

A detailed diagram of this pathway is shown in Figure 6-6 which includes all the species and molecular interactions. Here viruses inject their single-stranded RNA into the host cell, which leads to the formation of double-stranded RNA. Double-stranded RNA triggers the activation of virus-activated kinase (VAK), which phophorylates IRF-3. Phosphorylated IRF-3 is a transcription factor for the IFN-Beta gene. The expression of this gene results in the initial production of IFN-Beta.

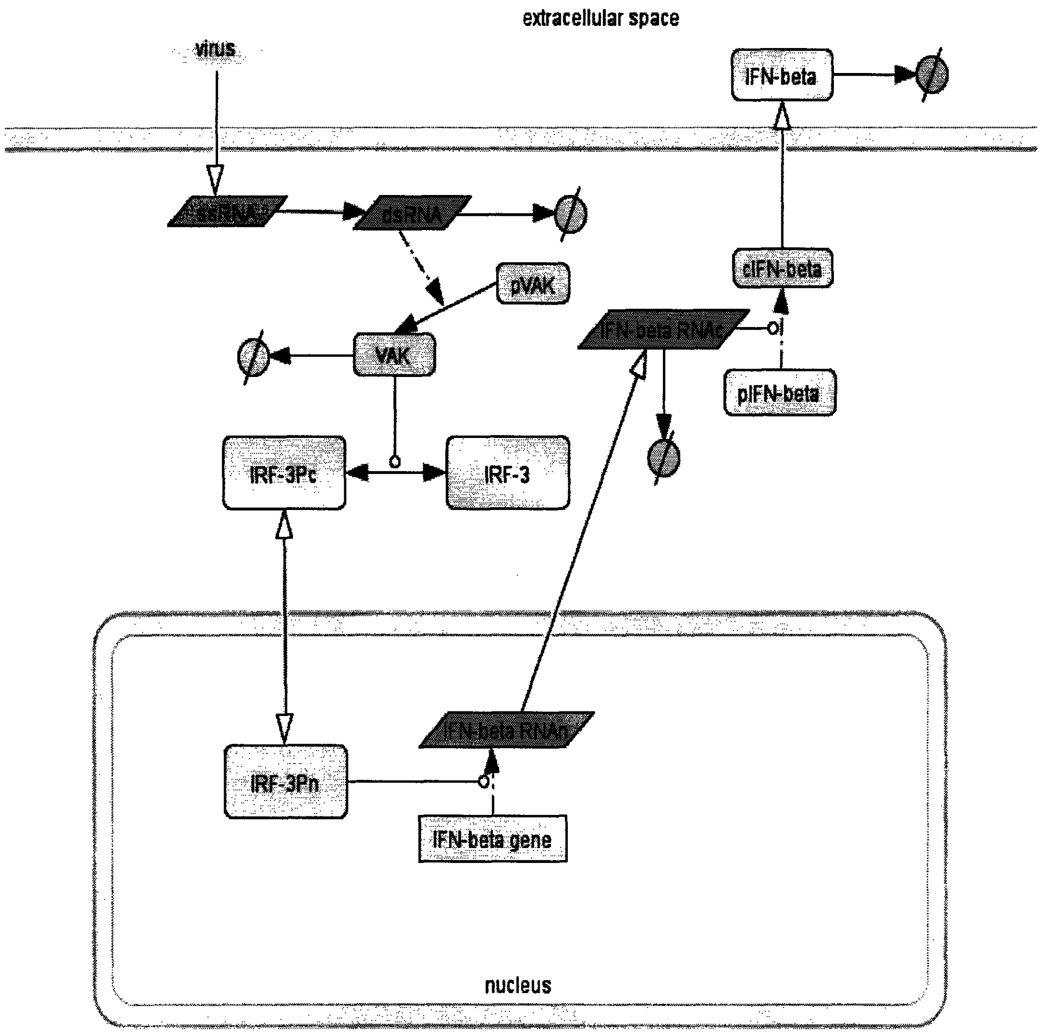


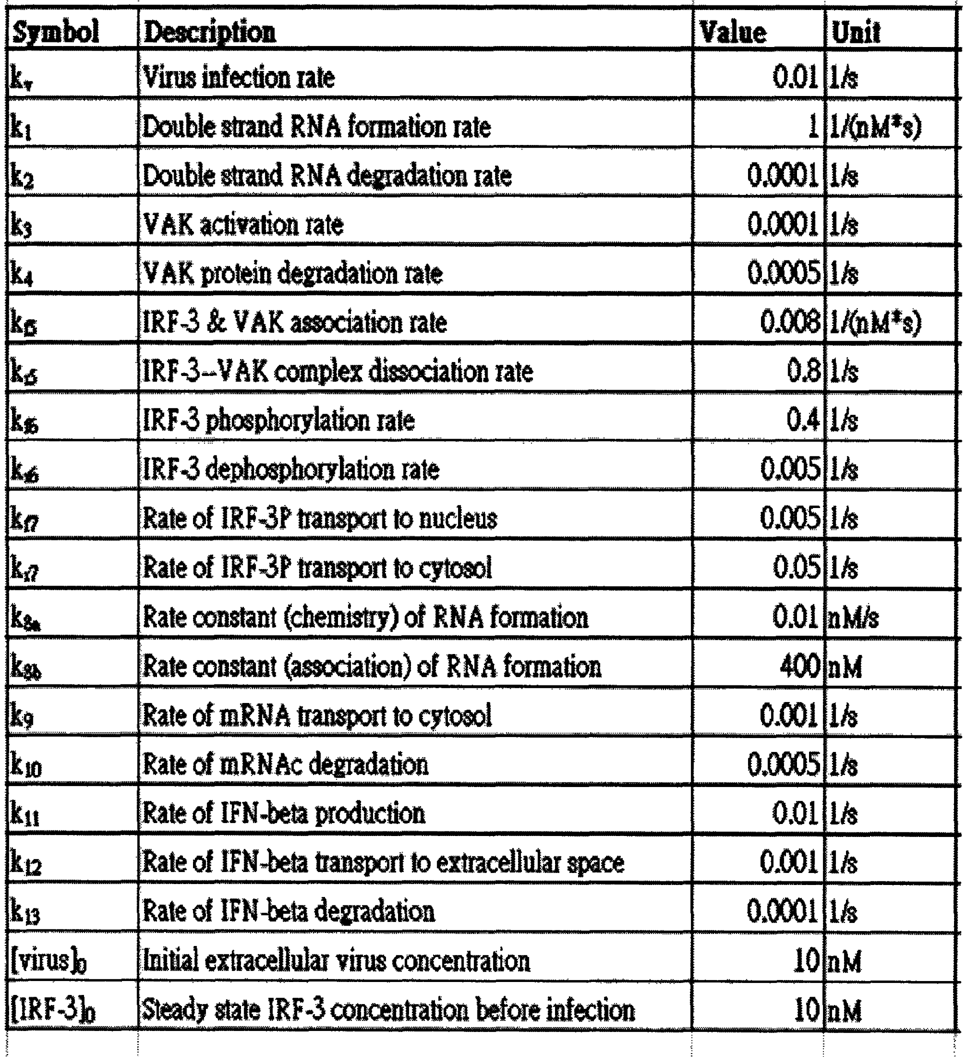
Figure 6-6 – Virus infection pathway detailed mechanics.

In Figure 6-7 are listed the differential equations defining the molecular interactions for this pathway.

Reactions	Math
[virus]→[ssRNA]	$\frac{dRxn}{dt} = k_v \cdot [virus]$
[ssRNA] →[dsRNA]	$\frac{dRxn}{dt} = k_1 \cdot [ssRNA]$
[dsRNA] degradation	$\frac{dRxn}{dt} = k_2 \cdot [dsRNA]$
[VAK] activation	$\frac{dRxn}{dt} = k_3 \cdot [dsRNA]$
[VAK] degradation	$\frac{dRxn}{dt} = k_4 \cdot [VAK]$
[IRF-3]←→[IRF-3Pc]	$\frac{dRxn}{dt} = \frac{k_{f6} \cdot [IRF-3] \cdot [VAK]}{[IRF-3] + \frac{k_{r5} + k_{f6}}{k_{f5}}} - k_{r6} \cdot [IRF-3Pc]$
[IRF-3Pc]←→[IRF-3Pn]	$\frac{dRxn}{dt} = k_{f7} \cdot [IRF-3Pc] - k_{r7} \cdot [IRF-3Pn]$
[IFN-beta RNAn] production	$\frac{dRxn}{dt} = \frac{k_{8a} \cdot [IRF-3Pn]}{[IRF-3Pn] + k_{8b}}$
[IRF-beta RNAn]←→[IFN-beta RNAc]	$\frac{dRxn}{dt} = k_9 \cdot [IFN-beta\_RNAn]$
[IFN-beta RNAc] degradation	$\frac{dRxn}{dt} = k_{10} \cdot [IFN-beta\_RNAc]$
[cIFN-beta] production	$\frac{dRxn}{dt} = k_{11} \cdot [IFN-beta\_RNAc]$
[cIFN-beta] →[IFN-beta]	$\frac{dRxn}{dt} = k_{12} \cdot [cIFN-beta]$
[IFN-beta] degradation	$\frac{dRxn}{dt} = k_{13} \cdot [IFN-beta]$

Figure 6-7 – Differential equations for viral infection pathway.

In Figure 6-8 are listed the internal parameters, rate constants and initial conditions for this pathway.



Symbol	Description	Value	Unit
k_v	Virus infection rate	0.01	1/s
k_1	Double strand RNA formation rate	1	1/(nM*s)
k_2	Double strand RNA degradation rate	0.0001	1/s
k_3	VAK activation rate	0.0001	1/s
k_4	VAK protein degradation rate	0.0005	1/s
k_{f5}	IRF-3 & VAK association rate	0.008	1/(nM*s)
k_{r5}	IRF-3--VAK complex dissociation rate	0.8	1/s
k_{f6}	IRF-3 phosphorylation rate	0.4	1/s
k_{r6}	IRF-3 dephosphorylation rate	0.005	1/s
k_{f7}	Rate of IRF-3P transport to nucleus	0.005	1/s
k_{r7}	Rate of IRF-3P transport to cytosol	0.05	1/s
k_{8a}	Rate constant (chemistry) of RNA formation	0.01	nM/s
k_{8b}	Rate constant (association) of RNA formation	400	nM
k_9	Rate of mRNA transport to cytosol	0.001	1/s
k_{10}	Rate of mRNAc degradation	0.0005	1/s
k_{11}	Rate of IFN-beta production	0.01	1/s
k_{12}	Rate of IFN-beta transport to extracellular space	0.001	1/s
k_{13}	Rate of IFN-beta degradation	0.0001	1/s
$[virus]_0$	Initial extracellular virus concentration	10	nM
$[IRF\text{-}3]_0$	Steady state IRF-3 concentration before infection	10	nM

Figure 6-8 – Parameters for viral infection pathway.

IFN Receptor Signaling Model

The high level biological pathway of IFN receptor signaling is depicted in Figure 6-9 below. This biological pathway produces IRF-7 as a preparation mechanism for the infected cell and neighboring cells. Scientists in China in 2005 defined this pathway model in *FEBS* (Zi, 2005). Only a pathway diagram along with parameters exists for this pathway model, but no software code.

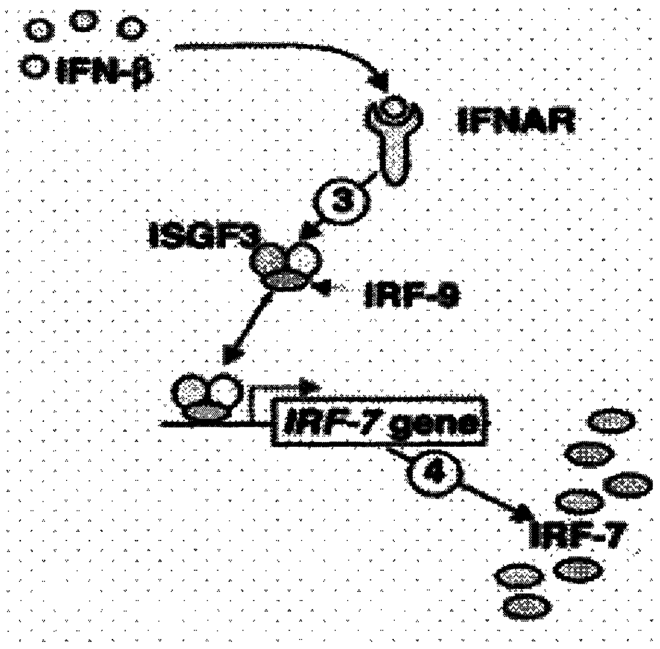


Figure 6-9 – IFN receptor signaling (Zi, 2005).

A detailed diagram of the this pathway is shown in Figure 6-10 which includes all the species and molecular interactions. In Figure 6-10, IFN-Beta (or IFN-Alpha) lands on the IFNAR receptor to initiate the up regulation of IRF-7 which is a critical protein for signaling the cell itself as well as neighboring cell of the virus infection. The binding of IFN with the receptor leads to the STAT protein being phosphorylated. The phophorylated STAT forms a homo-dimer and becomes the transcription factor for IRF-7 gene after binding to IRF-9, which leads to expression of IRF-7. This signaling mechanism *prepares* the cell for further defenses by producing IRF-7.

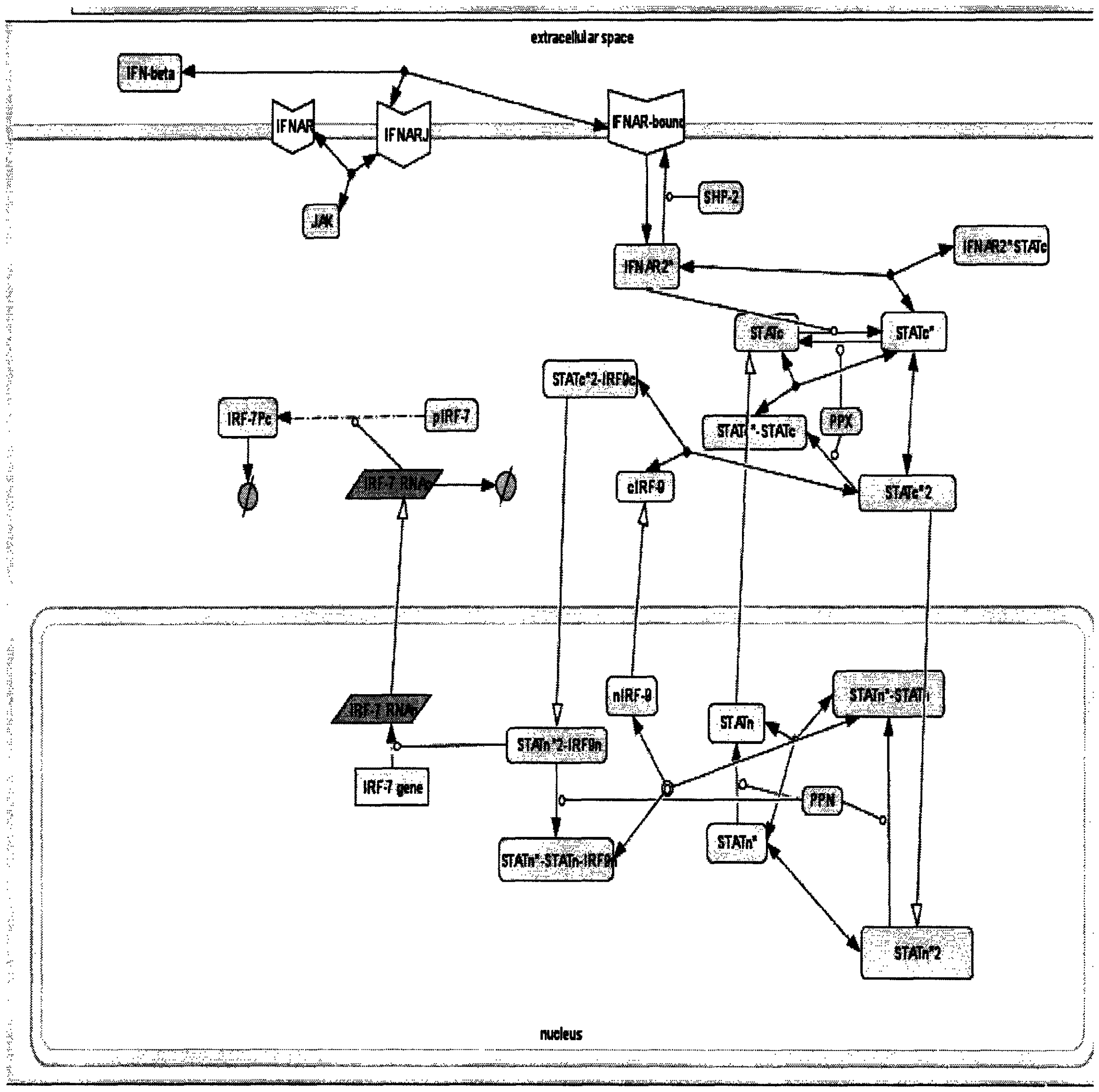


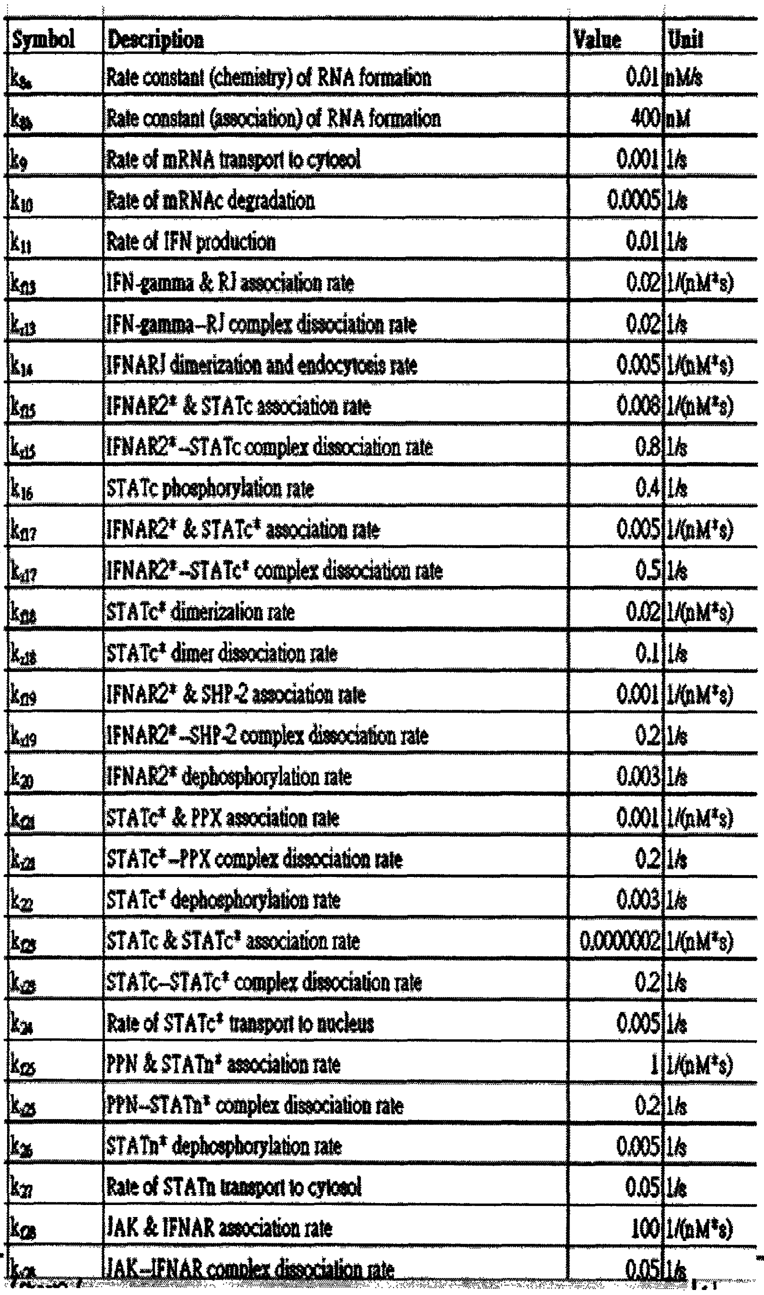
Figure 6-10 - IFN receptor signaling pathway detailed mechanics.

In Figure 6-11 are listed the differential equations defining the molecular interactions for this pathway.

Reactions	Math
[JAK] & [IFNAR] association	$\frac{dRxn}{dt} = k_{f28} \cdot [JAK] \cdot [IFNAR] - k_{r28} \cdot [IFNARJ]$
[IFNARJ] & [IFN-beta] association	$\frac{dRxn}{dt} = k_{f13} \cdot [IFN-beta] \cdot [IFNARJ] - k_{r13} \cdot [IFN-beta-bound]$
[IFN-beta-bound] dimerization	$\frac{dRxn}{dt} = k_{f33} \cdot [IFN-beta-bound]^2 - k_{r33} \cdot [IFN-beta-bound\_2]$
[IFN-beta-bound] phosphorylation	$\frac{dRxn}{dt} = k_{14} \cdot [IFN-beta-bound\_2]$
[IFNBR2*] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{20} \cdot [IFNBR2^*] \cdot [SHP2]}{[IFNBR2^*] + \frac{k_{r19} + k_{20}}{k_{f19}}}$
[IFNBR2*] & [STATc*] association	$\frac{dRxn}{dt} = k_{f17} \cdot [IFNBR2^*] \cdot [STATc^*] - k_{r17} \cdot [IFNBR2 * STATc]$
[STATc] phosphorylation	$\frac{dRxn}{dt} = \frac{k_{16} \cdot [IFNBR2^*] \cdot [STATc]}{[STATc] + \frac{k_{r15} + k_{16}}{k_{f15}}}$
[STATc*] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{22} \cdot [PPX] \cdot [STATc^*]}{[STATc^*] + \frac{k_{r21} + k_{22}}{k_{f21}}}$
[STATc] & [STATc*] association	$\frac{dRxn}{dt} = k_{f23} \cdot [STATc] \cdot [STATc^*] - k_{r23} \cdot [STATc * -STATc]$
[STATc*] dimerization	$\frac{dRxn}{dt} = k_{f18} \cdot [STATc^*]^2 - k_{r18} \cdot [STATc * 2]$
[STATc*2] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{22} \cdot [PPX] \cdot [STATc * 2]}{[STATc * 2] + \frac{k_{r21} + k_{22}}{k_{f21}}}$
[STATc*2] transport to nucleus	$\frac{dRxn}{dt} = k_{24} \cdot [STATc * 2]$
[STATn*-STATn] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{26} \cdot [PPN] \cdot [STATn * 2]}{[STATn * 2] + \frac{k_{r25} + k_{26}}{k_{f25}}}$

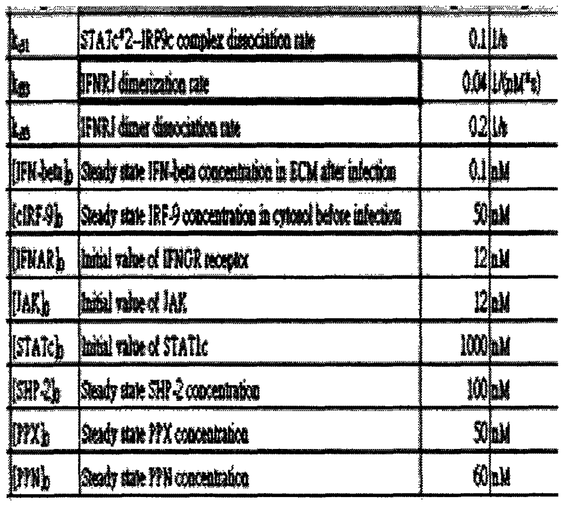
Figure 6-11 – Differential equations for IFN receptor signaling pathway.

In Figure 6-12a and 6-12b are listed the internal parameters, rate constants and initial conditions for this pathway.



Symbol	Description	Value	Unit
k_{8a}	Rate constant (chemistry) of RNA formation	0.01	nM/s
k_{8b}	Rate constant (association) of RNA formation	400	nM
k_9	Rate of mRNA transport to cytosol	0.001	1/s
k_{10}	Rate of mRNAc degradation	0.0005	1/s
k_{11}	Rate of IFN production	0.01	1/s
k_{f13}	IFN-gamma & RJ association rate	0.02	1/(nM*s)
k_{r13}	IFN-gamma--RJ complex dissociation rate	0.02	1/s
k_{14}	IFNARJ dimerization and endocytosis rate	0.005	1/(nM*s)
k_{f15}	IFNAR2* & STATc association rate	0.008	1/(nM*s)
k_{r15}	IFNAR2*--STATc complex dissociation rate	0.8	1/s
k_{16}	STATc phosphorylation rate	0.4	1/s
k_{f17}	IFNAR2* & STATc* association rate	0.005	1/(nM*s)
k_{r17}	IFNAR2*--STATc* complex dissociation rate	0.5	1/s
k_{f18}	STATc* dimerization rate	0.02	1/(nM*s)
k_{r18}	STATc* dimer dissociation rate	0.1	1/s
k_{f19}	IFNAR2* & SHP-2 association rate	0.001	1/(nM*s)
k_{r19}	IFNAR2*--SHP-2 complex dissociation rate	0.2	1/s
k_{20}	IFNAR2* dephosphorylation rate	0.003	1/s
k_{f21}	STATc* & PPX association rate	0.001	1/(nM*s)
k_{r21}	STATc*--PPX complex dissociation rate	0.2	1/s
k_{22}	STATc* dephosphorylation rate	0.003	1/s
k_{f23}	STATc & STATc* association rate	0.0000002	1/(nM*s)
k_{r23}	STATc--STATc* complex dissociation rate	0.2	1/s
k_{24}	Rate of STATc* transport to nucleus	0.005	1/s
k_{f25}	PPN & STATn* association rate	1	1/(nM*s)
k_{r25}	PPN--STATn* complex dissociation rate	0.2	1/s
k_{26}	STATn* dephosphorylation rate	0.005	1/s
k_{27}	Rate of STATn transport to cytosol	0.05	1/s
k_{f28}	JAK & IFNAR association rate	100	1/(nM*s)
k_{r28}	JAK--IFNAR complex dissociation rate	0.05	1/s

Figure 6-12a – Parameters for IFN receptor signaling pathway.



k_{31}	STATc*2–IRF9c complex dissociation rate	0.1	1/s
k_{33}	IFNRJ dimerization rate	0.04	1/(nM*s)
k_{35}	IFNRJ dimer dissociation rate	0.2	1/s
$[IFN\text{-}beta]_0$	Steady state IFN-beta concentration in ECM after infection	0.1	nM
$[cIRF\text{-}9]_0$	Steady state IRF-9 concentration in cytosol before infection	50	nM
$[IFNAR]_0$	Initial value of IFNGR receptor	12	nM
$[JAK]_0$	Initial value of JAK	12	nM
$[STATc]_0$	Initial value of STAT1c	1000	nM
$[SHP\text{-}2]_0$	Steady state SHP-2 concentration	100	nM
$[PPX]_0$	Steady state PPX concentration	50	nM
$[PPN]_0$	Steady state PPN concentration	60	nM

Figure 6-12b – Parameters for IFN receptor signaling pathway.

IFN Amplification Cycle Model

The IFN amplification cycle is a critical step in the response to protect the cell from virus infection. It is depicted in Figure 6-13 below. A team of scientists from the America created a dynamic model of this pathway. The article was published in the *Journal of Theoretical Biology* (Hancioglu, 2007). They programmed the pathway in XPAUT which can be saved in SBML.

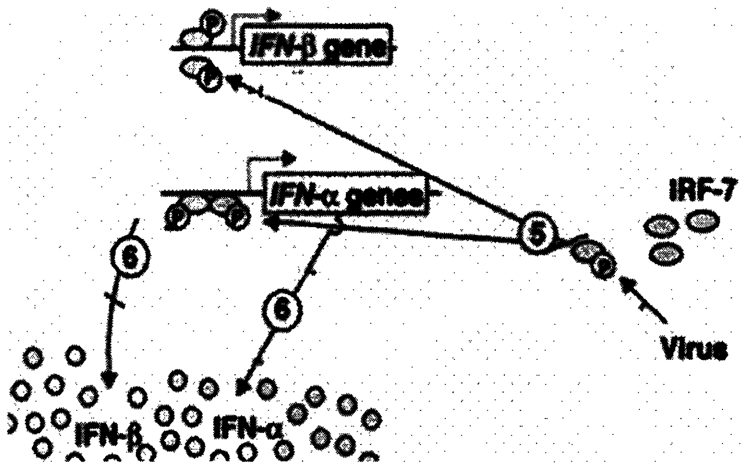


Figure 6-13 – IFN amplification cycle pathway (Hancioglu, 2007).

A detailed diagram of the this pathway is shown in Figure 6-114 which includes all the species and molecular interactions. In Figure 6-14, virus interaction with IRF-7 not only serves to up-regulate IFN-beta but also serves to up-regulate IFN-alpha.

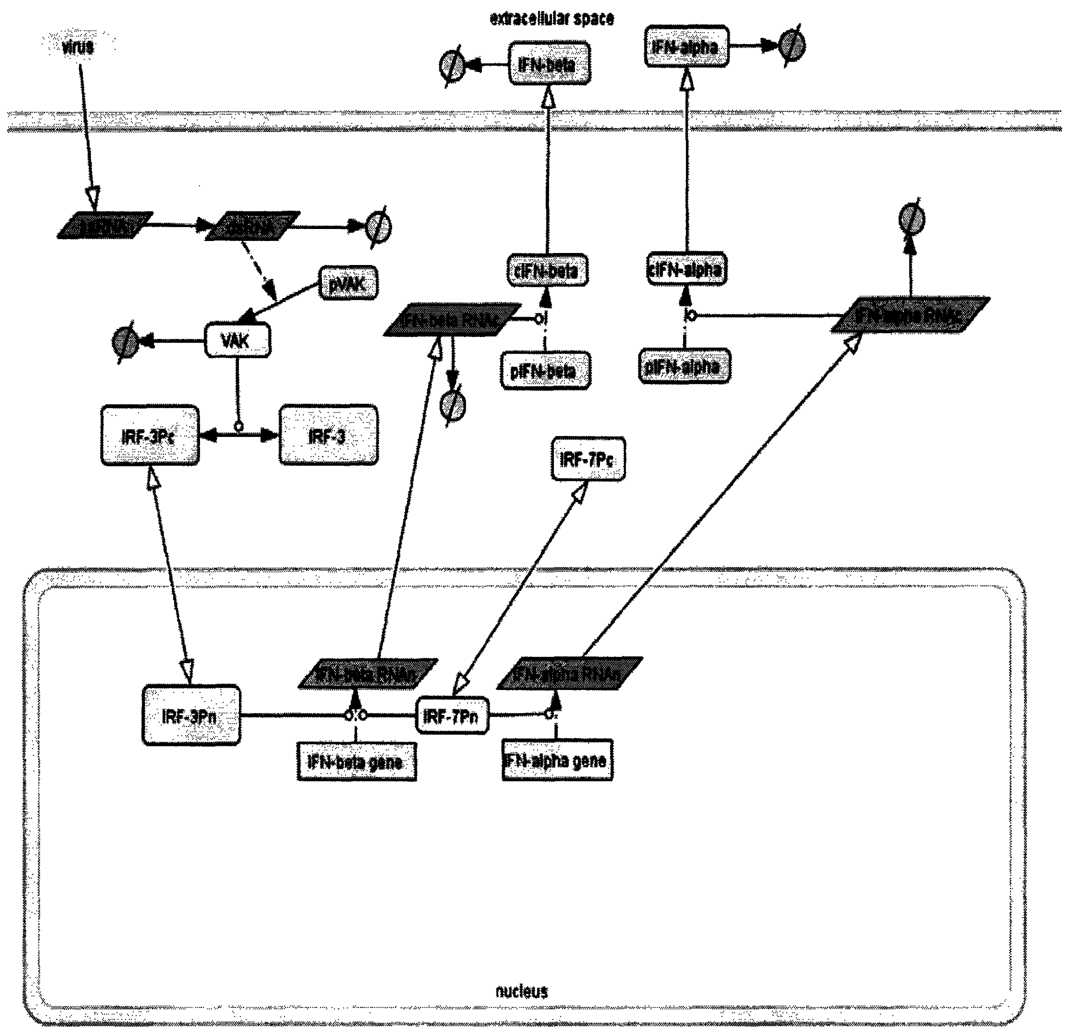


Figure 6-14 – IFN amplification cycle pathway detailed mechanics.

In Figure 6-15a and 6-15b are listed the differential equations defining the molecular interactions for this pathway.

Reactions	Math
[virus]→[ssRNA]	$\frac{dRxn}{dt} = k_7 \cdot [virus]$
[ssRNA] →[dsRNA]	$\frac{dRxn}{dt} = k_1 \cdot [ssRNA]$
[dsRNA] degradation	$\frac{dRxn}{dt} = k_2 \cdot [dsRNA]$
[VAK] activation	$\frac{dRxn}{dt} = k_3 \cdot [dsRNA]$
[VAK] degradation	$\frac{dRxn}{dt} = k_4 \cdot [VAK]$
[IRF-3]⟷[IRF-3Pc]	$\frac{dRxn}{dt} = \frac{k_{f6} \cdot [IRF-3] \cdot [VAK]}{[IRF-3] + \frac{k_{r5} + k_{f6}}{k_{f5}}} - k_{r6} \cdot [IRF-3Pc]$
[IRF-3Pc]⟷[IRF-3Pn]	$\frac{dRxn}{dt} = k_{f7} \cdot [IRF-3Pc] - k_{r7} \cdot [IRF-3Pn]$
[IFN-beta RNAn] production	$\frac{dRxn}{dt} = \frac{k_{8a} \cdot [IRF-3Pn]}{[IRF-3Pn] + k_{8b}} + \frac{k_{8a} \cdot [IRF-7Pn]}{[IRF-7Pn] + k_{8b}}$
[IRF-beta RNAn]⟷[IFN-beta RNAc]	$\frac{dRxn}{dt} = k_9 \cdot [IFN-beta\_RNAn]$
[IFN-beta RNAc] degradation	$\frac{dRxn}{dt} = k_{10} \cdot [IFN-beta\_RNAc]$
[cIFN-beta] production	$\frac{\partial Rxn}{\partial t} = k_{11} \cdot [IFN-beta\_RNAc]$
[cIFN-beta] →[IFN-beta]	$\frac{dRxn}{dt} = k_{12} \cdot [cIFN-beta]$
[IFN-beta] degradation	$\frac{dRxn}{dt} = k_{13} \cdot [IFN-beta]$
[IFN-alpha RNAn] production	$\frac{dRxn}{dt} = \frac{k_{8a} \cdot [IRF-7Pn]}{[IRF-7Pn] + k_{8b}}$

Figure 6-15 – Differential equations for IFN amplification cycle pathway.

[IFN-alpha RNAn]⟷[IFN-alpha RNAc]	$\frac{dRxn}{dt} = k_9 \cdot [IFN-alpha\_RNAn]$
[IFN-alpha RNAc] degradation	$\frac{dRxn}{dt} = k_{10} \cdot [IFN-alpha\_RNAc]$
[cIFN-alpha] production	$\frac{dRxn}{dt} = k_{11} \cdot [IFN-alpha\_RNAc]$
[cIFN-alpha] →[IFN-alpha]	$\frac{\partial Rxn}{\partial t} = k_{12} \cdot [cIFN-alpha]$
[IFN-alpha] degradation	$\frac{dRxn}{dt} = k_{13} \cdot [IFN-alpha]$
[IRF-7Pc]⟷[IRF-7Pn]	$\frac{dRxn}{dt} = k_{f7} \cdot [IRF-7Pc] - k_{r7} \cdot [IRF7-Pn]$

Figure 6-15b – Differential equations for IFN amplification cycle pathway.

In Figure 6-16 are listed the internal parameters, rate constants and initial conditions for this pathway.

Pathway	Symbol	Description	Value	Unit
phase_two_part_two	k_v	Virus infection rate	0.01	1/s
phase_two_part_two	k_1	Double strand RNA formation rate	1	1/(nM*s)
phase_two_part_two	k_2	Double strand RNA degradation rate	0.0001	1/s
phase_two_part_two	k_3	VAK activation rate	0.0001	1/s
phase_two_part_two	k_4	VAK protein degradation rate	0.0005	1/s
phase_two_part_two	k_5	IRF-3 & VAK association rate	0.008	1/(nM*s)
phase_two_part_two	k_6	IRF-3–VAK complex dissociation rate	0.8	1/s
phase_two_part_two	k_{6a}	IRF-3 phosphorylation rate	0.4	1/s
phase_two_part_two	k_{6d}	IRF-3 dephosphorylation rate	0.005	1/s
phase_two_part_two	k_{f7}	Rate of IRF-3P transport to nucleus	0.005	1/s
phase_two_part_two	k_{r7}	Rate of IRF-3P transport to cytosol	0.05	1/s
phase_two_part_two	k_{8c}	Rate constant (chemistry) of RNA formation	0.01	nM/s
phase_two_part_two	k_{8a}	Rate constant (association) of RNA formation	400	nM
phase_two_part_two	k_9	Rate of mRNA transport to cytosol	0.001	1/s
phase_two_part_two	k_{10}	Rate of mRNAc degradation	0.0005	1/s
phase_two_part_two	k_{11}	Rate of IFN production	0.01	1/s
phase_two_part_two	k_{12}	Rate of IFN transport to extracellular space	0.001	1/s
phase_two_part_two	k_{13}	Rate of IFN degradation	0.0001	1/s
phase_two_part_two	$[virus]_0$	Initial extracellular virus concentration	10	nM
phase_two_part_two	$[IRF\text{-}3]_0$	Steady state IRF-3 concentration before infection	10	nM
phase_two_part_two	$[IRF\text{-}7Pc]_0$	Steady state IRF-7 concentration in cytosol after infection	0.01	nM

Figure 6-16 – Parameters for IFN amplification cycle pathway.

SOCS1 Regulation Model

The high level biological pathway of virus infection is depicted in Figure 6-17 below. This biological pathway produces SOCS1 to regulate and balance the production of IFNs. It is depicted in Figure 6-7 below. Without this pathway, the additional levels of IFNs, beyond what is necessary to stop the virus infection, can itself have detrimental effects on the cell. . Scientists in Japan in 2001 defined this pathway model in *Genome Informatics* (Yamada, 2001). They programmed the pathway in MATLAB.

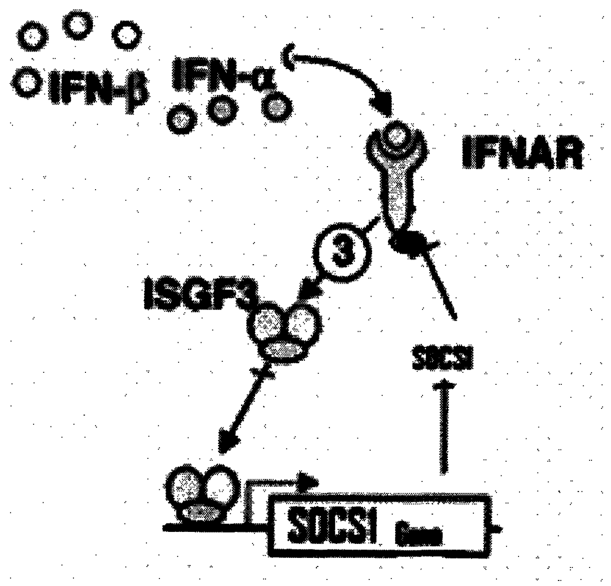


Figure 6-17 – SOCS1 regulation pathway(Yamada, 2001).

A detailed diagram of this pathway is shown in Figure 6-18 which includes all the species and molecular interactions. Here, JAK binds to the IFN receptor and forms the JAK-IFNR receptor complex. Once IFN binds to the receptor, the resulting complex associate with each other and forms a homo-dimer. This dimer undergoes phosphorylation, leading to a form as IFNRJ2*, which catalyzes the phosphorylation of STAT1. The phosphorylated STAT1 also forms a homo-dimer and acts as a transcription factor of SOCS1 gene. The resulting protein, SOCS1, inhibits the kinase activity of IFNRJ2 and is the key component of the negative feedback loop.

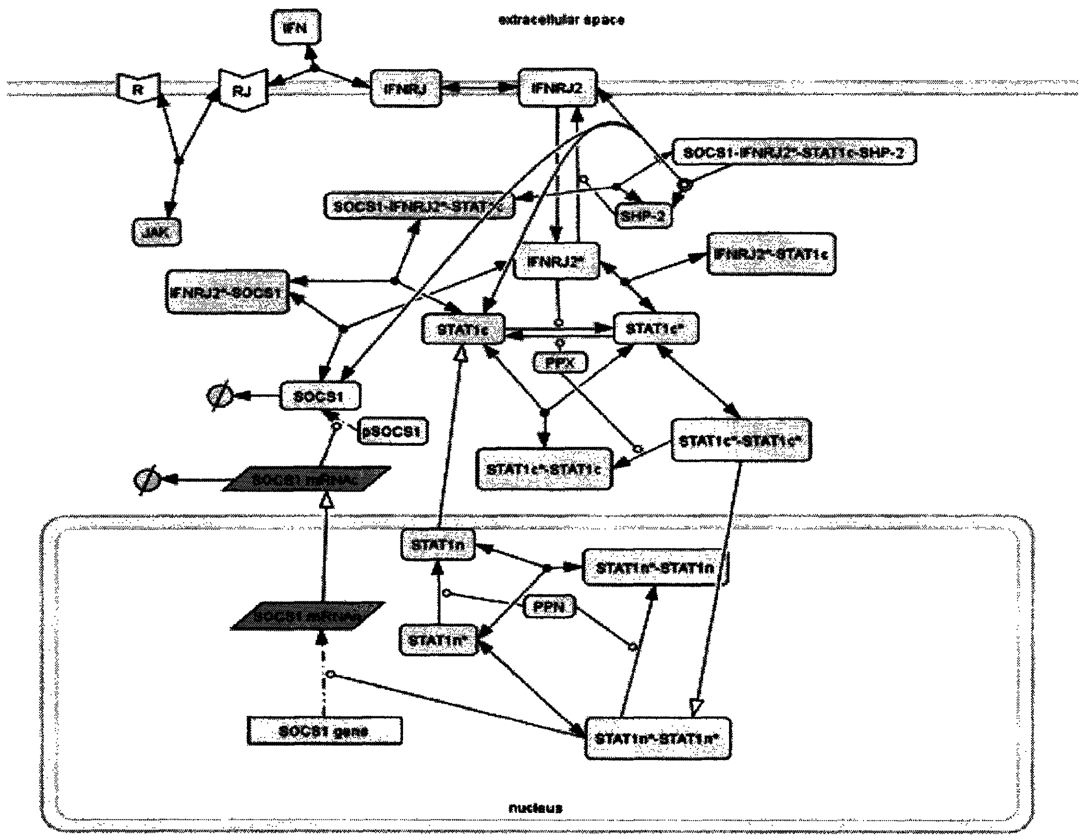
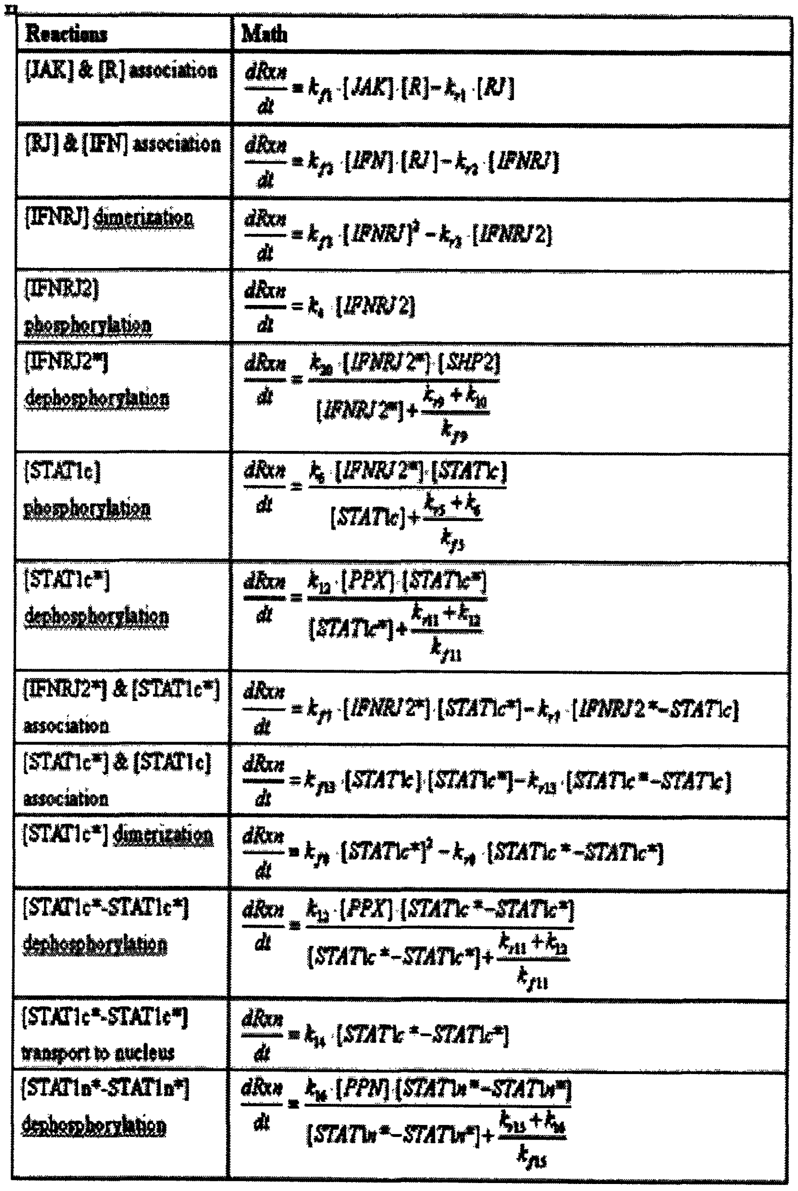


Figure 6-18 – SOCS1 regulation pathway detailed mechanics.

In Figure 6-19a and 6-19b are listed the differential equations defining the molecular interactions for this pathway.



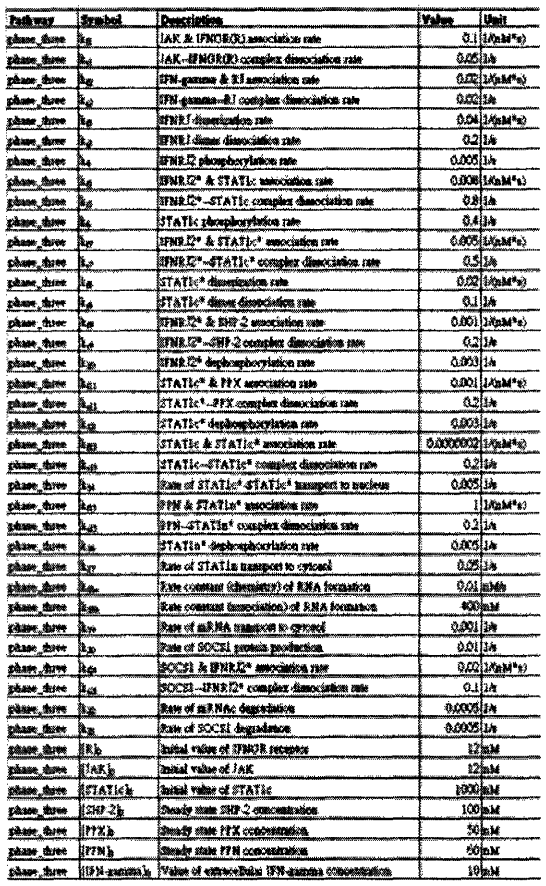
Reactions	Math
[JAK] & [R] association	$\frac{dRxn}{dt} = k_{f1} \cdot [JAK] \cdot [R] - k_{r1} \cdot [RJ]$
[RJ] & [IFN] association	$\frac{dRxn}{dt} = k_{f2} \cdot [IFN] \cdot [RJ] - k_{r2} \cdot [IFNRJ]$
[IFNRJ] dimerization	$\frac{dRxn}{dt} = k_{f3} \cdot [IFNRJ]^2 - k_{r3} \cdot [IFNRJ2]$
[IFNRJ2] phosphorylation	$\frac{dRxn}{dt} = k_4 \cdot [IFNRJ2]$
[IFNRJ2*] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{10} \cdot [IFNRJ2^*] \cdot [SHP2]}{[IFNRJ2^*] + \frac{k_{r9} + k_{10}}{k_{f9}}}$
[STAT1c] phosphorylation	$\frac{dRxn}{dt} = \frac{k_6 \cdot [IFNRJ2^*] \cdot [STAT1c]}{[STAT1c] + \frac{k_{r5} + k_6}{k_{f5}}}$
[STAT1c*] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{12} \cdot [PPX] \cdot [STAT1c^*]}{[STAT1c^*] + \frac{k_{r11} + k_{12}}{k_{f11}}}$
[IFNRJ2*] & [STAT1c*] association	$\frac{dRxn}{dt} = k_{f7} \cdot [IFNRJ2^*] \cdot [STAT1c^*] - k_{r7} \cdot [IFNRJ2^* - STAT1c]$
[STAT1c*] & [STAT1c] association	$\frac{dRxn}{dt} = k_{f13} \cdot [STAT1c] \cdot [STAT1c^*] - k_{r13} \cdot [STAT1c^* - STAT1c]$
[STAT1c*] dimerization	$\frac{dRxn}{dt} = k_{f8} \cdot [STAT1c^*]^2 - k_{r8} \cdot [STAT1c^* - STAT1c^*]$
[STAT1c*-STAT1c*] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{12} \cdot [PPX] \cdot [STAT1c^* - STAT1c^*]}{[STAT1c^* - STAT1c^*] + \frac{k_{r11} + k_{12}}{k_{f11}}}$
[STAT1c*-STAT1c*] transport to nucleus	$\frac{dRxn}{dt} = k_{14} \cdot [STAT1c^* - STAT1c^*]$
[STAT1n*-STAT1n*] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{16} \cdot [PPN] \cdot [STAT1n^* - STAT1n^*]}{[STAT1n^* - STAT1n^*] + \frac{k_{r15} + k_{16}}{k_{f15}}}$

Figure 6-19a – Differential equations for SOCS1 regulation pathway.

[STAT1n*-STAT1n*] dissociation	$\frac{dRxn}{dt} = k_{r8} \cdot [STAT1n^* - STAT1n^*] - k_{f8} \cdot [STAT1n^*]^2$
[STAT1n*] & [STAT1n] association	$\frac{dRxn}{dt} = k_{f13} \cdot [STAT1n^*] \cdot [STAT1n] - k_{r13} \cdot [STAT1n^* - STAT1n]$
[STAT1n*] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{16} \cdot [PPN] \cdot [STAT1n^*]}{[STAT1n^*] + \frac{k_{r15} + k_{16}}{k_{f15}}}$
[STAT1n] transport to cytosol	$\frac{dRxn}{dt} = k_{17} \cdot [STAT1n]$
[SOCS1 mRNAn] production	$\frac{dRxn}{dt} = \frac{k_{18a} \cdot [SOCS1\_mRNAn]}{[SOCS1\_mRNAn] + k_{18b}}$
[SOCS1 mRNAn] export to cytosol	$\frac{dRxn}{dt} = k_{19} \cdot [SOCS1\_mRNAn]$
[SOCS1 mRNAc] degradation	$\frac{dRxn}{dt} = k_{22} \cdot [SOCS1\_mRNAc]$
[SOCS1] protein production	$\frac{dRxn}{dt} = k_{20} \cdot [SOCS1\_mRNAc]$
[SOCS1] degradation	$\frac{dRxn}{dt} = k_{23} \cdot [SOCS1]$
[SOCS1] & [IFNRJ2*] association	$\frac{dRxn}{dt} = k_{f21} \cdot [IFNRJ2^*] \cdot [SOCS1] - k_{r21} \cdot [IFNRJ2^* - SOCS1]$
[STAT1c] & [IFNRJ2*-SOCS1] association	$\frac{dRxn}{dt} = k_{f3} \cdot [IFNRJ2^* - SOCS1][STAT1c]$ $- k_{r3} \cdot [SOCS1 - IFNRJ2^* - STAT1c]$
[SOCS1-IFNRJ2*-STAT 1c] & [SHP2] association	$\frac{dRxn}{dt} = k_{f9} \cdot [SOCS1 - IFNRJ2^* - STAT1c][SHP2]$ $- k_{r9} \cdot [SOCS1 - IFNRJ2^* - STAT1c - SHP2]$
[SOCS1-IFNRJ2*-STAT 1c-SHP2] dissociation	$\frac{dRxn}{dt} = k_{30} \cdot [SOCS1 - IFNRJ2^* - STAT1c - SHP2]$

Figure 6-19b – Differential equations for SOCS1 regulation pathway.

In Figure 6-20 are listed the internal parameters, rate constants and initial conditions for this pathway.



Pathway	Symbol	Description	Value	Unit
phase_three	[illegible]	JAK & IFNGR(R) association rate	0.1	1/(nM*s)
phase_three	[illegible]	JAK–IFNGR(R) complex dissociation rate	0.05	1/s
phase_three	[illegible]	IFN-gamma & RJ association rate	0.02	1/(nM*s)
phase_three	[illegible]	IFN-gamma–RJ complex dissociation rate	0.02	1/s
phase_three	[illegible]	IFNRJ dimerization rate	0.04	1/(nM*s)
phase_three	[illegible]	IFNRJ dimer dissociation rate	0.2	1/s
phase_three	[illegible]	IFNRJ2 phosphorylation rate	0.005	1/s
phase_three	[illegible]	IFNRJ2* & STAT1c association rate	0.008	1/(nM*s)
phase_three	[illegible]	IFNRJ2*–STAT1c complex dissociation rate	0.8	1/s
phase_three	[illegible]	STAT1c phosphorylation rate	0.4	1/s
phase_three	[illegible]	IFNRJ2* & STAT1c* association rate	0.005	1/(nM*s)
phase_three	[illegible]	IFNRJ2*–STAT1c* complex dissociation rate	0.5	1/s
phase_three	[illegible]	STAT1c* dimerization rate	0.02	1/(nM*s)
phase_three	[illegible]	STAT1c* dimer dissociation rate	0.1	1/s
phase_three	[illegible]	IFNRJ2* & SHP-2 association rate	0.001	1/(nM*s)
phase_three	[illegible]	IFNRJ2*–SHP-2 complex dissociation rate	0.2	1/s
phase_three	[illegible]	IFNRJ2* dephosphorylation rate	0.003	1/s
phase_three	[illegible]	STAT1c* & PPX association rate	0.001	1/(nM*s)
phase_three	[illegible]	STAT1c*–PPX complex dissociation rate	0.2	1/s
phase_three	[illegible]	STAT1c* dephosphorylation rate	0.003	1/s
phase_three	[illegible]	STAT1c & STAT1c* association rate	0.000002	1/(nM*s)
phase_three	[illegible]	STAT1c–STAT1c* complex dissociation rate	0.2	1/s
phase_three	[illegible]	Rate of STAT1c*–STAT1c* transport to nucleus	0.005	1/s
phase_three	[illegible]	PPN & STAT1n* association rate	1	1/(nM*s)
phase_three	[illegible]	PPN–STAT1n* complex dissociation rate	0.2	1/s
phase_three	[illegible]	STAT1n* dephosphorylation rate	0.005	1/s
phase_three	[illegible]	Rate of STAT1n transport to cytosol	0.05	1/s
phase_three	[illegible]	Rate constant (chemistry) of RNA formation	0.01	nM/s
phase_three	[illegible]	Rate constant (association) of RNA formation	400	nM
phase_three	[illegible]	Rate of mRNA transport to cytosol	0.001	1/s
phase_three	[illegible]	Rate of SOCS1 protein production	0.01	1/s
phase_three	[illegible]	SOCS1 & IFNRJ2* association rate	0.02	1/(nM*s)
phase_three	[illegible]	SOCS1–IFNRJ2* complex dissociation rate	0.1	1/s
phase_three	[illegible]	Rate of mRNAc degradation	0.0005	1/s
phase_three	[illegible]	Rate of SOCS1 degradation	0.0005	1/s
phase_three	$[R]_0$	Initial value of IFNGR receptor	12	nM
phase_three	$[JAK]_0$	Initial value of JAK	12	nM
phase_three	$[STAT1c]_0$	Initial value of STAT1c	1000	nM
phase_three	$[SHP\text{-}2]_0$	Steady state SHP-2 concentration	100	nM
phase_three	$[PPX]_0$	Steady state PPX concentration	50	nM
phase_three	$[PPN]_0$	Steady state PPN concentration	60	nM
phase_three	$[IFN\text{-}gamma]_0$	Value of extracellular IFN-gamma concentration	10	nM

Figure 6-20 - Parameters for SOCS1 regulation pathway.

6.4 Individual Model Solutions

In this section for each of the pathway models in the previous section, we solve each pathway model using Cytosolve. For each solution, a single graph is presented which provides the time-dependent variations in species concentration of the relevant species.

Virus Infection Model Solution

This system assumes the infection of a cell by one virus does not inhibit further infection by other viruses. It is also assumed that the only source of virus is from initial application of virus.

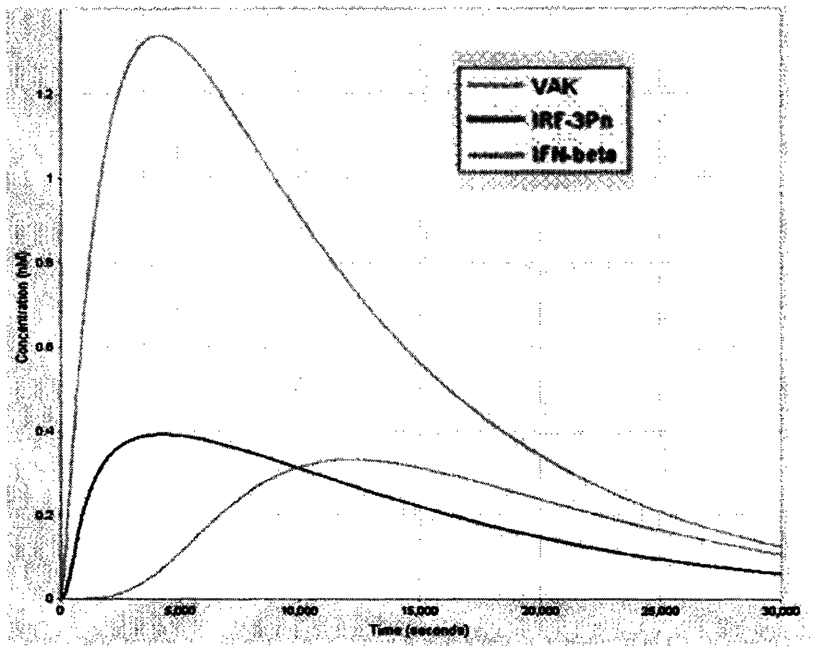


Figure 6-21 – Virus infection model solution.

Initially, as shown in Figure 6-21, the concentration of VAK and IRF-3 both increase. The IRF-3 then activates the expression of IFN-beta. Due to the limiting amount of virus and absence of a positive feedback system, the concentration of IFN-beta reaches its maximum value of 0.33 nM in the extra cellular space at around 3.6 hours and gradually declines.

IFN Receptor Signaling Model Solution

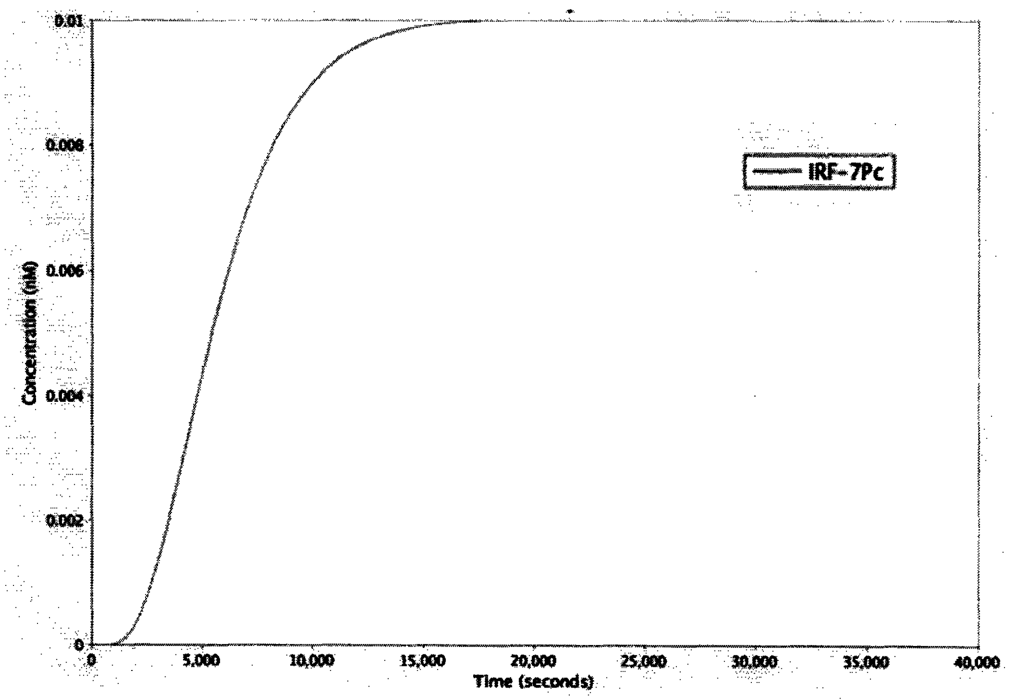


Figure 6-22 – IFN receptor signal model solution.

In Figure 6-22, the concentration of IRF-7 through time is a sigmoidal curve which reaches near the steady state value of 0.01 nM at around 4 hours. Fast activation of IRF-7 is required to activate the positive feedback system.

IFN Amplification Cycle Model Solution

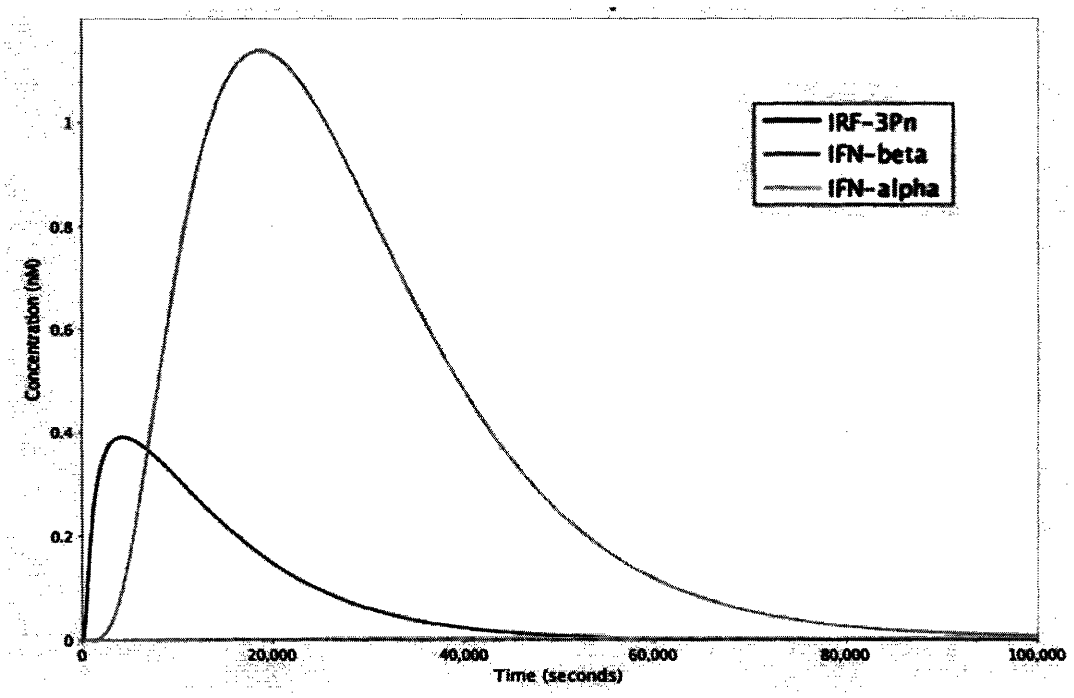


Figure 6-23 – IFN amplification cycle model solution.

In Figure 6-23, most of the IRF-7 is used to produce IFN-Beta. Very little IRF-7 exists for significant production of IFN-Alpha.

SOCS1 Regulation Model Solution

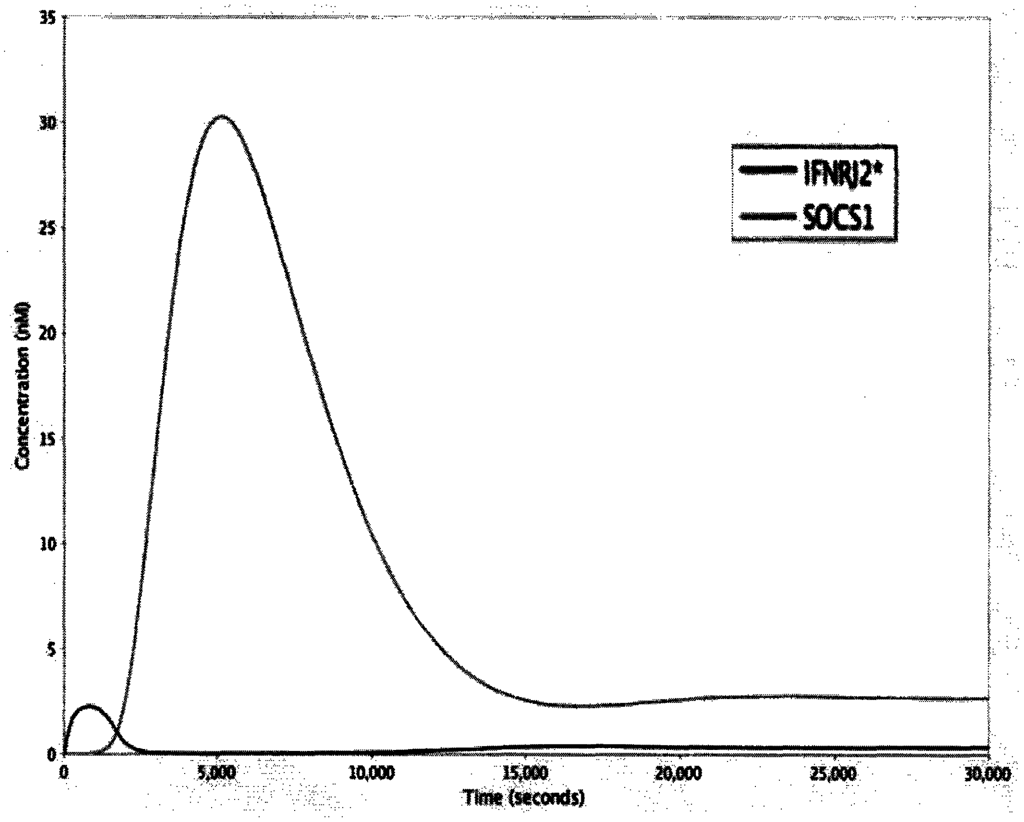


Figure 6-24 - SOCS1 regulation model solution.

In Figure 6-24, it is assumed the extra-cellular concentration of IFN is constant. The concentration of IFNR2* reaches its maximum value of 2 nM several minutes after starting the simulation, and gradually declines due to inhibition by SOCS1. The concentration of SOCS1 has a maximum value of 30 nM at 1.5 hour. After the peak, SOCS1 decreases to reach a steady state value of 2.5 nM because of the negative feedback loop.

6.5 Integrated Model of IFN Response

The high level diagram of the integrated model is shown in Figure 6-25. This integrated model illustrates the integration of the four biological pathway models involved in the IFN response.

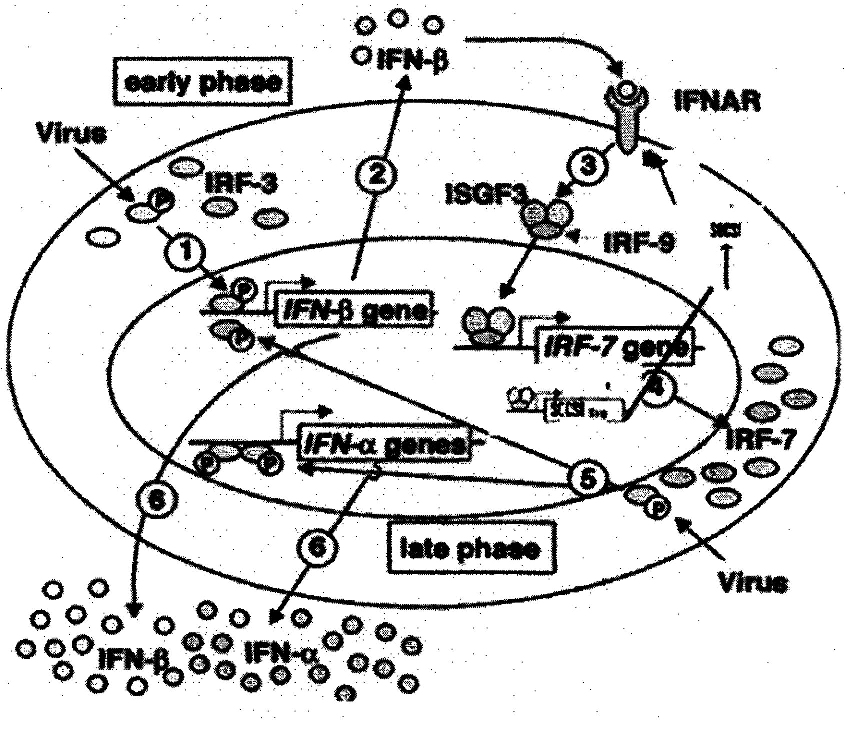


Figure 6-25 - Integrated model of the IFN response to virus infection.

A detailed diagram of this pathway is shown in Figure 6-26 which includes all the species and molecular interactions of the integrated model.

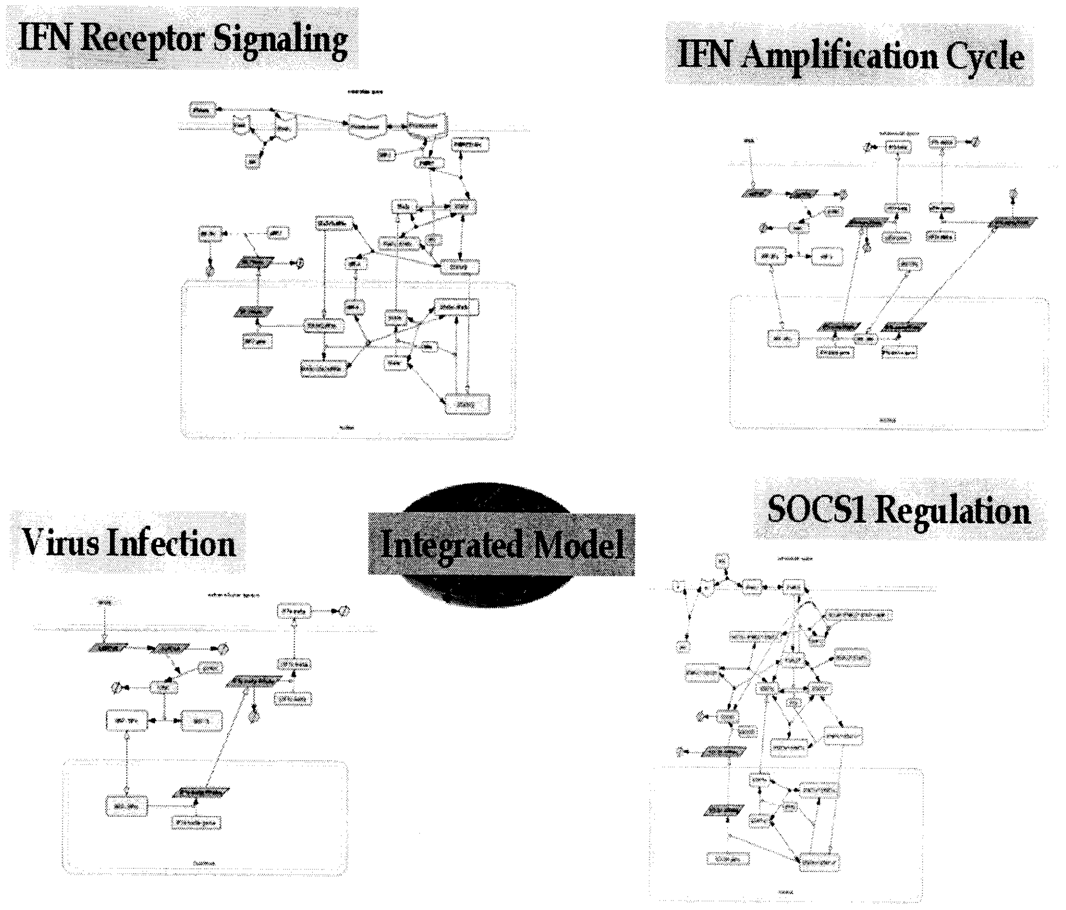


Figure 6-26 – Detailed integrated model of the IFN response to virus infection.

This integrated model combines the four models of the interferon pathway: virus infection leads to up regulation of interferon beta; massive production of IFN-alpha and IFN-beta with a positive feedback system; and, then negative-feedback control of JAK/STAT signal transduction pathway by SOCS1.

Integrated Model Solution

<u>Cytosolve Approach</u>

In the first case, we integrate the four IFN models using Cytosolve as shown in Figure 6-27.

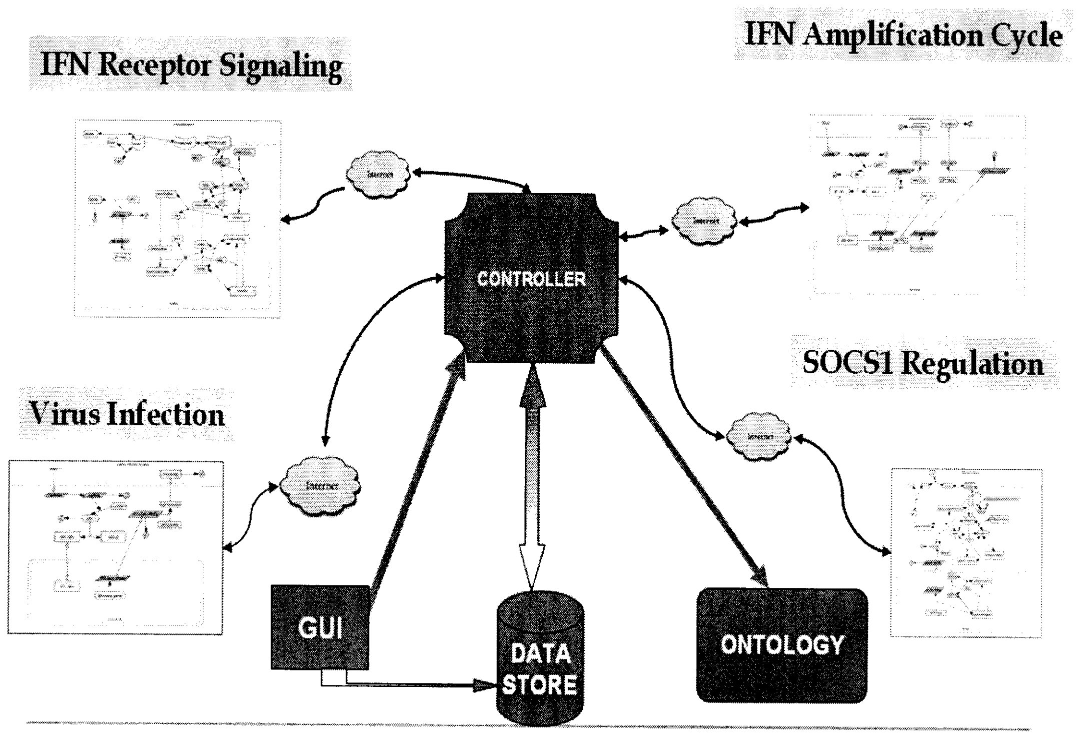


Figure 6-27 – Cytosolve approach to integrating the four models.

Figure 6-28 contains the solution from the integration of all four biological pathway models.

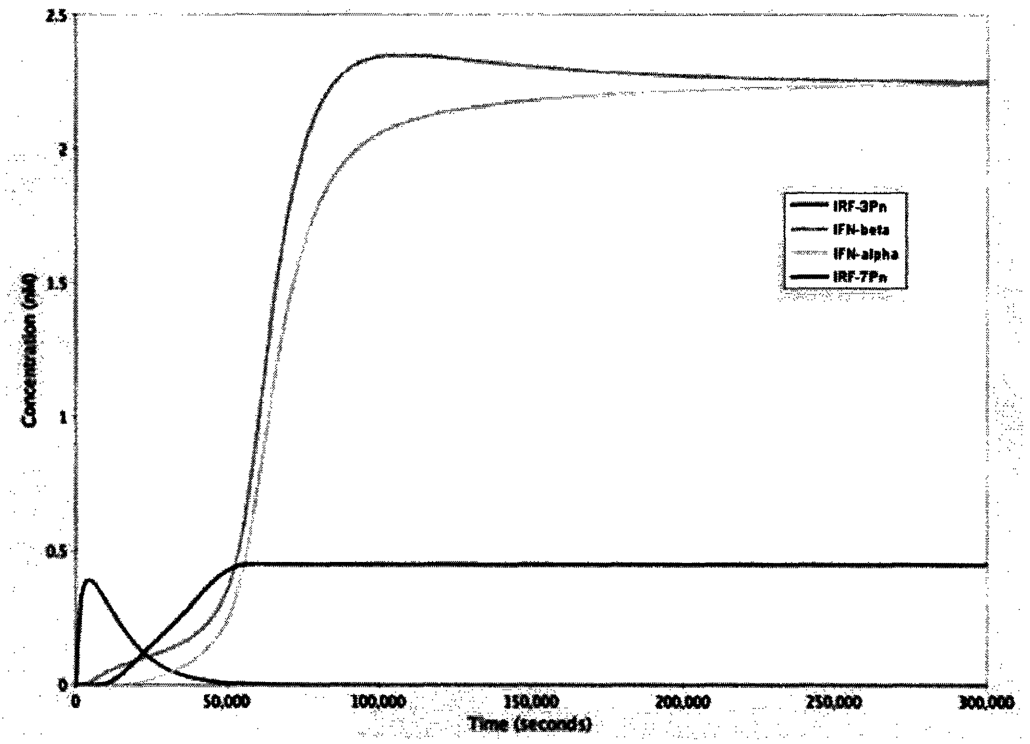


Figure 6-28 - Integrated model solution.

The key molecular species presented in this figure are IRF-3, IRF-7, IFN_Beta and IFN-Alpha. This integrated model combines the four pieces of interferon pathway: virus infection leads to up regulation of IFN-Beta; IFN-Beta then results in the creation of IRF-7; the existence of IRF-7 then results in positive feedback to which increases a massive production of IFN-Alpha and IFN-Beta; finally, control of JAK/STAT signal transduction pathway by SOCS1 results in regulating and balancing the production of IFN-Alpha and IFN-Beta. Close review of Figure 6-28 reveals several important elements of the integrated model.

First, during the first ~13 hours (~50,000 seconds), the concentration of IRF-7 through time is a sigmoidal curve which reaches the steady state value of 0.7 nM .

Second, during this same first ~13 hour period, the concentration of IFN-Beta and IFN-Alpha slowly increases. What is interesting to note is that, in Figure 6-29 (below) which offers a zoomed in version of the first 3.3 hours (~12,000 seconds), the initial production of IFN-Beta is then followed by the production of IFN-Alpha. IFN-Beta is produced within the first 30-40 minutes (~2000 seconds to ~2500 seconds). The initial production of IFN-Beta after the 40 minute period and before the 3.3 hour period is defined by a marked increase in IFN-Beta production.

Third, in Figure 6-28, above, after the first ~13 hours (~50,000 seconds) to ~25 hours (~90,000 seconds), IFN-Beta and IFN-Alpha exponentially increases.

Fourth, in Figure 6-28, above, after ~25 hours (~90,000 seconds), IFN-Beta and IFN-Alpha concentrations reach their maximum and gradually approach steady state due to the balance between positive feedback system and negative feedback control from SOCS1 activation.

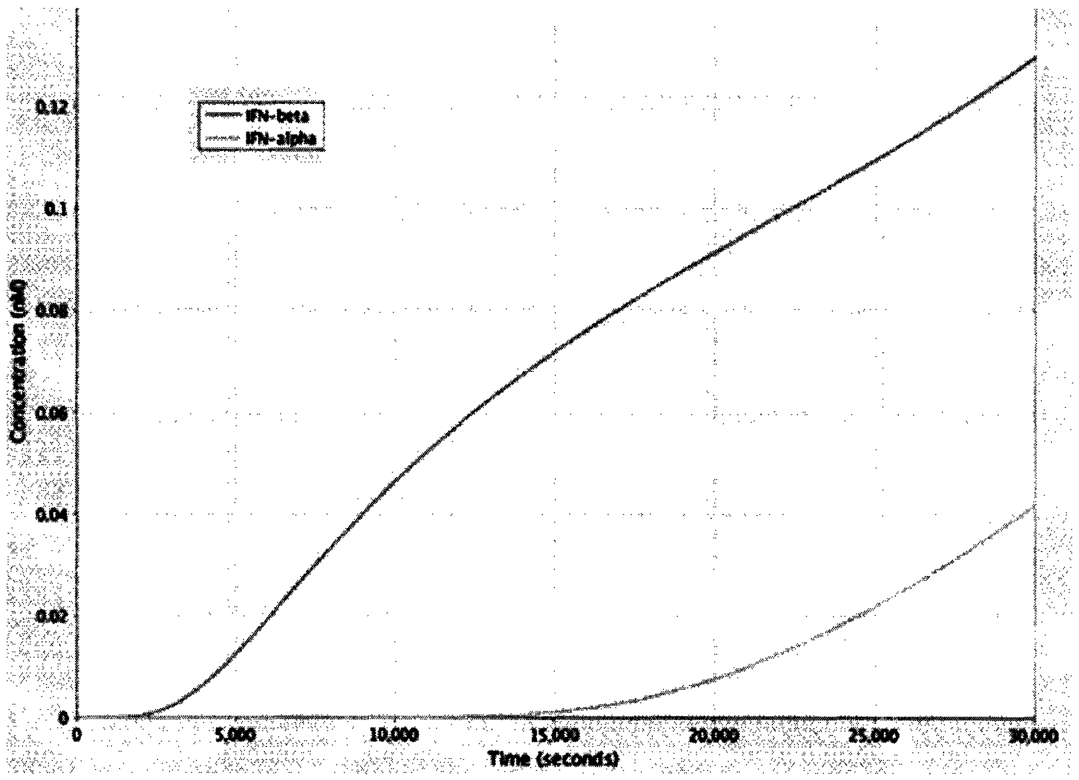


Figure 6-29 – A detailed, zoomed in, view of IFN-Beta and IFN-Alpha production.

Monolithic Approach

In the second case, we integrate the four IFN models using Cell Designer by merging all the models as shown in Figure 6-30.

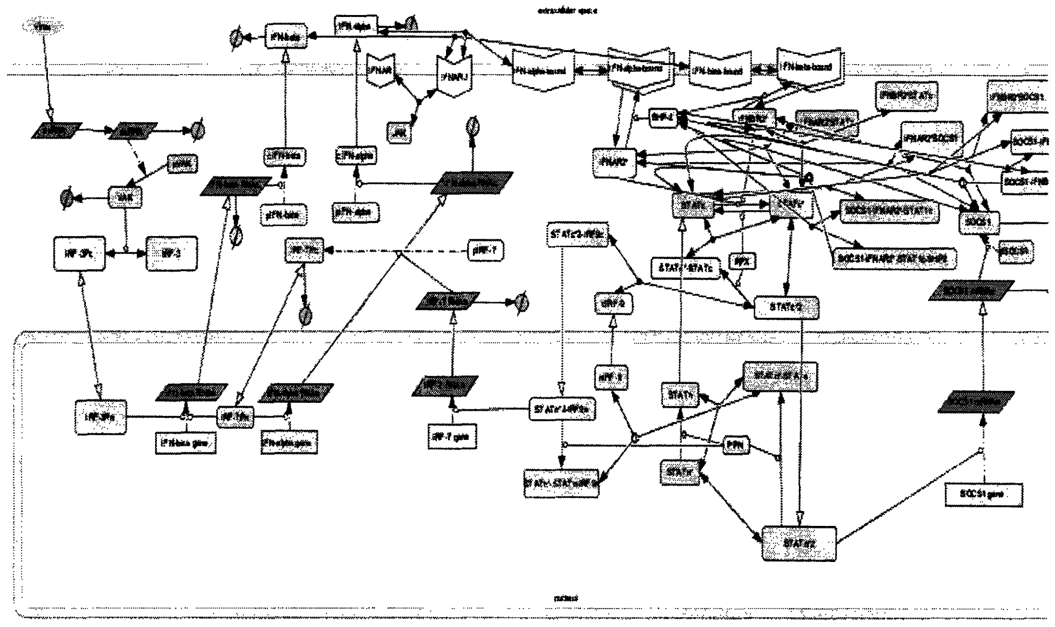


Figure 6-30 - Monolithic approach (Cell Designer) to integrating the four models.

The integration of the above models yields the same results as in Figure 6-28. The mathematical equations and internal parameters for this approach are included in Appendix C.

6.6 Re-Integration and Maintenance Test

In earlier chapters, we discussed that the central aspect Cytosolve's architecture is its ability to scale. Given the nature of biology, which is an experimental science, where new protein structures and new protein-protein interactions are discovered daily, it is clear that any one biological pathway model in an integrated model will change. Such changes mean that the integrated model itself will have to be rebuilt or re-integrated constantly. Thus, as an integrated model grows, the maintenance of such a model may become onerous to maintain, resulting in a lack of scalability for further growing and maintaining such a model. In this section, we measure Cytosolve's ability to maintain an integrated model.

Methodology

To demonstrate Cytosolve's ability to easily update changes to an integrated model, we consider the SOCS1 regulation pathway model previously mentioned. In this case, we consider a different variation of this model, denoted as the *SOCS1'* model, consisting of the same species as shown in Figure 6-18, but, in this version of the model, we have changed the kinds of molecular interactions as well as the internal parameters. Such a scenario is highly likely given the nature of biology. Using the SOCS1' model, we measure the time and effort involved to reintegrate or "swap out" the SOCS1 model from Figure 6-26, with the new SOCS1′ model. We contrast the effort of doing this reintegration first using the Cytosolve approach and then with the monolithic approach.

Cytosolve Approach

Reintegration of SOCS1′ with the SOCS1 model using Cytosolve took less than 2 hours to perform. The effort involved the following steps:

1. Loading the SOCS1′ model on a server
2. Updating the PID file
3. Running the Controller

Monolithic Approach

Reintegration of SOCS1′ with the SOCS1 model using the monolithic approach took approximately 4 days to perform. The following efforts were involved in this reintegration effort:

1. The rate constants needed to be reorganized to avoid confusion.
2. While doing copy and paste of different reactions form SOCS1′ to replace reactions in the SOCS1 model, the mathematics of the reactions did not automatically change with the updated species identifiers in Cell Designer. This problem required the identifiers to be reset manually to make sure the right reactants go to the right product.

3. It became very tedious to link the actual arrows with the new SOCS1′ model and errors were very easy to make.
4. Deleting the changes in the SOCS1 model with the new SOCS1′ model is a very tedious, manual and menial process, in short a boring process.
5. Many mistakes were made in the reintegration effort required repeated rework.

Discussion

It is clear from the above results, Cytosolve offers a much more scalable approach to maintain and update an existing integrated model. Cell Designer took far more effort than Cytosolve (e.g. 2 hours versus 4 days). While a more rigorous analysis can be performed, the effort of reintegrating one of the four models into the IFN integrated model serves to demonstrate the many challenges in using a monolithic approach.

6.7 Analysis of IFN Integrated Model

Cytosolve has been used to develop an integrated model of the IFN response to virus infection. One of the goals of building larger models from integrating smaller models is to reveal new understanding of biological phenomena not possible through experimentation. In this section, we perform various numerical experiments to reveal such understanding.

Decreasing the SOCS1 Degradation Rate

The degradation rate of SOCS1 is set from the original value 5.0E-4 to zero with an interval 1.0E-4. As the degradation rate is lowered, more SOCS1 is present in the system. The presence of more SOCS1 means less IFN. This modification is valid based on the assumption that the production rate of SOCS1 highly exceeds its degradation rate, the few days after the negative feedback control is turned on and increases.

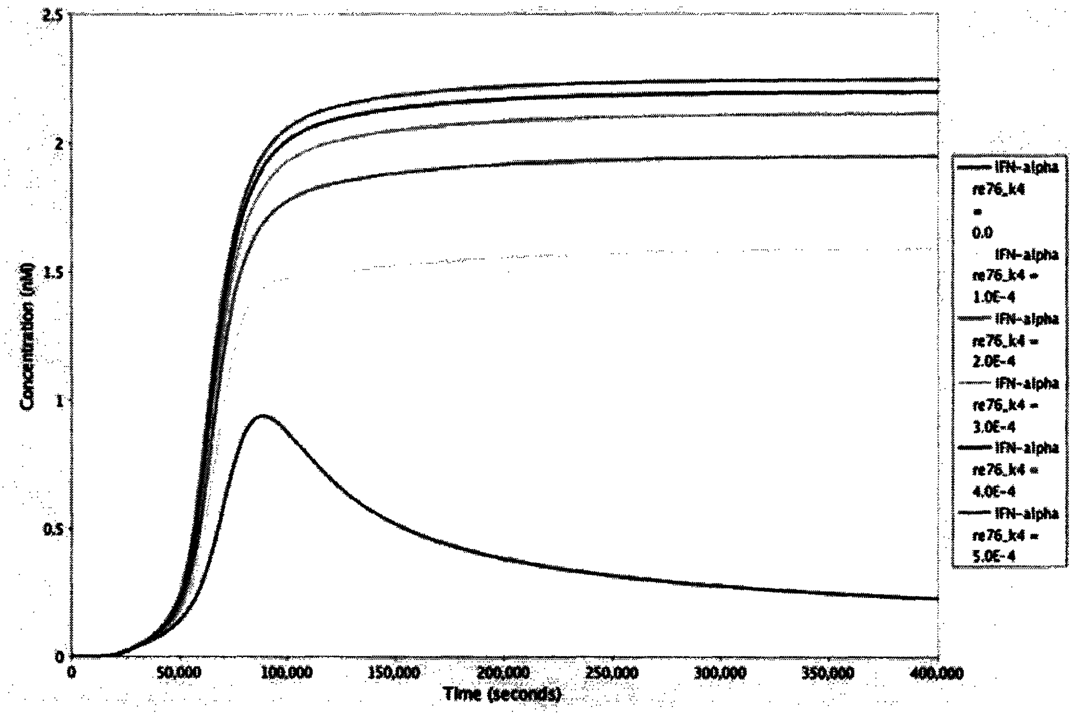


Figure 6-31 – IFN-Alpha lowers as the degradation rate of SOCS1 is increased (more SOCS1 is produced to suppress production of IFN-Alpha).

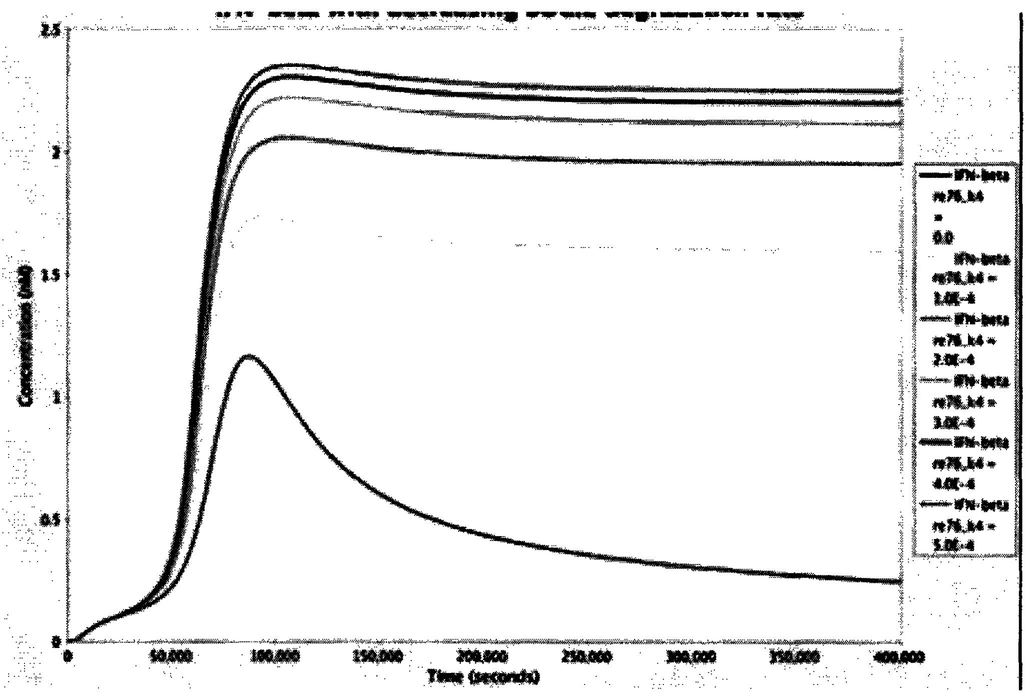


Figure 6-32 – IFN-Beta lowers as the degradation rate of SOCS1 is increased (more SOCS1 is produced to suppress production of IFN-Alpha).

Thus, by lowering the degradation rate, we are simulating this phenomena. In Figure 6-31 and 6-32, we present the graphs for both IFN-Alpha and IFN-Beta, respectively. Therefore, the concentration of SOCS1 can build up through time. The simulation shows that the maximum level of interferon, both IFN-Alpha and IFN-Beta, is lowered. In the extreme case that the SOCS1 degradation rate is set to zero, the bottom most curve in each figure, IFNs gradually degrade during the 4-5 days after the peak.

Increasing the VAK Activation Rate

The VAK activation rate is increased from 1.0E-4 to as high as 8.0E-4.

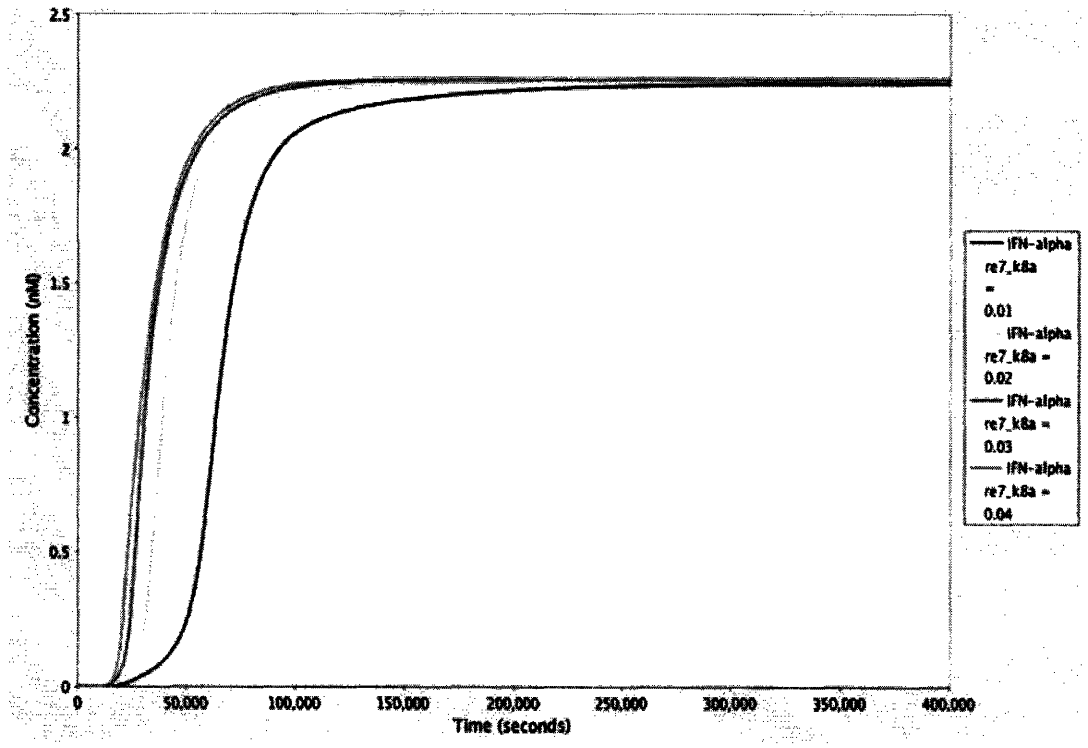


Figure 6-33 – IFN-Alpha is relatively unaffected by changes to VAK activation rate..

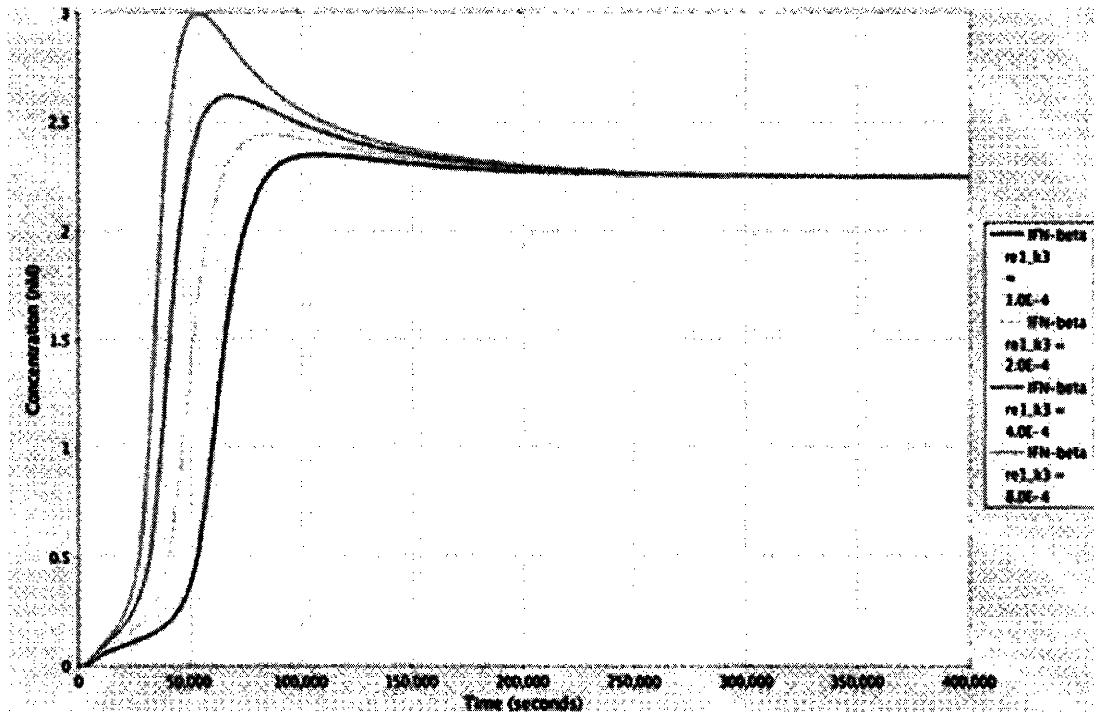


Figure 6-34 – IFN-Beta peak is raised with increase of VAK activation rate..

We now refer to Figure 6-33 and 6-34 above. Increasing this rate of VAK activation does not significantly affect the IFN-Alpha curve. Increasing the rate, however, does raise the peak of IFN-Alpha. Recall, that VAK phosphorylates IRF-3, which is a transcription factor for IFN-Beta activation. Thus, it makes sense why the peak is raised during the initial phase as more IFN-Beta will be created; however, this change does not affect the steady state level of both IFN, which suggests that the steady state level is independent of initiation process, but is determined by the balance between positive feedback amplification and negative feedback control.

Increasing Rate of Transcription of IFN Beta

The rate of transcription of IFN-Beta is set from the original value 1.0E-2 to as high as 4.0E-2.

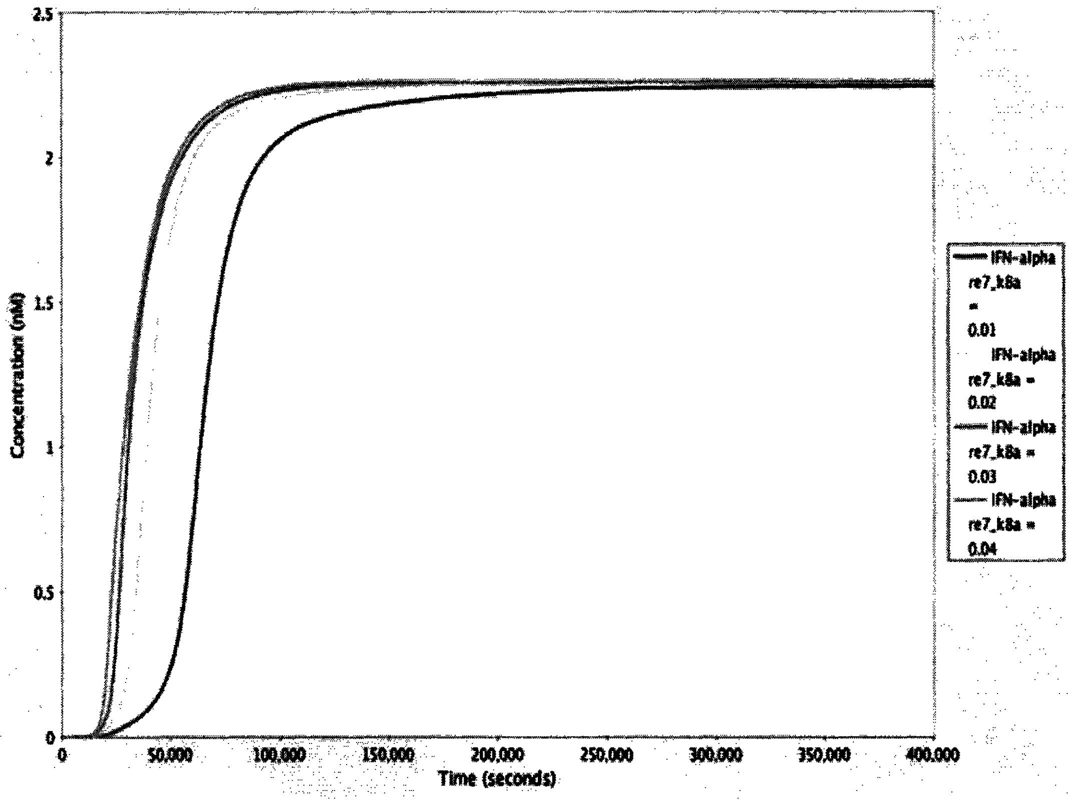


Figure 6-35 – IFN-Beta is relatively unaffected.

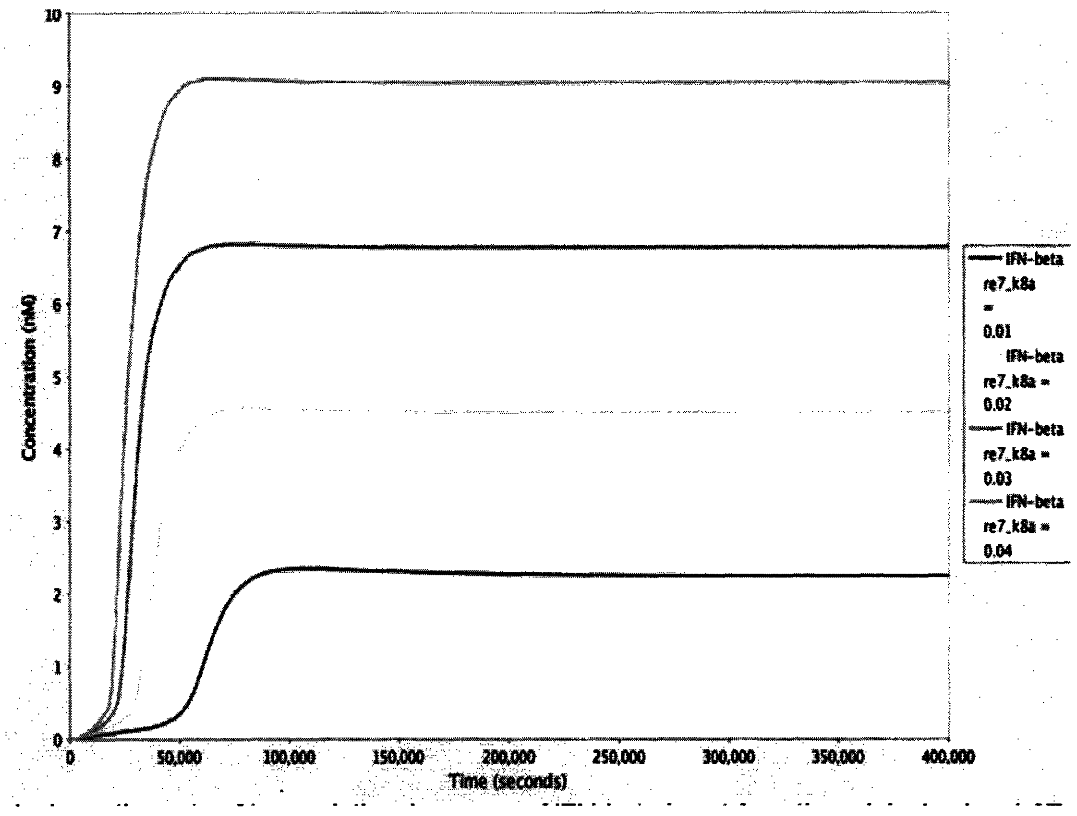


Figure 6-36 – IFN-Beta production increases with rate of transcription increase.

We now refer to Figure 6-35 and 6-36. The steady state concentration of IFN-Alpha is not affected. However, the steady state concentration of IFN-Beta significantly increases. This phenomenon is possible if conformational change of RNA polymerase induced by binding to IFN-Beta gene and the transcription factor is more favorable.

Decreasing Association Rate of IFN-Beta Transcription

The physical association process of IFN-beta is set from the original value 400 to as high as 1600 (high value means less efficient association). This is possible if there are more transcription factors required to diffuse to the transcription starting sites to initiate transcription.

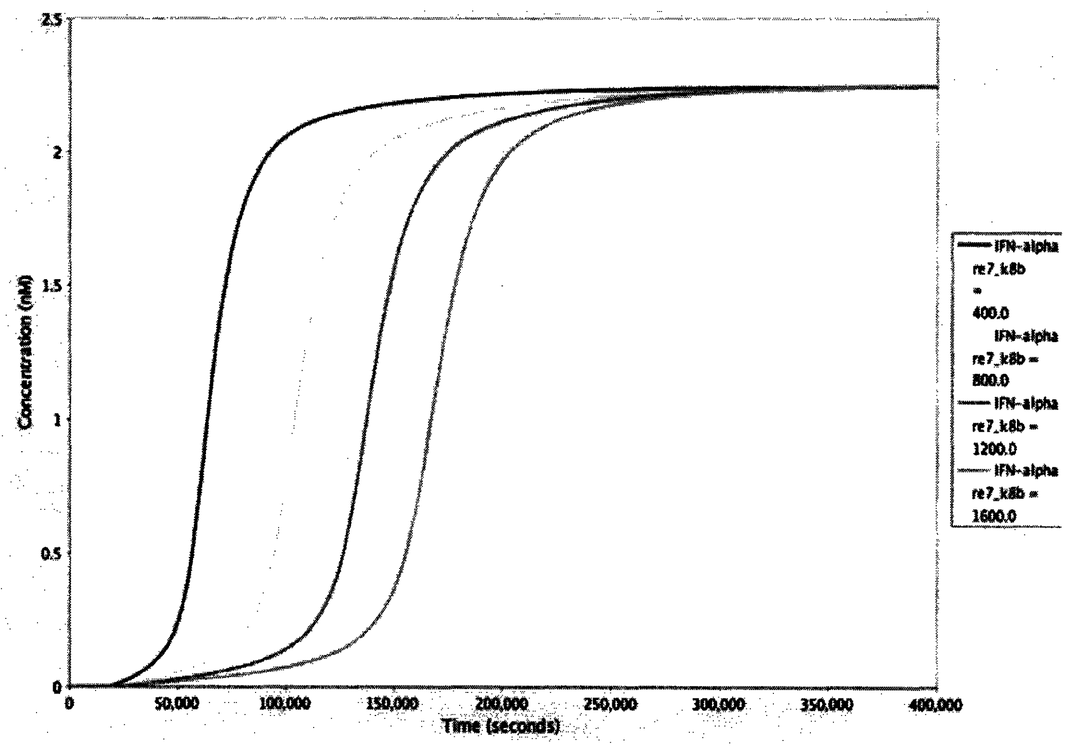


Figure 6-37 – IFN-Alpha reaches same steady state; however, greater delay exists in reaching steady state with decreasing the association rate.

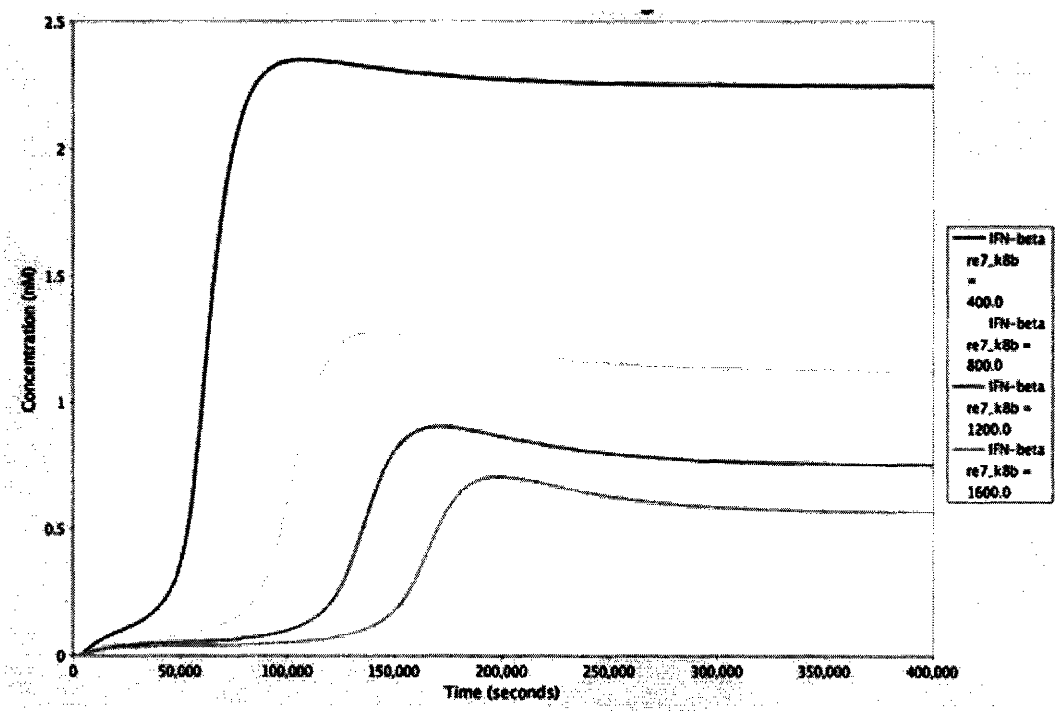


Figure 6-38 – IFN-Beta lowers as the degradation rate of SOCS1 is increased (more SOCS1 is produced to suppress production of IFN-Alpha).

We now refer to Figure 6-37 and 6-38. IFN-Alpha reaches same steady state; however, greater delay exists in reaching steady state with decreasing the association rate. This change significantly reduces the IFN-Beta produced, and increases the time required for the system to reach steady state. These figures and the previous set of modifications suggest that the ratio between the two IFNs is greatly affected by the relative efficiency of transcription process. Similarly, the relative translation rate (production rate) and protein degradation rate also plays a key role in determining the steady state level of both IFNs.

Increasing Rate of pSTAT2c-IRF9c Complex Transport to Nucleus

The rate of pSTAT2c-IRF9c complex transport rate to nucleus is set from the original value 0.005 to as high as 0.02. This complex plays a major role in the nucleus and in transcribing IRF-7, which itself is a transcription factor for IFN-Alpha and IFN-Beta during the amplification cycle. As a result, there are more IFNs being produced which leads to a higher steady state level for both IFNs as shown in Figures 6-39 and 6-40.

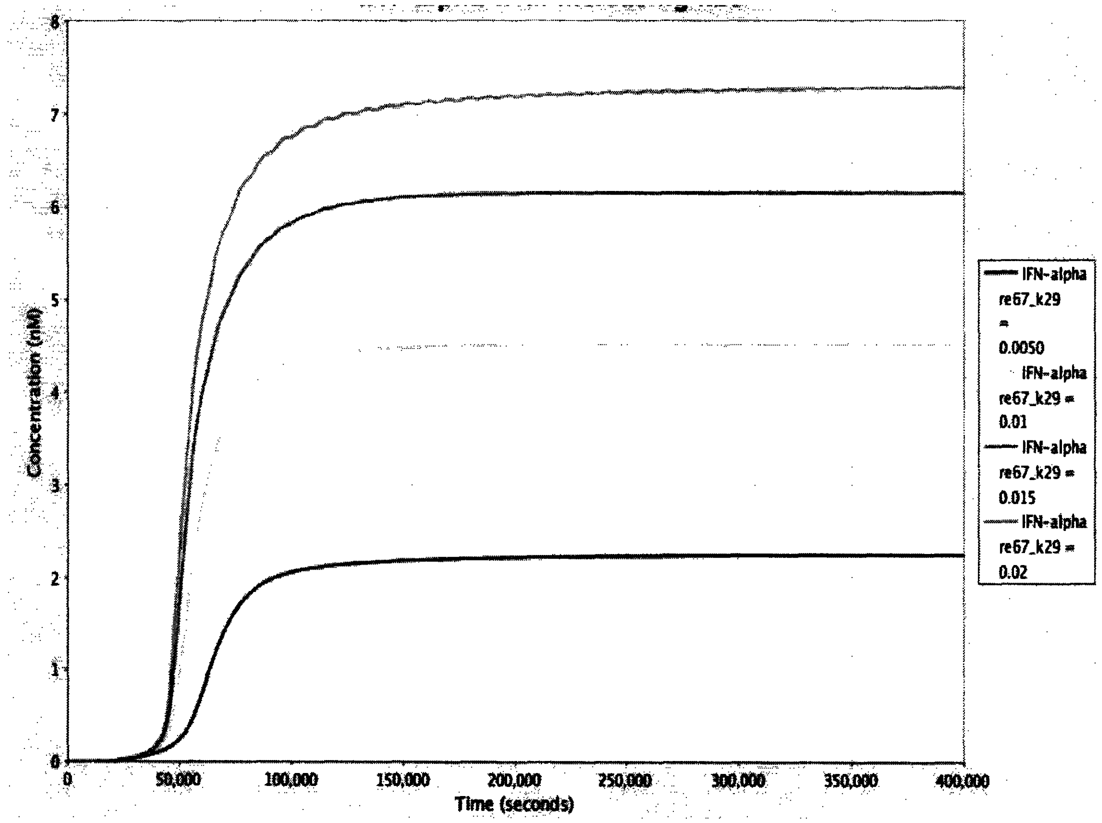


Figure 6-39 – IFN-Alpha increases with greater complex transport.

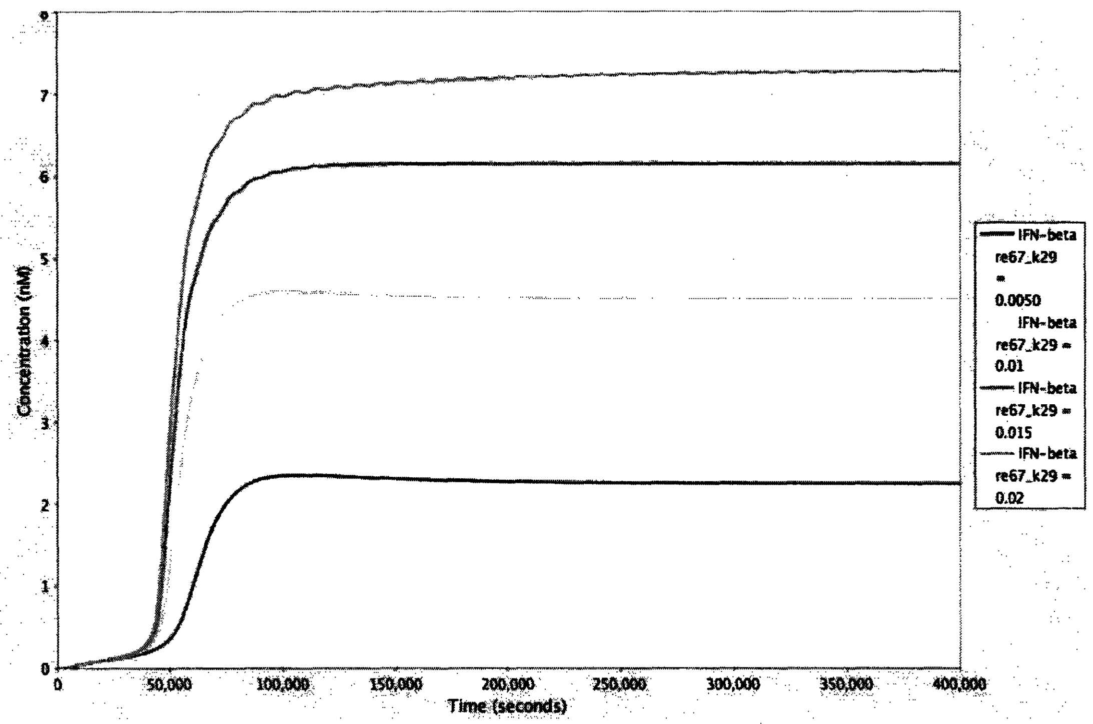


Figure 6-40 – IFN-Beta increases with greater complex transport.

6.8 Summary

The purpose of this chapter was to demonstrate that Cytosolve can be used to solve a heretofore unsolved problem by integrating multiple biological pathway models. Cytosolve has successfully integrated four biological pathway models to create and integrative model of the IFN response to virus infection. This is the first time such an integrative model has been developed.

More importantly, the integrated model has verified known phenomena and has offered new insights to biological phenomena.

First, the IFN Beta is produced in the first 30 to 40 minutes as expected by experimental data.

Second, IFN-Alpha begins production in after ~3 hours delay time as is the approximate expected time required for the positive feedback cycle to start.

Third, both IFNs reach their peak in the ~20 hour range as predicted by various experimental research as shown in Figures 6-41, 6-42, 6-43 (Cella, 1999; Cooley, 1987; Takauji, 2002).

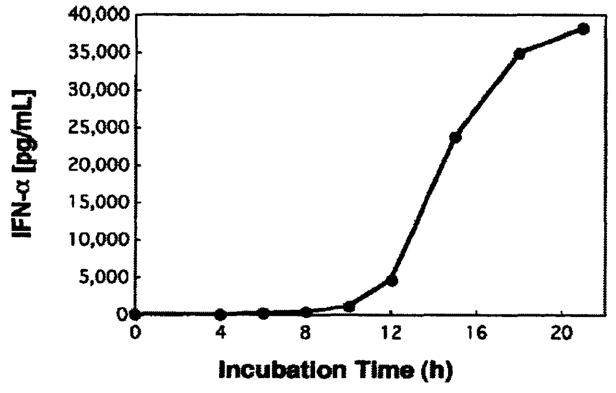


Fig. 1. Time course of CpG-DNA-induced IFN-α production in PDC. IFN-α production was analyzed in PDC incubated with CpG DNA (5 μM) for the indicated periods. The culture supernatants were harvested, and the amounts of IFN-α were measured by ELISA. The data shown are representative of three experiments using PDC from different donors. IFN-α was not detected at each time point throughout the culture with medium alone or control oligo 2GC in

Figure 6-41 - IFN-Alpha reaches a peak ~20 hours (Takauji, 2002).

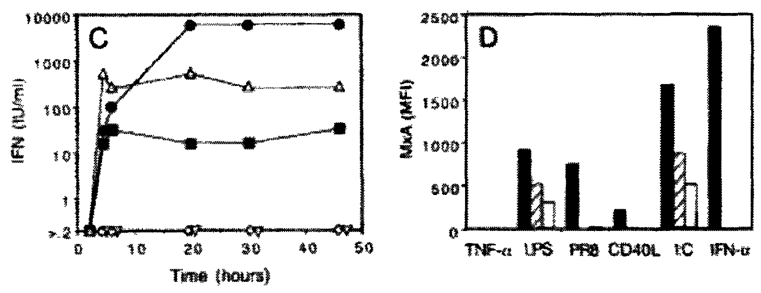


Figure 3. MxA expression is rapidly induced in DCs by LPS, poly I:C and viral infection. Time course of MxA upregulation as detected by intracellular staining (A) or immunoblotting (B). (C) Time course of type I IFN production in culture supernatant. DCs were stimulated with the following: 50 U/ml IFN-α (O, panel A only), LPS (Δ), 20 μg/ml poly I:C (■), 1 HAU PR8 (●), TNF-α (∇), and CD40L (◇). (D) MxA induction after 5 h of stimulation in the absence (black bar) or in the presence of two neutralizing sheep antisera to human type I IFN: Iivari, hatched bars, and Kaaleppi, empty bars.

Figure 6-42 - IFN-Alpha reaches a peak ~20 hours (Cella, 1999).

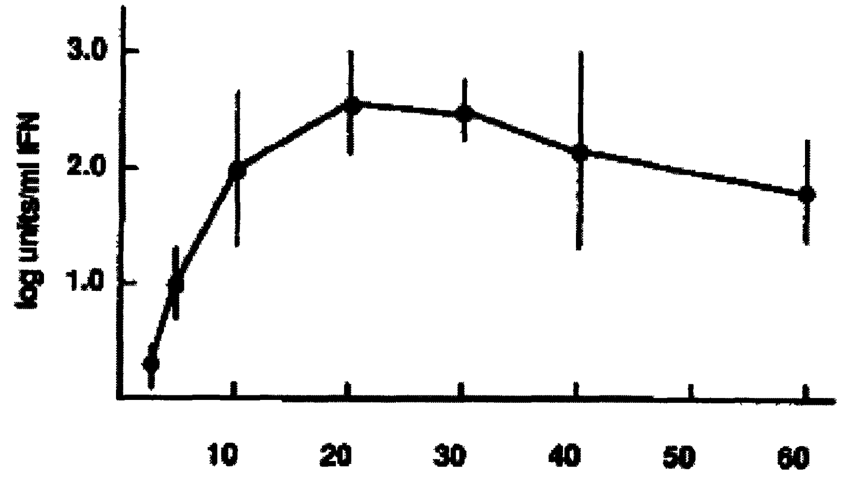


Figure 6-42 - IFN-Alpha reaches a peak ~20 hours(Cooley, 1987).

Four, the initial production of IFN-Beta after the 40 minute period and before the 3.3 hour period is defined by a marked increase in IFN-Beta. This makes sense since most of the IRF7 produced initially is used for IFN-Beta; however, as IFN-Alpha starts to be produced, both IFNs share in the consumption of IRF-78.

Fifth, the integration of the SOCS1 for regulating the IFN response is of utmost value; otherwise, while still valuable, the integrated model would only provide the amplification cycle, not the regulation and balance phenomena provided by the SOCS1.

Sixth, the time scale of the integrated model also matches the various experiments. For example, IFN-Alpha starts its production in Figure 6-41 during the 4-6 hour range as also shown by the integrated model at ~4 hours. In addition, IFN-Beta, based on most literature starts within the ~half-hour, as is also predicted by experiments.

Thus, we have an integrated model of IFN response to virus infection that can be used as the basis for studying biological phenomena. In addition, this model, with the Cytosolve approach, can be expanded and refined by adding new models and/or updates to existing models. The relative ease by which Cytosolve supports such reintegration was effectively demonstrated in Section 6.6.

In summary, Cytosolve has shown its viability to integrate an ensemble of biological pathway models in a scalable manner to create an integrated model to explore new biological phenomena.

Chapter 7

Quantitative Methodology to Evaluate Architectures for Integrating Biological Pathway Models

7.0 Introduction

Currently, the design of computing architectures is a "black art". Over the past few years, in other field such as finance, e-business and defense, there has been a trend to define formalisms, both qualitative and quantitative methods, to evaluate computing architectures (Clements, 2007; Dabous, 2005; Kazman, 2001). Systems biology can benefit from this trend and emerging body of work. In the area of e-business applications, a very nice formulation and formalism for deriving types of architectures based on specifying architectural design has been accomplished (Dabous, 2005). Others have also defined quantitative methods for evaluating a particular architecture's value to stakeholders based on their particular needs and priorities (Kazman, 2001). In recent work, there has also been an attempt to quantify the economic impact of architecture decisions (Clements, 2007).

The earlier chapters of this thesis have served to provide a framework for recognizing the complexity of development and integration of biological pathway models, given the experimental nature of biology which requires enormous effort of focused teams to just characterize one protein or one protein-protein interaction. In addition, early chapters discussed the effort required to develop and maintain just one biological pathway diagram or model. We also reviewed the value of integrating biological pathway models to provide new understanding of cellular function. The main architectural approach that exits today for integrating biological pathway models is a monolithic approach that involves the manual or semi-automatic creation of a single source code from multiple biological pathway models' source code, which is then run on one central computer. This thesis has presented an alternative distributed and parallel approach for integrating biological pathway models. Initial tests were run to understand the computational times and accuracy of the Cytosolve approach versus the monolithic approach using the

Kholodenko EGFR model. The Cytosolve approach was then used to solve the IFN response to viral infection.

This chapter serves to provide a unique quantitative approach for evaluating and comparing architectures for integrating biological pathway models by uniquely combining the approaches of two works by (Dabous, 2005; Kazman, 2001) and applying this approach to systems biology. Specifically, we use our current knowledge of extant monolithic approaches and the distributed and parallel approach of Cytosolve to develop this quantification. The second section presents the mathematical formalism to represent the architectural notation. The third section presents the critical requirements for evaluating architectures for integrating biological pathway models. The fourth section presents different types *stakeholders* of an architecture. The fifth section provides the description of the two main architectures for integrating biological pathway models. The sixth section presents the architectural design elements. The seventh section provides the architectural design alternatives. The eighth section provides quantitative architectural selection by stake holder. The ninth section provides a conclusion and summary of results for this chapter.

7.1 Architectural Notation

This section provides the mathematical formalism use to represent the architectural notation. This notation is used for the first time to describe architectures for integrating biological pathway models; however, it combines the approaches of previous work (Clements, 2007; Dabous, 2005; Kazman, 2001) to evaluate architectural designs in various other non-biological fields.

Biological Pathway Models

Earlier we used the notation M to denote biological pathway models. Here, we still refer to M, but define it in the context of the architectural notation and lower case m refers to a particular biological pathway model. Here, a biological pathway model corresponds to a particular model that is part of a larger cellular function, also defined below formally, and involves a set of species and molecular interactions, as previously defined. We now define using standard set theory nomenclature:

$$M^{all} = \{m_i : 1 \leq i \leq |M^{all}|\}$$

to represent the set of *all biological pathway models* within the whole cell.

Common Biological Pathway Models

Common biological pathway models refer to a set of biological pathway models that are similar. This could be the real case in biology where two different research groups may be working on the exact same biological pathway; however, one group has different species, interactions and rate constants than another group. Or it could be the case where there are the exact same number of species and interactions but the rate constants vary. We assume in this case that the implementations of these models are not exactly the same. We now define:

$$C = \{c_i : 1 \leq i \leq |C|\}$$

to represent the set of all groups of *equivalent biological pathway models.* With each,

$$c_i \subset M^{all}$$

being the ith set of a number of equivalent biological pathway models such that

$$|c_i| \geq 1$$

and,

$$c_a \cap c_b = \varnothing : a \neq b$$

and,

$$\bigcup_{i=1}^{|C|} c_i = M^{all}$$

Cellular Functions

Cellular functions are derived from linking a set of biological pathway models and also represent the connections and flow between one biological pathway model and another. Typically a diagram is associated with a cellular function. In Figure 7-1, are various biological pathways linked for a portion of the cellular

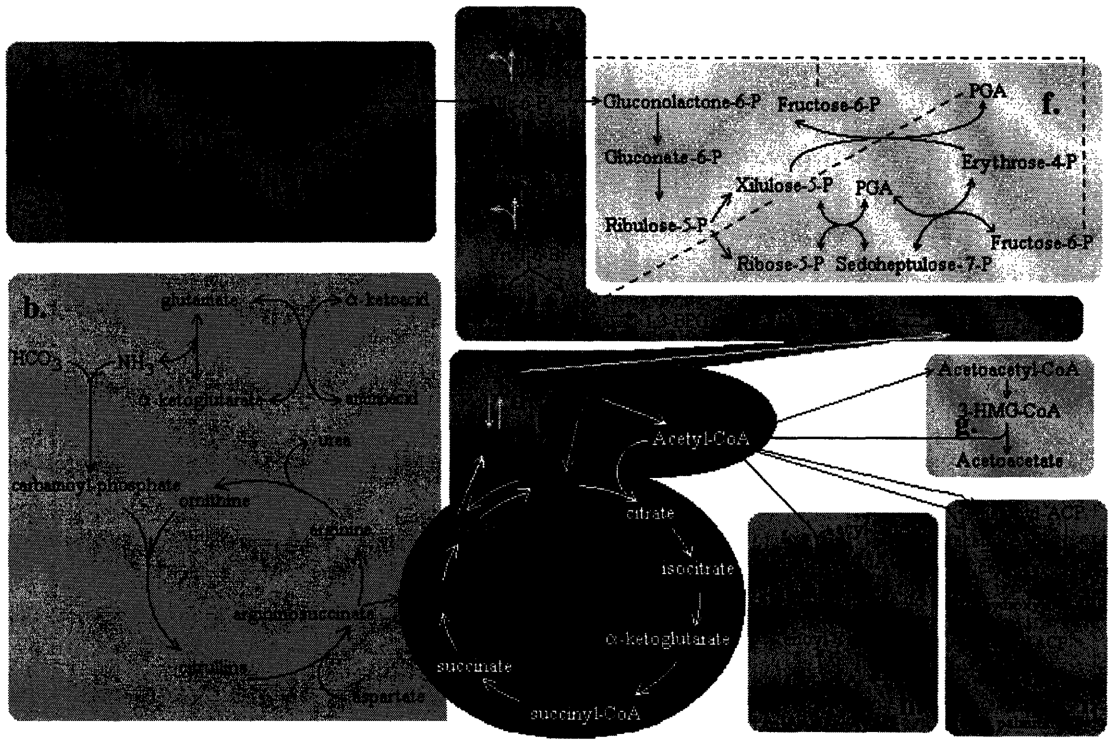


Figure 7-1: An example of how biological pathways interact to support a cellular function. Eight (8) biological pathways including **a.** glycogen metabolism, **b.** amino acid degradation and urea cycle, **c.** glycolysis, **d.** gluconeogenesis, **e.** citric acid cycle, **f.** pentose phosphate, **g.** ketogenesis, **h.** fatty acids beta-oxidation, and **i.** fatty acids synthesis involved in the cellular function of metabolism (Silva, 2002).

function of metabolism. We now define:

$$F = \{f_i : 1 \leq i \leq |F|\}$$

to represent all the *cellular functions* and we use the notation:

$$pathways(f_i) \subseteq M^{all}$$

to represent a function which identifies the set of biological pathway models that are required by a particular cellular function *i*. This function will return the set of all biological pathway models within its cellular function. For Figure 7-1, for example, 8 different biological pathway models will be returned. We will also assume that every:

$$m \in M^{all}$$

has at least one corresponding

$$f \in F^{all}$$

such that

$$m \in pathways(f_i)$$

Or,

$$\bigcup_{i=1}^{|F|} pathways(f_i) = M^{all}$$

Existing Biological Pathway Models

Currently approximately 300 biological pathway models are published within a variety of databases, publications and repositories. These we consider to be existing biological pathway models. We will assume two classes of such models: 1. where the source codes are accessible and 2. where the source codes are not accessible. While some academic and research organizations will freely publish their model source codes, certain commercial for-profit organizations will not publish their source codes. We denote:

$$(P \cup O) \subseteq M^{all}$$

where, **O** represents the set of all the *existing* biological pathway models where source code *is* available

$$O = \{o_i : 1 \leq i \leq |O|\}$$

and where, ***P*** represents the set of all existing biological pathway models where the source code is *not* available. Interaction with these kinds of biological pathway models can only be achieved through defined interfaces. We will also use the following notation:

$$P = \{p_i : 1 \leq i \leq |P|\}$$

to denote those existing biological pathway models without source code accessibility. Note that within ***P*** there may be biological pathway models which are similar. In such a case if,

$$m_x, m_y \in p_i$$

then,

$$\{m_x, m_y\} \subseteq c_j \forall c_j \in C$$

Pathway Communication Methods

Pathway communication methods refer to the way a particular module in the architecture is accessed. We define the following notation:

$$T = \{t_i : 1 \leq i \leq |T|\}$$

to represent the different communication methods. Here are four examples of communication methods identified by (Dabous, 2005):

- T_1 includes the communication at the network protocol levels such as TCP/IP packets
- T_2 includes the use of facilities offered by the operating system(s) and Remote Procedure Calls (RPCs)
- T_3 includes the use of distributed object technology interfaces such as CORBA DIL and Java RMI
- T_4 includes the use of service-oriented standards such as Web Service standards like SOAP/WSDL.

Cell Definition

Based on the notation developed in the previous discussions, we define the cell by a tuple denoted by *<O, P, M^{all}, C, F>*. To conceptually illustrate how this notation can be used, we assume that a particular cell has 4 cellular functions, 15 biological pathway models, 12 groups (labeled **C** in Figure 6-2) of common biological pathway models, and 7 biological pathway models which exist (among which 3 do not have source code availability) and 8 which have yet to be built. Figure 7-2 illustrates this diagrammatically.

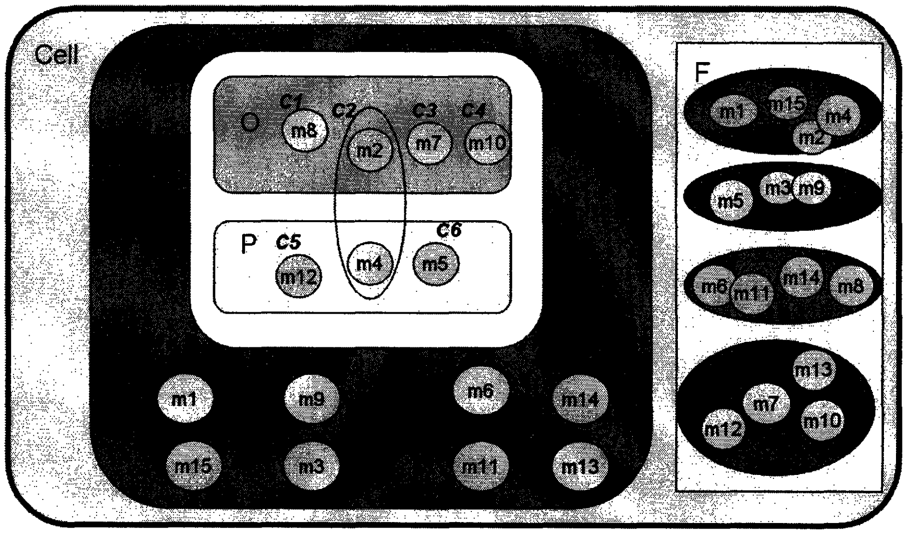


Figure 7-2: Formal representation of a **Cell** using architectural notation references. M^{all} is the set of all possible biological pathway models. **O** is the set of models which have source code available and accessible. **P** is the set of models for which source code is not accessible. **C** denotes the sets of models that are similar. **F** are the functions that the cell provides by integrating various models.

7.2 Critical Requirements for Architectural Evaluation

In previous chapters we have itemized numerous functional architectural requirements for a computing architecture to integrate biological pathway models. In this section, we propose four critical architectural evaluation requirements to assess competing computational architectures.

Low Development Costs

The development costs represent the overall effort in developing a whole cell model or even a cellular function, which is the integration of multiple biological pathway models. One way to perform such calculation is to use Person Months (PM) to determine how many developers will be required to integrate a set of biological pathway models to produce the integrated model. Development costs can also include the cost to integrate just *one* biological pathway model into an existing integrated model. Alternatively, development costs can be associated with the need for a user to swap one similar biological pathway model with another similar biological pathway model.

Performance

This requirement addresses the need for an implementation to be efficient. Efficiency includes fast computation time of the integrated model as well as the ability to efficiently handle complex biological pathway models where there may be varying time scales across a set of models.

Low Maintenance Costs

This requirement addresses the issues of maintaining the integrated model. As discussed earlier, any one biological pathway model may be changing constantly due to new biological experiments; thus, as time progresses and the integrated model grows, maintenance costs can potentially grow, based on the architecture selected.

Security

This requirement addresses the need for ensuring that certain biological pathway models may need to be secured with limited access to the source codes. In biology, certain pharmaceutical companies, who have invested significant amounts in development of a particular biological pathway model, will require that such a model's source code is secure or it can only be accessed via a firewall at a distributed location.

There are potentially many other requirements for evaluating architectures; however, in this thesis we propose that the above four be the critical criteria for evaluating an architecture.

7.3 Stakeholders of an Architecture

System architectures are not developed in a vacuum. An architecture will be *used* by individuals or stakeholders for a particular purpose (Kazman, 2001). While there may be many types of stakeholders, using the criteria established in the previous section, we identify four types of stakeholders based on the needs that will be served by the architecture. In the discussion below, for each stakeholder, we qualitatively assign their relative weight or *Stakeholder Score* (*SKScore*) for a particular criteria. In this approach a total of 100 points is assigned across the four criteria identified in the previous section to denote the relative importance of a criterion to a stakeholder.

Biologist

We define for the purpose of this discussion that a *biologis*t is a stakeholder who: 1. may from time to time develop a biological pathway model based on experimental data and 2. wishes to integrate her model into an existing integrated model or may wish to swap out a model from the integrated model to test the efficacy of their model. For a biologist, element (2) is of utmost value in determining how well their model (and the results of their experiments) compares with existing efforts. Figure 7-3, using the above criteria, qualitatively assigns relative importance to the four different criteria, from a biologist's view point, thus providing a unique profile of this stakeholder.

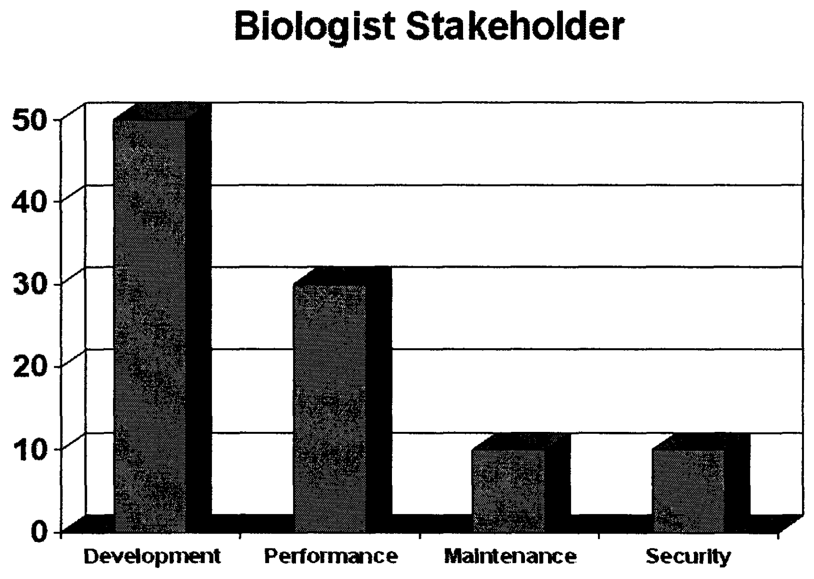


Figure 7-3: Biologists relative level of importance for architectural criteria.

In this case, we have assigned a high value of importance to keep Development costs low, since a biologist does not want to be burdened by significant development efforts to link one of their biological pathway models with an existing integrated model, for example. Performance, relative to the Development criteria, would be the next important criteria, with Maintenance and Security being the least concern to the biologist.

Systems Biologist

We define for the purpose of this discussion that a *systems biologist* is a stakeholder who: 1. focuses on developing a biological pathway models, 2. integrates existing biological pathway models to build larger models, and 3. has a much higher degree of computing and programming experience than a biologist. For a systems biologist, the ability to integrate existing models is important to their goals of modeling larger systems such as cellular functions and the cell. The ability to swap in new models, given updates from biological

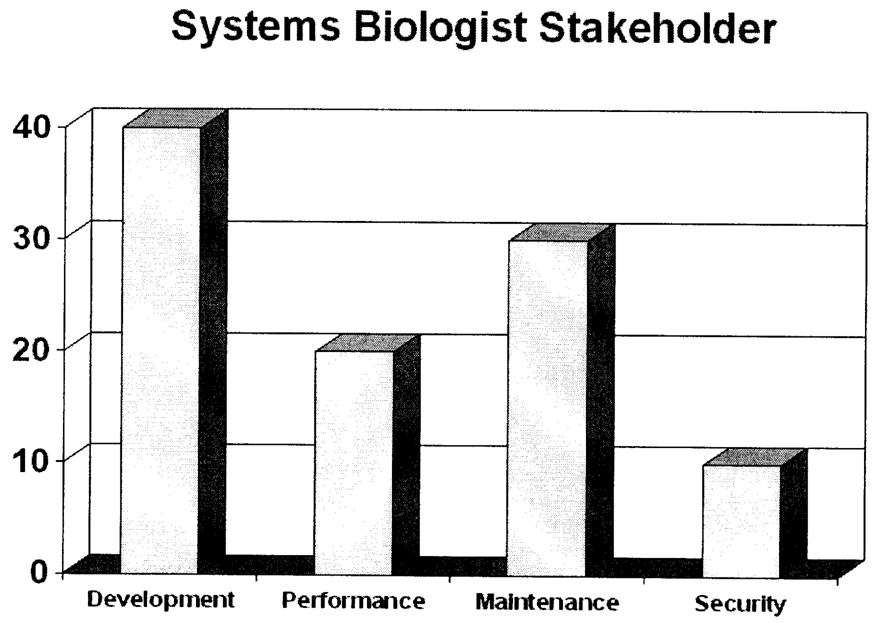


Figure 7-4: Systems biologists relative level of importance for architectural criteria.

experiments, will also be important to this stakeholder Figure 7-4, using the above criteria, qualitatively assigns importance to the four different criteria, from a systems biologist's view point, thus providing a unique profile of this stakeholder. In this case, as shown in Figure 7-4, we have assigned the highest values to low Development and Maintenance costs, next being Performance and finally Security. This makes intuitive sense since a systems biologists' interest is to rapidly develop larger pathway model from smaller one's will ensuring ease of maintenance as models constantly change.

Pharmaceutical Company

We define for the purpose of this discussion that a *pharmaceutical company* is a stakeholder who: 1. develops very particular models to support their internal initiatives for drug discovery, 2. they may wish to integrate their model into an existing integrated model or may wish to swap out a model from the integrated model to test the efficacy of their model and 3. they do not want others in the public domain to access the source codes of their models. Figure 7-4, using the above criteria, qualitatively assigns importance to the four different criteria, from a pharmaceutical company's view point, thus providing a unique profile of this stakeholder.

In Figure 7-4, we have assigned the highest value to Security of their biological pathway models. Next to these criteria, relative equal value has been assigned to the criteria of Development, Maintenance and Performance.

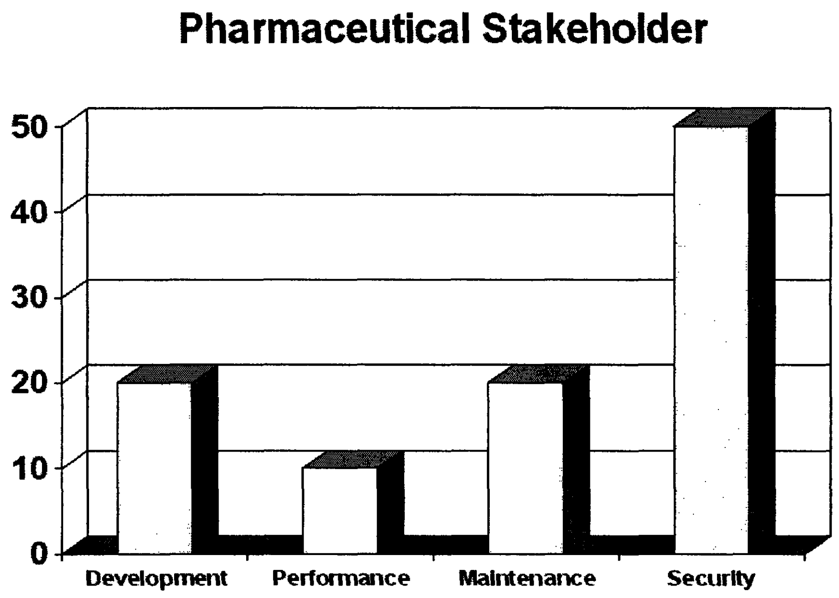


Figure 7-5: Pharmaceutical company's relative level of importance for architectural criteria.

Consumers

We define for the purpose of this discussion that a *consumer* is a stakeholder who: 1. may wish to use an integrated model of biological pathways, 2. vary certain parameters including perhaps their own personal data (such as blood test levels of certain species concentrations as initial conditions) and 3. will not do any development work of any individuals models or any such integration. Consumers of integrated models may in fact be a student who wishes as class exercise to vary parameters of a model of metabolism to gain insights or to merely answer some homework questions. Consumers in the future may be web-based users wishing to perform self-diagnosis as a part of a larger personalized medicine application. Figure 7-6, using the above criteria, qualitatively assigns importance to the four different criteria, from a consumer's view point, thus providing a unique profile of this stakeholder.

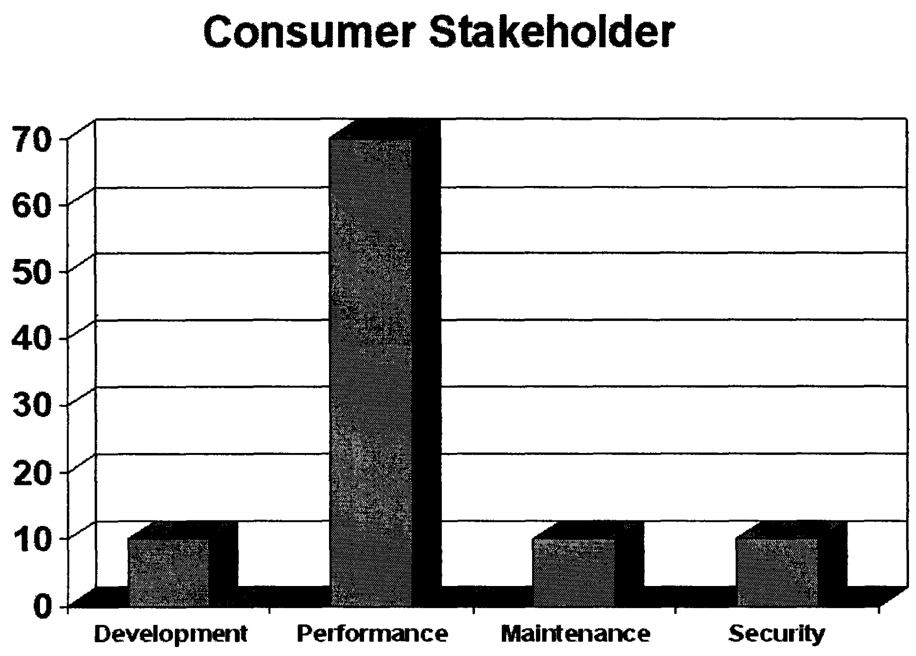


Figure 7-6: Consumers level of importance for architectural criteria.

As is evident from this figure, a consumer's main criteria of importance is performance. For example, if the consumer is using a web-based application to vary parameters and is seeking certain solutions, waiting hours or even minutes for execution will not be acceptable.

7.4 Description of Architectures

In the Chapter on *Prior Work,* we determined two broad classes of architectures: monolithic and messaging. Cytosolve is built on a messaging paradigm. In this section, using the notation developed in Section 6.1, we provide a formalism to describe these two architectures.

Architectural Diagrams

In this discussion, we review the high-level architectural diagrams for both the monolithic and Cytosolve architectures.

<u>Monolithic Architecture</u>

In Figure 7-7 is a high-level diagram of a typical monolithic architecture. This

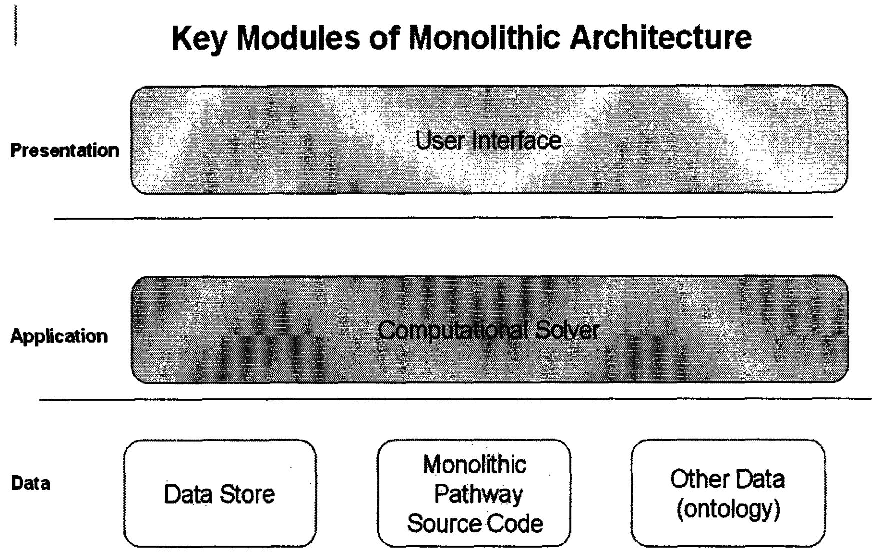


Figure 7-7: High-level monolithic architecture and key modules.

architecture has five key modules: 1. The User Interface at the Presentation Layer, 2. The Computational Solver at the Application Layer, 3. The Data Store, 4. Monolithic Pathway Source Code, and 5. Other data (e.g. Ontology). The last three modules are within the Data Layer. Note, that in this monolithic architecture all of the biological pathway model's source codes have been integrated into one monolithic source code as illustrated.

Cytosolve Messaging Architecture

In Figure 7-8 is a high-level diagram of the Cytosolve architecture. Here, as in

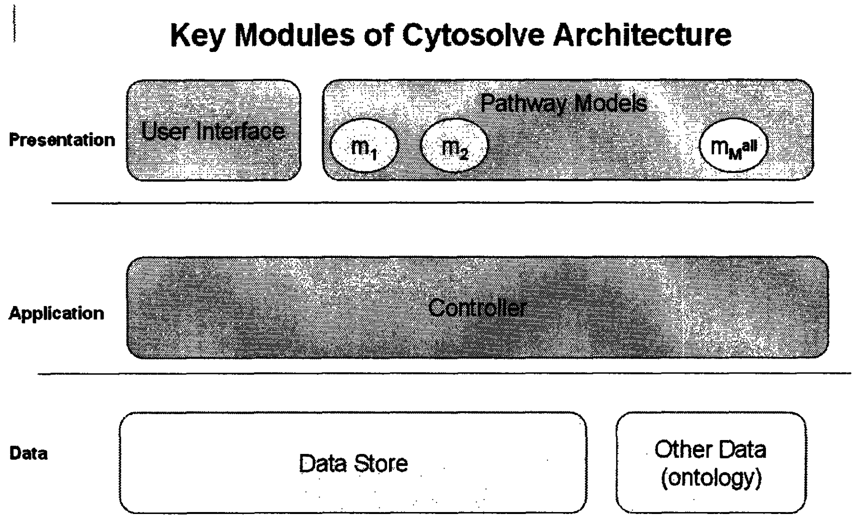


Figure 7-8: High-level distributed architecture and key modules.

the previous diagram, we have left out various details to focus on the key modules. This architecture has 4 + M^{all} key modules: 1. The User Interface at the Presentation Layer, 2. M^{all} number of Pathway Models, 3. The Controller at the Application Layer, 4. The Data Store, and 5. Other data (e.g. Ontology). The last two modules are within the Data Layer. Note that in this Cytosolve architecture the biological pathway model's entire source codes remain distributed within each Pathway Model.

Architectural Modules

An architecture is defined by a set of modules. We define an architecture in terms of a set of modules, ***H***, such that:

$$H = \{Q_i : 1 \leq i \leq |H|\}$$

A module is defined such that,

$$Q_i \in H$$

is described according to four essential features (Dabous, 2005): 1. the tasks that are supported by the module, 2. the set of modules accessed by this module, 3. the set of modules that access this module, and 4. the method by which each module is accessed. Using these features we define any Qi module by the tuple <tasks(Qi), conTo(Qi), invBy(Qi), access(Qi)> using the formulation of (Dabous, 2005) and applying it to systems biology for modeling the cell.

Tasks

We define *tasks*(Q*i*) as a function that identifies the set of tasks that are supported by the module Q*i*. Each task can be one of three types.

The first one is the *implementation of a functionality* such as a biological pathway model (and/or the Controller for example) such that $m \in M^{all}$ that is denoted by $Z(m)$ and is used in three different cases. The first case is when $m \in P$ refers to an existing functionality within a pre-existing component. This would be the case of a biological pathway model that already exists. The second case is when $m \in P$ refers to redeveloping (i.e. reengineering) an existing functionality, such as the effort to recode an existing pathway model. The third case is when $m \in (M^{all} - P - O)$ i.e m is a new functionality such as a new biological pathway model that is not yet implemented.

The second type of tasks correspond to the *implementation of a wrapper* for a functionality m, denoted by $ZW(m)$. It is used when Qi masks out the actual module that embeds $Z(m)$ using a higher level access method.

The third type of tasks correspond to *implementing a cellular function f*, denoted by $ZBL(f)$. We will use the term cellular function to refer to the implementation code that integrates multiple biological pathway models, m. Therefore, each task corresponds to implementation code resulting from applying one of the task constructors Z, ZW, or ZBL on its parameters

Modules Invoked

We define conTo(Qi) $\subset$ H. This is a function that returns the set of modules that Qi invokes while executing its tasks. For example, for the Cytosolve architecture conTo(Controller) would return all the modules at the Data Layer and all the biological pathway models at the Presentation Layer.

Modules Invoked By

We define invBy(Qi) $\subset$ H. This is a function that returns the set of modules that invoke Qi while executing its tasks. invTime(Qi) returns the time of invocation of module Qi, the timestamp denoting the time (e.g. in GMT) on when the module was invoked. For example, for the Cytosolve architecture, invBy(m_1) would return the Controller, since that is the only module that can invoke a biological pathway model.

Access Method

We define *access(Qi)*. This is a function that returns the access method, Ti, used by component Qi. This means that accessType(access(Qi)) $\in$ T while executing its tasks. For example, for the Cytosolve architecture access(m_1) would return T_4 since SOAP/WSDL, Web services is used as the access method.

7.5 Architectural Design Elements

Architectural design elements are used to determine particular architectures. In this section, we will derive the variations of monolithic and the Cytosolve architectures based the following design elements: Model Reuse, Model Migration to Common Format, Distributed Models, Merged Models, Parallel Processing, Serial Processing.

Model Reuse

This design element emphasizes the importance of leveraging existing biological pathway models without having to recode them by wrapping the biological pathway model to provide access by other modules. This design element will create **P** new modules in **H.** Using our formalism,

```
For each $Z(m_k) \in tasks(Qi)$:
    Qx = createNew();
    access(Qx) = Ti; (the access method)
    tasks(Qx) = [ZW($m_k$)];
    invBy(Qi) = Qx;
    conTo(Qx) = Qi
    **H** = H $\cup$ [Qx]
```

This design element allows biological pathway models to be reused, swapped in and out, and shared across different cellular functions, ***F.***

Migration to Common Format

This design element refers to the need to migrate a biological pathway model to one common format (e.g. SBML). In cases where a systems biologists believes that an older format in which the model was coded, for example, in Fortran 77, will become a legacy, the task will be port or migrate that model to a common format. In this case the functionality of that particular pathway model will be redeveloped from scratch in a process that we refer to as 'migration'. There are different possibilities of rearranging the redeveloped functionalities of a biological pathway model in one format into either one other format within one new biological pathway model or splitting them into several biological pathway models. In this case, we define the migration process such that any biological pathway model that is created from this migration is treated the same whether it exists in the collection or is a new biological pathway model.

Here we group each set of equivalent biological pathway models $c \in C$ into one module. Each new biological pathway model contains a task $Z(m_k)$ for each $m_k \in c$. This design element also uses a uniform access method across all new biological pathway models. Using our formalism, we denote this design element as:

```
For each $c_i \in C$:
    Qx = createNew();
    access (Qx) = Ti;
    tasks (Qx) = $[Z(m_k): m_k \in c_i]$
    **H** = $H \cup [Qx]$
```

Distributed Models

A distributed model is one in which each biological pathway model can communicate with other modules of the architecture and the source codes are not merged into one source code. This means that other modules can invoke a model and a model can invoke another module. This is formally denoted as follows:

```
For each $m_i \in M^{all}$:
    $invBy(m_i) \subset H$;
    $conTo(m_i) \subset H$;
```

Merged Models

A merged model is one in which the source codes of all biological pathway models are merged to form one monolithic source code, which becomes a module of the architecture. This module, as shown in Figure 7-7, can be invoked by other modules; however, the merged model cannot invoke other modules of the architecture. This is formally denoted as follows:

```
For each $m_i \in M^{all}$:
    $invBy(M) \subset H$;
    $conTo(M) \not\subset H$;
```

Parallel Processing

Parallel processing is when computation is performed in parallel across a collection of biological pathway models. This means that at each invocation all M set of models are invoked in parallel. The calling module of all M models may not receive the replies at the same time; however, invocation occurs at the same time and each model performs its processing in parallel independent of the other models for an invocation time period. Formally, we define as follows:

For each $m_i \in M^{all}$:

$m_x, m_y \in m_i : x \neq y$

$invTime(m_x) = invTime(m_y)$

Serial Processing

Serial processing is when computation is performed in serial across a collection of biological pathway models. This means that at each invocation each model within the M set of models is invoked at: 1. a different time, and 2. after the completion the invoked module returns control to the module which called it. This is expressed by the following:

For each $m_i \in M^{all}$:

$m_x, m_y \in m_i : x \neq y$

$invTime(m_x) \neq invTime(m_y)$

7.6 Architectural Design Alternatives

All of the above sections serve to help us finally define architectural alternatives. This process involves specifying two elements as defined by (Dabous, 2005): Design Decisions and Alternatives. In this process based on a Design Decision, we elect to use certain design elements, from the previous, section. Table 7-1 below provides us the Design Decisions and the Alternatives based on the design elements.

	Alternatives	
Design Decisions	**1st Alternative**	**2nd Alternative**
DD1	Model Reuse	Migration to Common Format
DD2	Distributed Models	Merged Models
DD2	Parallel	Serial

Table 7-1: Design decisions and alternatives

This table shows that each of DD1, DD2 and DD3 has two alterative designs.

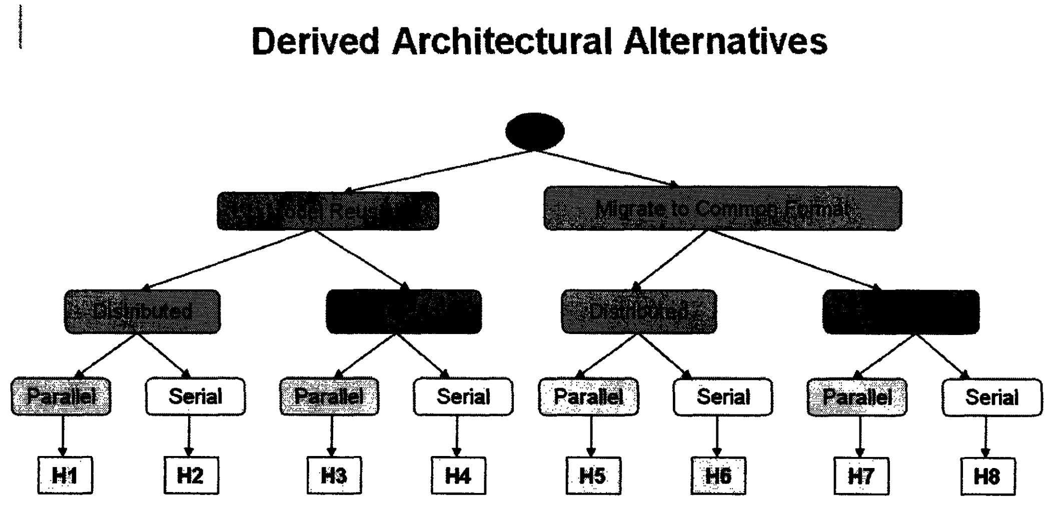


Figure 7-9: Eight possible architectures predicted based on design decisions and alternatives.

Using this Table, we can now construct a tree which shows eight paths which represent all possible architectures, as shown in Figure 7-9.

7.7 Architectural Evaluation

We have now systematically, using architectural formalisms, identified the possible architectures for integrating biological pathway models based on relevant design elements. In this section, we assign quantitative measures to the ability of a particular architecture to satisfy different criteria, previously developed, based on experience in software engineering. The assignment of quantitative measures, in this section, was applied using the author's personal experience; in future research, other approaches could be developed to assign quantitative assignments based on objective historical data. Currently, given that systems biology is a relatively new field, less than five years old, little historical data are currently available.

In an earlier discussion in Section 6.2, the following critical requirements were defined:

- Low Development Costs
- Performance
- Low Maintenance Costs
- Security

We now qualitatively build a matrix, called the *architectural matrix* (*ArchM*$_{i,j}$) that measures these critical requirements across the eight different architectures. In Table 7-2, this matrix is provided based on the author's discretion. The reasoning for which is described below.

	Critical Criteria			
Architecture	**Low Development Costs**	**High Performance**	**Low Maintenance Costs**	**High Security**
H1	3	2	3	3
H2	3	1	3	3
H3	2	2.5	2	2
H4	2	2.5	2	2
H5	2.5	2	3	1
H6	2.5	1	3	1
H7	1	3	1	1
H8	1	3	1	1

Table 7-2: Qualitative favorability ratings by author of critical criteria for each architecture.

Reasoning Behind Quantitative Assignments

In Table 7-2, the author has assigned favorability ratings on a scale of 0 to 3 in increments of 0.5 for the Critical Criteria for each architecture. This assignment was done in a columnar approach and within each column in a relativistic manner.

Low Development Costs

Specifically, for this criterion, the relative costs were estimated in a step-like manner. In the first step, it is surmised that architectures H1, H2 and H5, H6, in which the models are distributed would have the lowest development costs in integrating a new model. We assign in this case H1 and H2 to be the lowest development costs with a rating of 3, while we give H5 and H6 a slighter lower rating or 2.5 since we believe that building a wrapper, in general, to an existing code base will be much easier than completely recoding and migrating an existing code base to a new format, since far more core level software testing will be needed for a model integrated into an H5 architecture.

In the second step, we assign a rating of 2 to H3 and H4 since this describes a module-based monolithic approach (see Chapter 2) which is a far easier integration effort than merging the source codes of all models in a manual or semi-manual monolithic approach.

In the third step, we assign the lowest rating of 1 to H7 and H8, since they represent the manual or semi-manual monolithic approach of merging all the codes into one monolithic source code. In fact, both of these architectures in practice are the same since H7 and H8 are the merged source codes, so there is no distinction between serial and parallel here.

High Performance

Specifically, for this criterion, the relative performance was estimated in a step-like manner. In the first step, it is surmised that architectures H7 (and H8 since running a merged model in serial mode is the same as running it in parallel mode, since there is only *one* model) will be the fastest for obvious reasons since no Transmission Time nor any orchestration of model calculations is needed, and receive a rating of 3.

In the second step, we surmise that H3 and H4 will be the next fastest since while they are merged, they are merged using a module-based monolithic approach which involves intra-process calls between each pathway model. We assign a rating of 2.5 for these two architectures. Note again, as in the reasoning for H7 and H8, whether they run serially or in parallel, since there is only one model, there will be no performance differences.

In the third step, we surmise that H2 and H6 will be the slowest, with a rating of 1, since they are distributed and serially invoked.

In the fourth step, by process of elimination and a relativistic view of earlier assignments, we assign H1 and H5 a rating of 2. Intuitively, these two architectures will be faster than H2 and H6 but slower than the others.

Low-Maintenance Costs

Specifically, for this criterion, the relative to costs were estimated in a step-like manner. In the first step, it is surmised that architectures H1, H2, H5 and H6 will all have the lowest maintenance costs since all models are distributed and if any one model undergoes changes, no recoding of the original model will be needed.

In the second step, we surmise that the hardest to maintain will be the merged models in a single source code format. These are architectures H7 and H8 and receive the lowest rating of 1.

In the third step, by process of elimination and a relativistic view of earlier assignments, we assign H3 and H4 a rating of 2. Intuitively, these two architectures will be easier to maintain, since they are merged using a module-based monolithic approach. If the source code of any one model changes, only that model and the communication interface needs to change.

Security

Specifically, for this criterion, the relative security was estimated in a step-like manner. In the first step, it is surmised that architectures H1, H2 will have the highest security since models are distributed and the source codes need not be revealed.

In the second step, we surmise that the worst security will be models H5, H6, H7 and H8 since all source code needs to be revealed either to merge them or to distribute them since models need to be migrated into a common format.

In the third step, by process of elimination and a relativistic view of earlier assignments, we assign H3 and H4 a rating of 2. Intuitively, these two architectures will be more secure than H5, H6, H7 and H8, since merging them in a module-based approach does not require the need to fully expose th source code, but rather, in most cases, the variable interface.

7.8 Quantitative Architectural Selection for Stakeholder

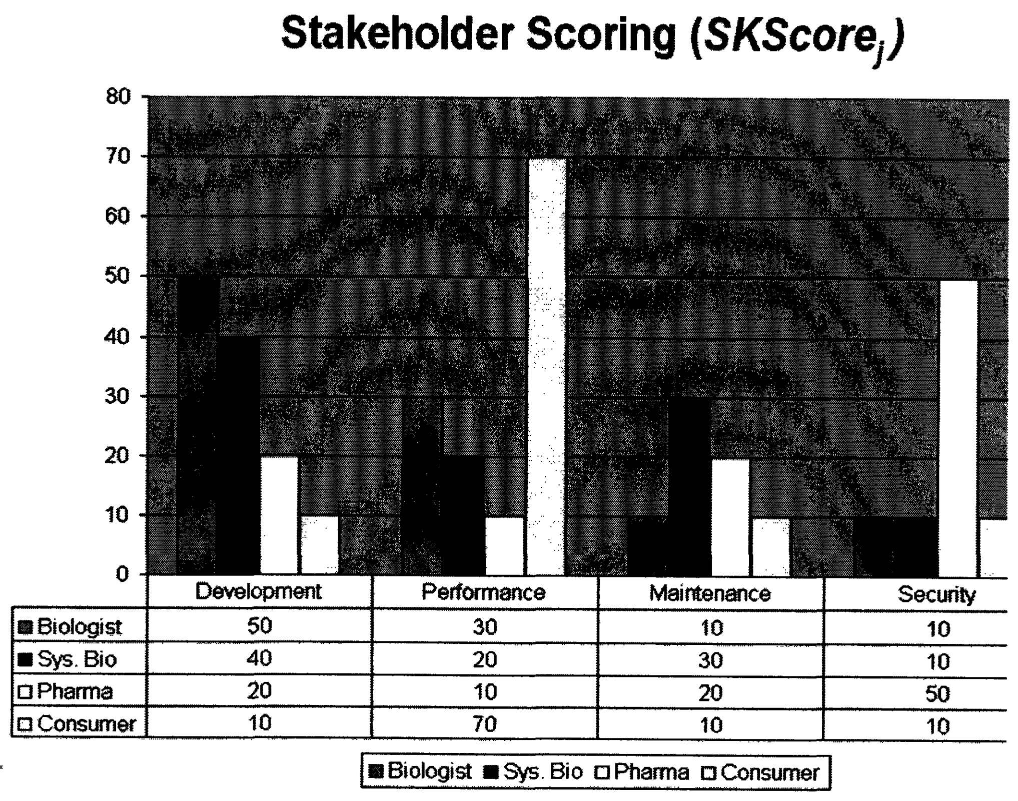
Our goal now is to for each stakeholder: Biologist, Systems Biologist, Pharmaceutical Company and Consumer identify the optimal architecture among the eight that were systematically identified using our process.

Calculation of Value Matrix

Using a similar approach by (Kazman, 2001) we now evaluate the Value of a particular architecture to a stakeholder by using the following formula:

$$Value(H_i) = \sum_j Arch_{i,j} SKScore_j$$

Figure 7-9 contains the summary of the *SKScore*$_j$ across the four different stakeholders of Biologist, Systems Biologist, Pharmaceutical Company and Consumers.



	Development	Performance	Maintenance	Security
■ Biologist	50	30	10	10
■ Sys. Bio	40	20	30	10
□ Pharma	20	10	20	50
□ Consumer	10	70	10	10

Figure 7-9: Stakeholder scoring summary data.

Table 7-3 contains the results of calculating Value (Hi). Here I = 1 to 8, and j = 1 to 4.

Value Matrix of Architecture to Stakeholder

$Arch_{i,j}$

	Biologist	Systems Biologist	Pharmaceutical Company	Consumers	Mean	Rating
H1	270	280	270	270	272.5	1
H2	240	260	240	240	245	2
H3	215	210	215	215	213.75	4
H4	215	210	215	215	213.75	4
H5	225	240	225	225	228.75	3
H6	195	220	195	195	201.25	5
H7	160	140	160	160	155	6
H8	160	140	160	160	155	6
Max	270(H1)	280(H1)	270(H1)	270(H1)		

Table 7-3: Value matrix calculation for each architecture by stakeholder.

7.9 Summary

The Table in 7-3 provides a great deal of information. In Figure 7-10, this table is plotted using a surface plot format. Along the x-axis are the different architectures. Along the y-axis are the different stakeholders. Along the z-axis are the Value scores evaluated using the formula from the previous discussion. This figure serves to show graphically which architectures have the most value to the particular stakeholder.

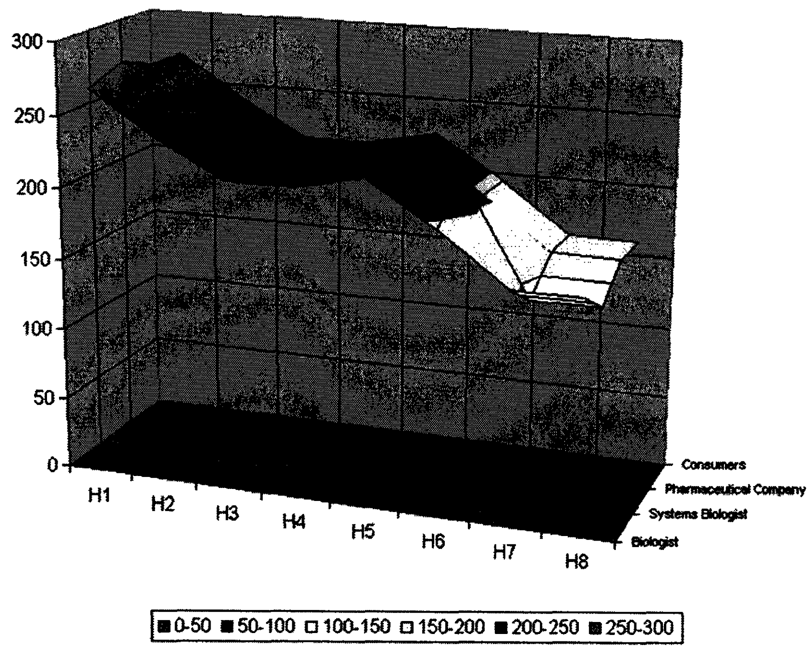


Figure 7-10: Surface plot of Value(Hi) by stakeholder and architecture.

First, the results demonstrate that regardless of the stakeholder, architecture H1 provides the best alternative. H1 represents the architecture that is the dynamic messaging approach and is in fact the Cytosolve architecture! What is interesting to note is H2, involving serial processing, with model reuse is second best architecture. This may be due to the high level of rating assigned for security, low development and low maintenance costs.

Second, the worst architectures are H7 and H8 which are the manual or semi-manual monolithic approaches. These are the monolithic architectures that involve the merging of source code and conversion to a common format such as SBML.

Third, architecture H5 is the third best architecture. This makes sense since this architecture, while requiring migration to a common format, uses the design elements of distributed models in a parallel processing mode. We believe that this architecture can be very optimal for the case where a set of models are in SBML, for example, and without having to merge them into one source code, they can be distributed and solved. In fact, there may be cases where competing standards may have distributed sets of integrated models using the H5 architecture.

The Cytosolve architecture is highly flexible and can operate any mode; however, it was designed to work in a distributed environment, either serial or parallel. Thus, architectures H1, H2, H5, H6 are representative of the Cytosolve architecture.

In conclusion, this chapter has provided a detailed formalism and a quantitative paradigm for assessing architectures for integrating biological pathway models. This analysis quantitatively demonstrates that for the stakeholders considered, the Cytosolve architecture is best suited.

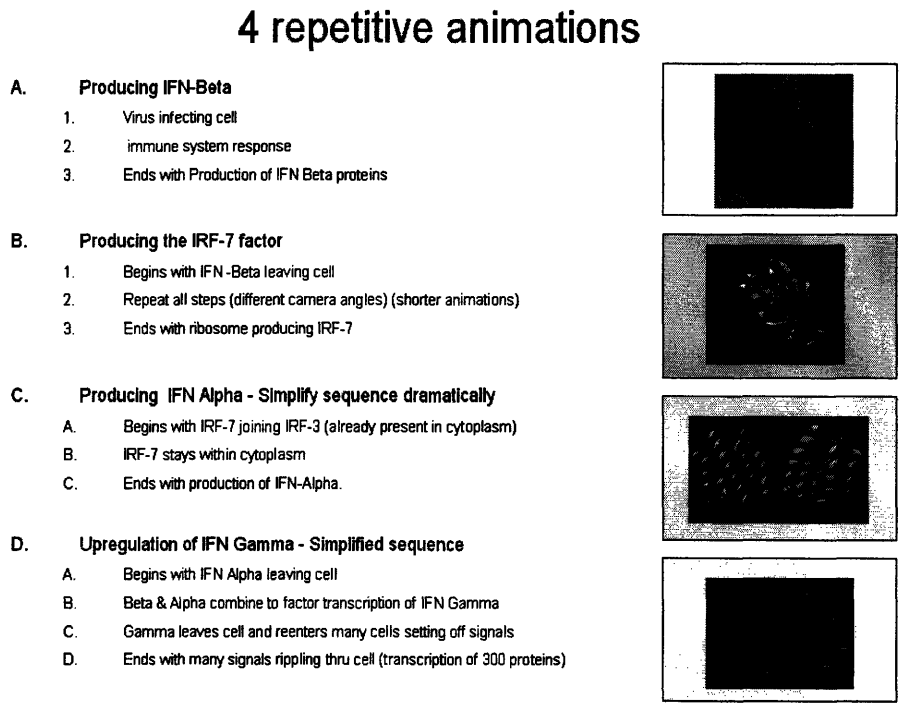
Chapter 8

3-D Animated Video of IFN Response

8.0 Introduction

In the summer of 2005, I met Philip Pfeifer who was in the midst of a career move from electrical engineering to biomedical visualization. In July of 2006, Philip and I reconnected and we agreed to collaborate with some members of his team of artisans who belonged to a virtual guild of 3-d animators. We agreed that I would direct the video sequence and provide them an opportunity to create a great demonstration for their future portfolio, for which they offered to render and animate the video. This chapter provides the details of that effort. The next section contains the storyboards for Scene 1. The third, fourth and fifth sections contain the storyboards for Scene 2, Scene 3 and Scene 4, respectively. Finally section six contains the video itself in DVD format.

In these scenes, Scene 1, 2, 3, and 4 are labeled as A, B, C, and D. Note, Scene D animates the up regulation of IFN-Gamma, something that is not part of the IFN model discussed previously and this scene is highly speculative. Figure 8-1 is the first outline story board.



4 repetitive animations

A. **Producing IFN-Beta**
   1. Virus infecting cell
   2. immune system response
   3. Ends with Production of IFN Beta proteins

B. **Producing the IRF-7 factor**
   1. Begins with IFN -Beta leaving cell
   2. Repeat all steps (different camera angles) (shorter animations)
   3. Ends with ribosome producing IRF-7

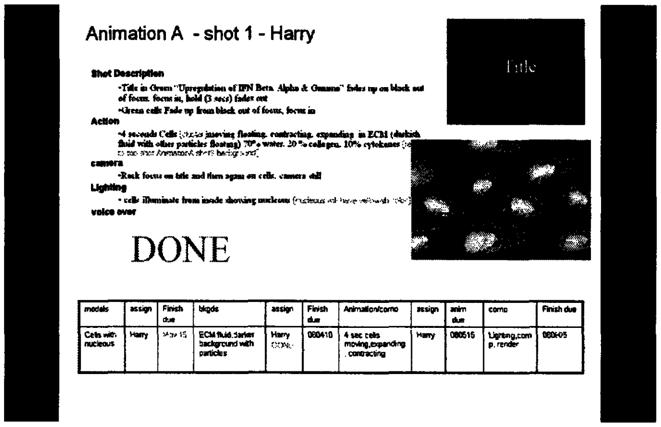
C. **Producing IFN Alpha - Simplify sequence dramatically**
   A. Begins with IRF-7 joining IRF-3 (already present in cytoplasm)
   B. IRF-7 stays within cytoplasm
   C. Ends with production of IFN-Alpha.

D. **Upregulation of IFN Gamma - Simplified sequence**
   A. Begins with IFN Alpha leaving cell
   B. Beta & Alpha combine to factor transcription of IFN Gamma
   C. Gamma leaves cell and reenters many cells setting off signals
   D. Ends with many signals rippling thru cell (transcription of 300 proteins)

Figure 8-1: All scenes A, B, C, D story board.

8.1 Scene 1 - Virus Infection

Scene 1 is the virus infection phase to produce IFN Beta. Storyboards are provided for this phase in Figures 8-2 to Figures 8-6.



Animation A - shot 1 - Harry

Shot Description

- Title in Green "Upregulation of IFN Beta, Alpha & Gamma" fades up on black out of focus, focus in, hold (3 secs) fades out
- Green cells Fade up from black out of focus, focus in

Action

- 4 seconds Cells [illegible] moving floating, contracting, expanding in ECM (darkish fluid with other particles floating) 70% water, 20% collagen, 10% cytokines [illegible]

camera

- Rack focus on title and then again on cells, camera still

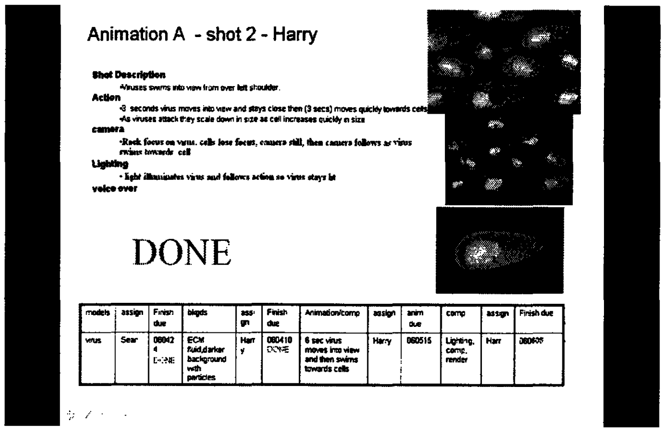
Lighting

- cells illuminate from inside showing nucleous [illegible]

voice over

DONE

models	assign	Finish due	bkgds	assign	Finish due	Animation/comp	assign	anim due	comp	Finish due
Cells with nucleous	Harry	May 15	ECM fluid, darker background with particles	Harry DONE	060410	4 sec cells moving, expanding, contracting	Harry	060515	Lighting, comp, render	060605



Animation A - shot 2 - Harry

Shot Description

- Viruses swims into view from over left shoulder.

Action

- 3 seconds virus moves into view and stays close then (3 secs) moves quickly towards cells
- As viruses attack they scale down in size as cell increases quickly in size

camera

- Rack focus on virus, cells lose focus, camera still, then camera follows as virus swims towards cell

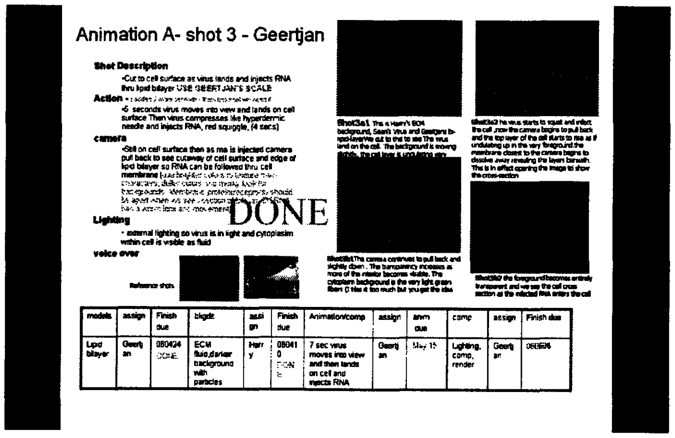
Lighting

- light illuminates virus and follows action so virus stays lit

voice over

DONE

models	assign	Finish due	bkgds	assign	Finish due	Animation/comp	assign	anim due	comp	assign	Finish due
virus	Sear	060424 DONE	ECM fluid, darker background with particles	Harry	060410 DONE	6 sec virus moves into view and then swims towards cells	Harry	060515	Lighting, comp, render	Harr	060605



Animation A- shot 3 - Geertjan

Shot Description

- Cut to cell surface as virus lands and injects RNA thru lipid bilayer USE GEERTJAN'S SCALE

Action [illegible]

- 6 seconds virus moves into view and lands on cell surface Then virus compresses like hyperdermic needle and injects RNA, red squiggle, (4 secs)

camera

- Still on cell surface then as rna is injected camera pull back to see cutaway of cell surface and edge of lipid bilayer so RNA can be followed thru cell membrane [illegible]

Lighting

- external lighting so virus is in light and cytoplasim within cell is visible as fluid

voice over

DONE

Reference shots

Shot3a1 This is Harry's ECM background, Sean's Virus and Geertjans bi-lipid-layer We cut to this to see The virus land on the cell. The background is moving slightly, the cell layer is undulating very

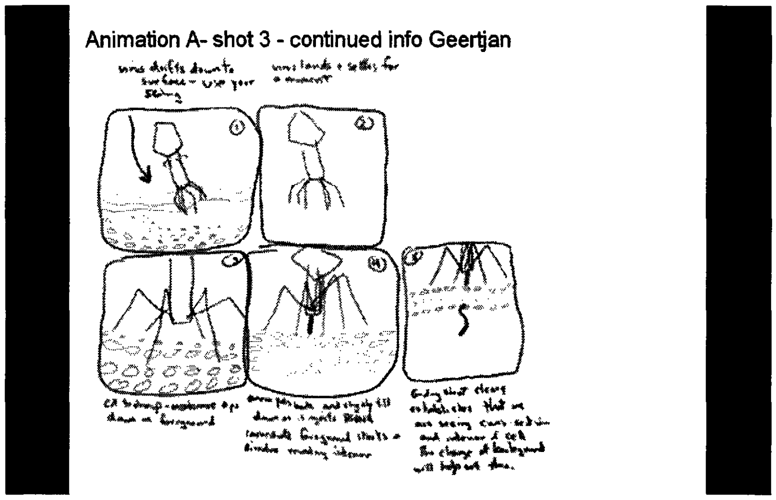
Shot3a2 he virus starts to squat and infect the cell ,now the camera begins to pull back and the top layer of the cell starts to rise as if undulating up in the very foreground the membrane closest to the camera begins to dissolve away revealing the layers beneath. This is in effect opening the image to show the cross-section

Shot3b1 The camera continues to pull back and slightly down . The transparency increases as more of the interior becomes visible. The cytoplasm background is the very light green fibers (I this it too much but you get the idea

Shot3b2 the foreground becomes entirely transparent and we see the cell cross section as the infected RNA enters the cell

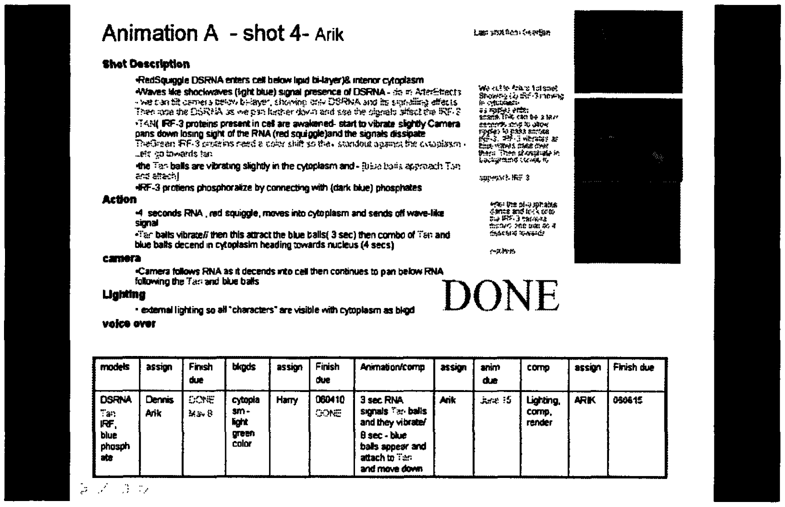
models	assign	Finish due	bkgds	assign	Finish due	Animation/comp	assign	anim due	comp	assign	Finish due
Lipid bilayer	Geertjan	060424 DONE	ECM fluid, darker background with particles	Harry	060410 DONE	7 sec virus moves into view and then lands on cell and injects RNA	Geertjan	May 15	Lighting, comp, render	Geertjan	060606

Figure 8-2: Storyboards for Scene 1.



Animation A- shot 3 - continued info Geertjan

virus drifts down to surface – use your [illegible]
virus lands + settles for a moment
[illegible]



Animation A - shot 4- Arik

Last shot from Geertjan

Shot Description

- RedSquiggle DSRNA enters cell below lipid bi-layer)& interior cytoplasm
- Waves like shockwaves (light blue) signal presence of DSRNA - do in AfterEffects - we can tilt camera below bi-layer, showing only DSRNA and its signalling affects Then lose the DSRNA as we pan farther down and see the signals affect the IRF-3
- TAN(IRF-3 proteins present in cell are awakened- start to vibrate slightly Camera pans down losing sight of the RNA (red squiggle)and the signals dissipate TheGreen IRF-3 proteins need a color shift so they standout against the cytoplasm - Lets go towards tan
- the Tan balls are vibrating slightly in the cytoplasm and - (blue balls approach Tan and attach)
- IRF-3 protiens phosphoralize by connecting with (dark blue) phosphates

Action

- 4 seconds RNA , red squiggle, moves into cytoplasm and sends off wave-like signal
- Tan balls vibrate// then this attract the blue balls(3 sec) then combo of Tan and blue balls decend in cytoplasim heading towards nucleus (4 secs)

camera

- Camera follows RNA as it decends into cell then continues to pan below RNA following the Tan and blue balls

Lighting

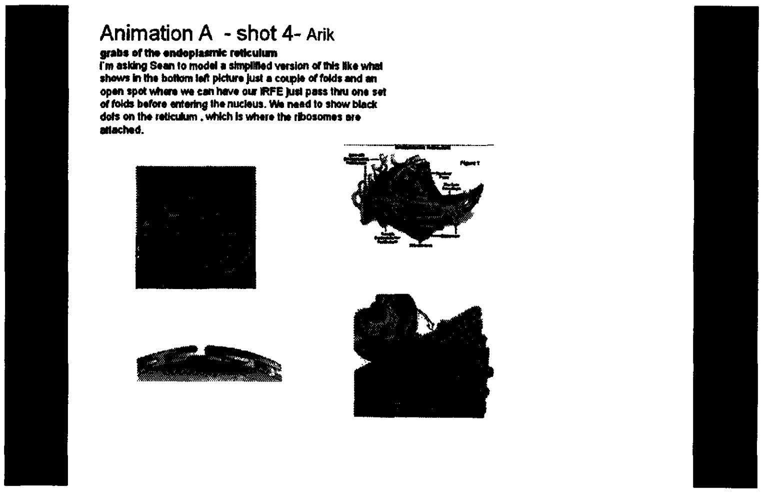
- external lighting so all "characters" are visible with cytoplasm as bkgd

voice over

[illegible]

DONE

models	assign	Finish due	bkgds	assign	Finish due	Animation/comp	assign	anim due	comp	assign	Finish due
DSRNA Tan IRF, blue phosph ate	Dennis Arik	DONE May 8	cytopla sm - light green color	Harry	060410 DONE	3 sec RNA signals Tan balls and they vibrate/ 8 sec - blue balls appear and attach to Tan and move down	Arik	June 15	Lighting, comp, render	ARIK	060615

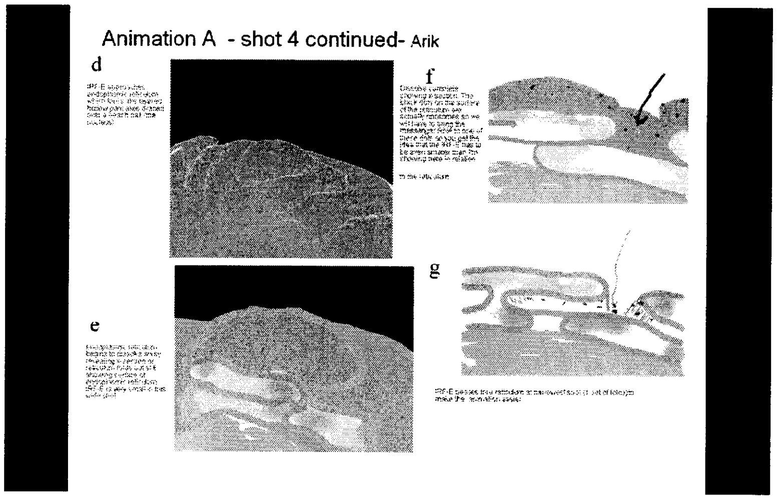


Animation A - shot 4- Arik

grabs of the endoplasmic reticulum

I'm asking Sean to model a simplified version of this like what shows in the bottom left picture just a couple of folds and an open spot where we can have our IRFE just pass thru one set of folds before entering the nucleus. We need to show black dots on the reticulum , which is where the ribosomes are attached.

Figure 8-3: Storyboards for Scene 1.



Animation A - shot 4 continued- Arik

d

[illegible]

e

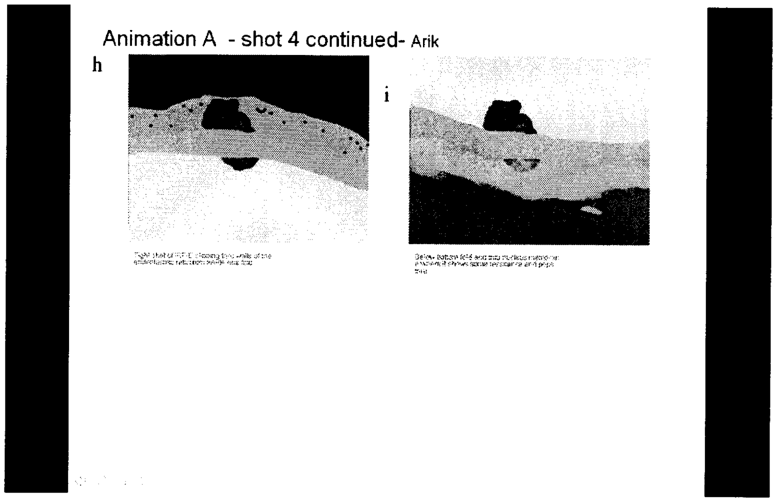
[illegible]

f

[illegible]

g

[illegible]



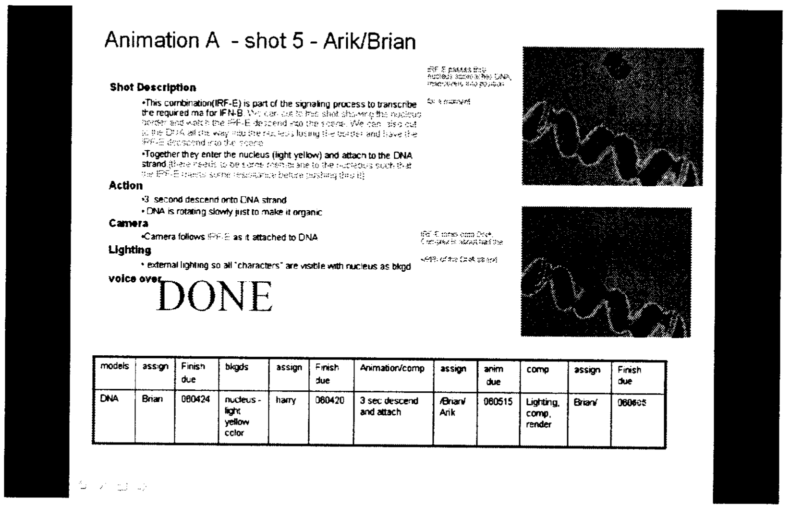
Animation A - shot 4 continued- Arik

h

[illegible]

i

[illegible]



Animation A - shot 5 - Arik/Brian

Shot Description

- This combination(IRF-E) is part of the signaling process to transcribe the required rna for IFN-B. [illegible]
- Together they enter the nucleus (light yellow) and attach to the DNA strand [illegible]

Action

- 3 second descend onto DNA strand
- DNA is rotating slowly just to make it organic

Camera

- Camera follows IRF-E as it attached to DNA

Lighting

- external lighting so all "characters" are visible with nucleus as bkgd

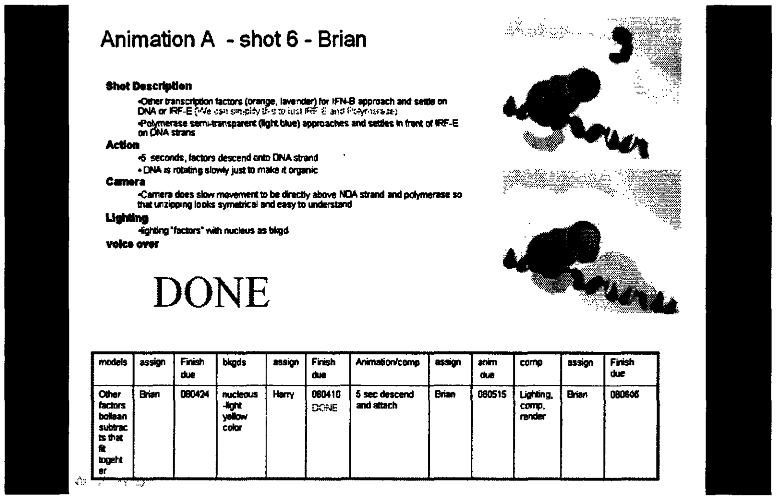
voice over

DONE

[illegible]

models	assign	Finish due	bkgds	assign	Finish due	Animation/comp	assign	anim due	comp	assign	Finish due
DNA	Brian	080424	nucleus - light yellow color	harry	080420	3 sec descend and attach	/Brian/ Arik	080515	Lighting, comp, render	Brian/	0806[illegible]

Figure 8-4: Storyboards for Scene 1.



Animation A - shot 6 - Brian

Shot Description

- Other transcription factors (orange, lavender) for IFN-B approach and settle on DNA or IRF-E (We can simplify B-s to just IRF E and Polymerase)
- Polymerase semi-transparent (light blue) approaches and settles in front of IRF-E on DNA strans

Action

- 5 seconds, factors descend onto DNA strand
- DNA is rotating slowly just to make it organic

Camera

- Camera does slow movement to be directly above NDA strand and polymerase so that unzipping looks symetrical and easy to understand

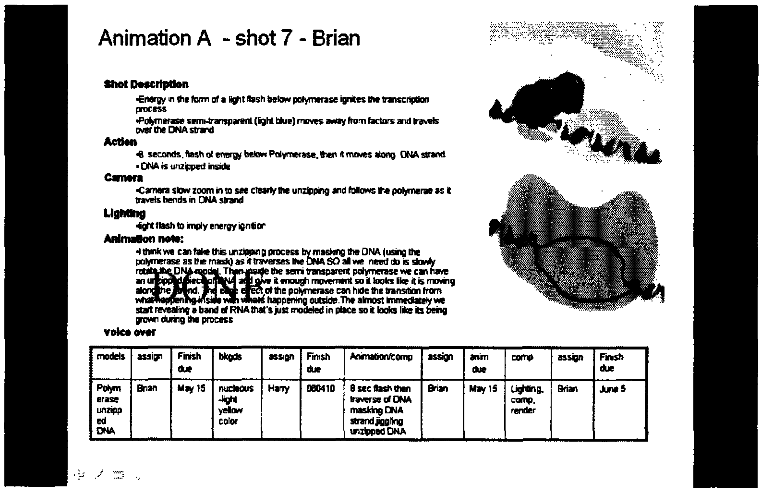
Lighting

- lighting "factors" with nucleus as bkgd

voice over

DONE

models	assign	Finish due	bkgds	assign	Finish due	Animation/comp	assign	anim due	comp	assign	Finish due
Other factors bollean subtracts that fit togehter	Brian	080424	nucleous -light yellow color	Harry	080410 DONE	5 sec descend and attach	Brian	080515	Lighting, comp, render	Brian	080606



Animation A - shot 7 - Brian

Shot Description

- Energy in the form of a light flash below polymerase ignites the transcription process
- Polymerase semi-transparent (light blue) moves away from factors and travels over the DNA strand

Action

- 8 seconds, flash of energy below Polymerase, then it moves along DNA strand
- DNA is unzipped inside

Camera

- Camera slow zoom in to see clearly the unzipping and follows the polymerae as it travels bends in DNA strand

Lighting

- light flash to imply energy ignition

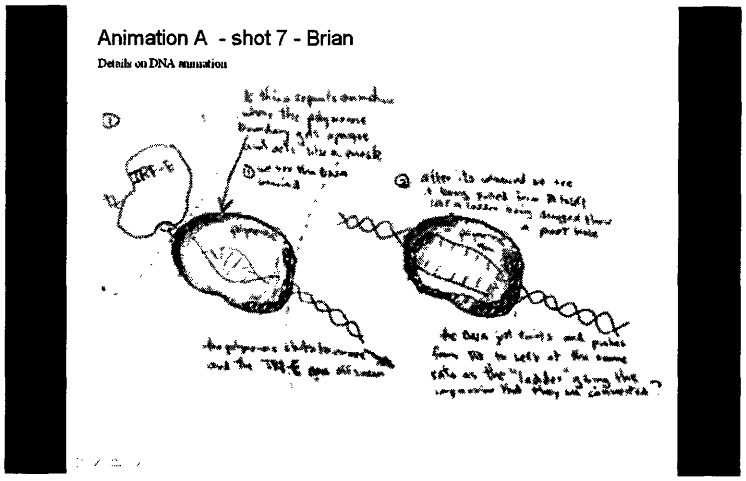
Animation note:

- I think we can fake this unzipping process by masking the DNA (using the polymerase as the mask) as it traverses the DNA SO all we need do is slowly rotate the DNA model. Then inside the semi transparent polymerase we can have an unzipped piece of DNA and give it enough movement so it looks like it is moving along the strand. The edge effect of the polymerase can hide the transition from what happening inside with whats happening outside. The almost immediately we start revealing a band of RNA that's just modeled in place so it looks like its being grown during the process

voice over

DONE

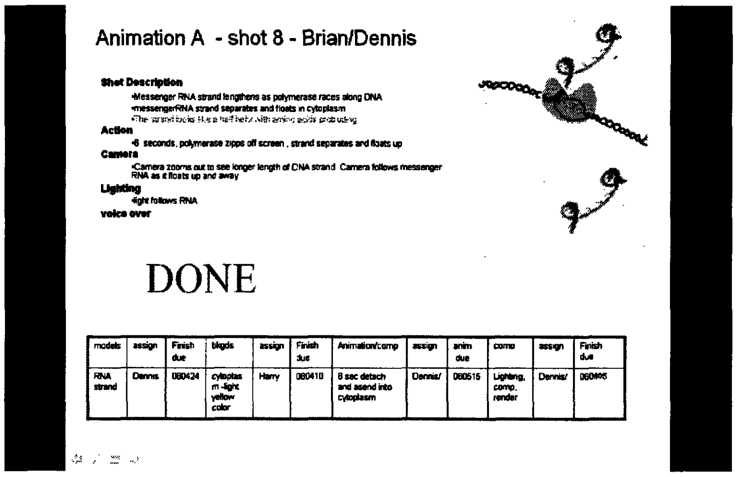
models	assign	Finish due	bkgds	assign	Finish due	Animation/comp	assign	anim due	comp	assign	Finish due
Polymerase unzipped DNA	Brian	May 15	nucleous -light yellow color	Harry	080410	8 sec flash then traverse of DNA masking DNA strand jiggling unzipped DNA	Brian	May 15	Lighting, comp, render	Brian	June 5



Animation A - shot 7 - Brian

Details on DNA animation

Figure 8-5: Storyboards for Scene 1.



Animation A - shot 8 - Brian/Dennis

Shot Description

- Messenger RNA strand lengthens as polymerase races along DNA
- messengerRNA strand separates and floats in cytoplasm
- The strand looks like a half helix with amino acids protruding

Action

- 6 seconds, polymerase zipps off screen , strand separates and floats up

Camera

- Camera zooms out to see longer length of DNA strand Camera follows messenger RNA as it floats up and away

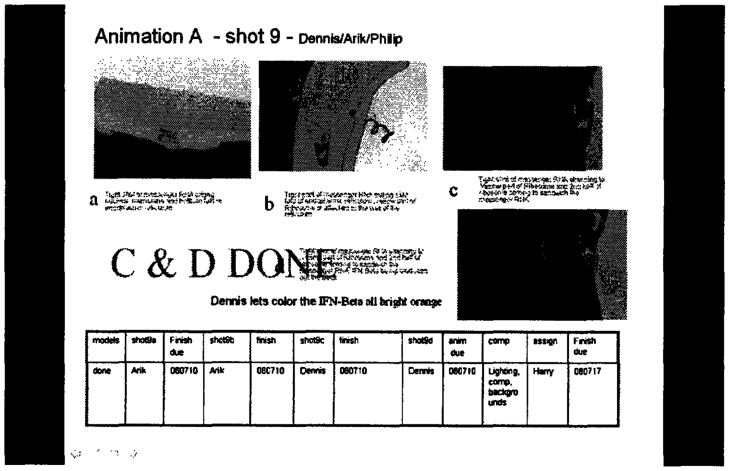
Lighting

- light follows RNA

voice over

DONE

models	assign	Finish due	bkgds	assign	Finish due	Animation/comp	assign	anim due	como	assign	Finish due
RNA strand	Dennis	080424	cytoplasm -light yellow color	Harry	080410	8 sec detach and asend into cytoplasm	Dennis/	080515	Lighting, comp, render	Dennis/	080605



Animation A - shot 9 - Dennis/Arik/Philip

a [illegible]

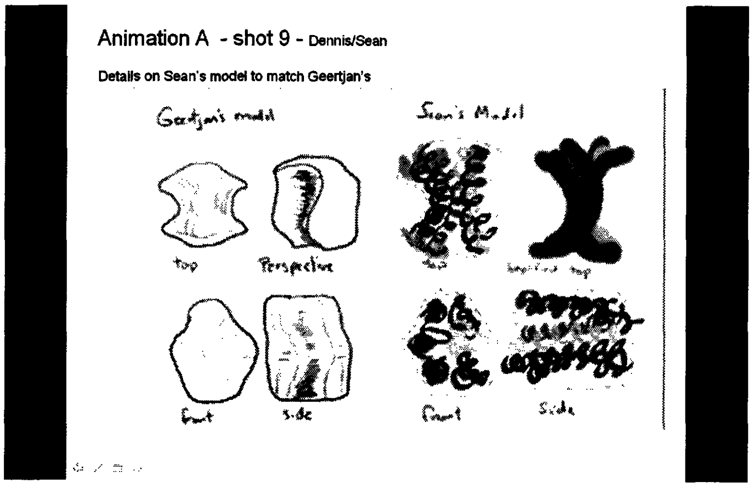
b [illegible]

c [illegible]

C & D DONE

Dennis lets color the IFN-Beta all bright orange

models	shot9a	Finish due	shot9b	finish	shot9c	finish	shot9d	anim due	comp	assign	Finish due
done	Arik	080710	Arik	080710	Dennis	080710	Dennis	080710	Lighting, comp, backgrounds	Harry	080717



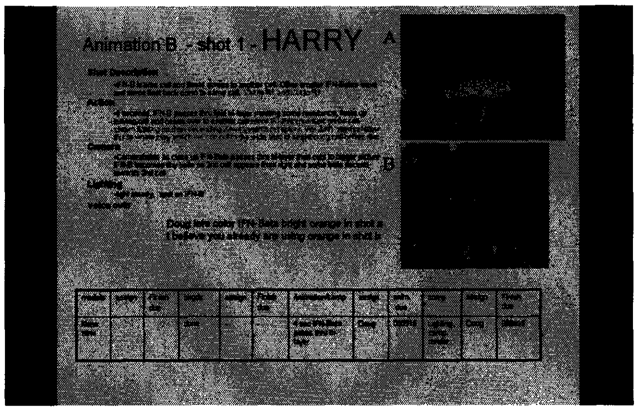
Animation A - shot 9 - Dennis/Sean

Details on Sean's model to match Geertjan's

Figure 8-6: Storyboards for Scene 1.

8.2 Scene 2 - IFN Signaling to IRF-7 Production

Scene 2 is the IFN-Beta signaling to produce IRF-7. Storyboards are provided in Figures 8-7 to Figures 8-11.



Animation B - shot 1 - HARRY

Shot Description

[illegible]

Action

[illegible]

Camera

[illegible]

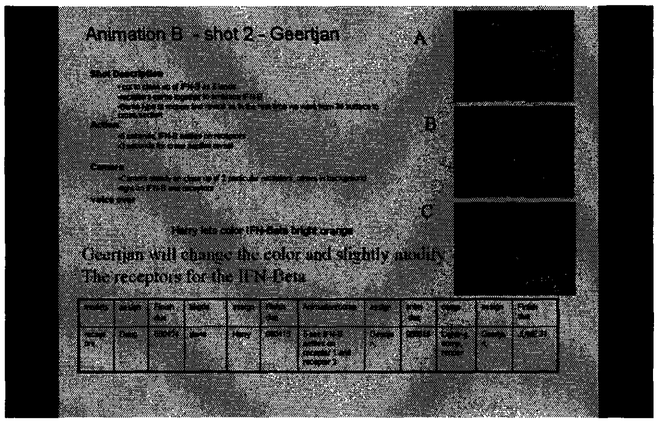
Lighting

[illegible]

Voice over

Doug lets color IFN-Beta bright orange in shot a

I believe you already are using orange in shot b



Animation B - shot 2 - Geertjan

Shot Description

[illegible]

Action

[illegible]

Camera

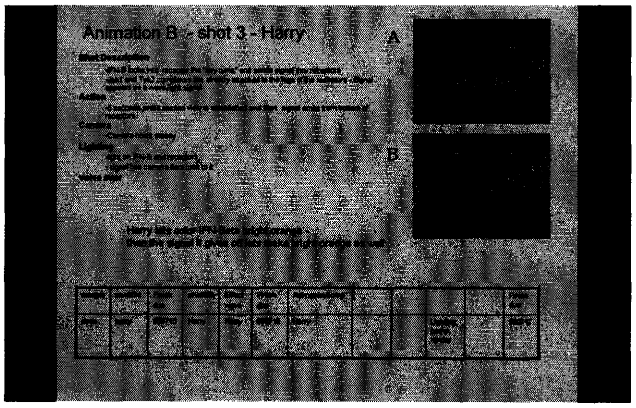
[illegible]

Voice over

Harry lets color IFN-Beta bright orange

Geertjan will change the color and slightly modify

The receptors for the IFN-Beta



Animation B - shot 3 - Harry

Shot Description

[illegible]

Action

[illegible]

Camera

[illegible]

Lighting

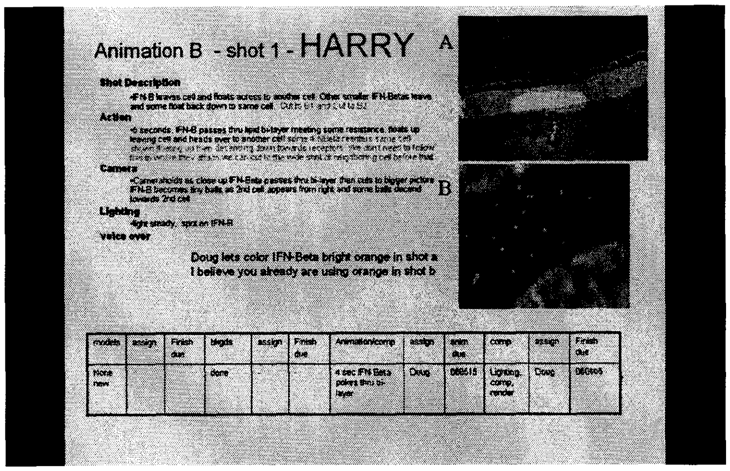
[illegible]

Voice over

Harry lets color IFN-Beta bright orange -

then the signal it gives off lets make bright orange as well

Figure 8-7: Storyboards for Scene 2.



Animation B - shot 1 - HARRY

Shot Description
•IFN-B leaves cell and floats across to another cell. Other smaller IFN-Betas leave and some float back down to same cell. [illegible]

Action
•6 seconds. IFN-B passes thru lipid bi-layer meeting some resistance, floats up leaving cell and heads over to another cell some [illegible] [illegible]

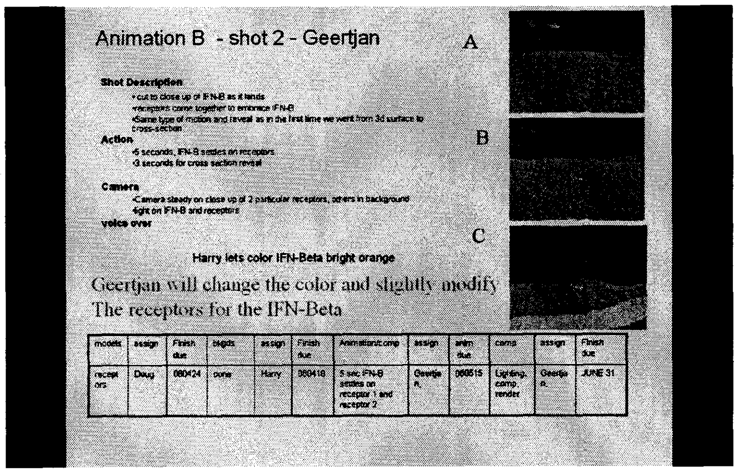
Camera
•Camera holds as close up IFN-Beta passes thru bi-layer then cuts to bigger picture IFN-B becomes tiny balls as 2nd cell appears from right and some balls decend towards 2nd cell

Lighting
•light steady, spot on IFN-B

voice over

Doug lets color IFN-Beta bright orange in shot a
I believe you already are using orange in shot b

models	assign	Finish due	bkgds	assign	Finish due	Animation/comp	assign	anim due	comp	assign	Finish due
None new			done	-	-	4 sec IFN Beta pokes thru bi-layer	Doug	060515	Lighting, comp, render	Doug	060605



Animation B - shot 2 - Geertjan

Shot Description
• cut to close up of IFN-B as it lands
•receptors come together to embrace IFN-B
•Same type of motion and reveal as in the first time we went from 3d surface to cross-section

Action
•5 seconds, IFN-B settles on receptors
•3 seconds for cross section reveal

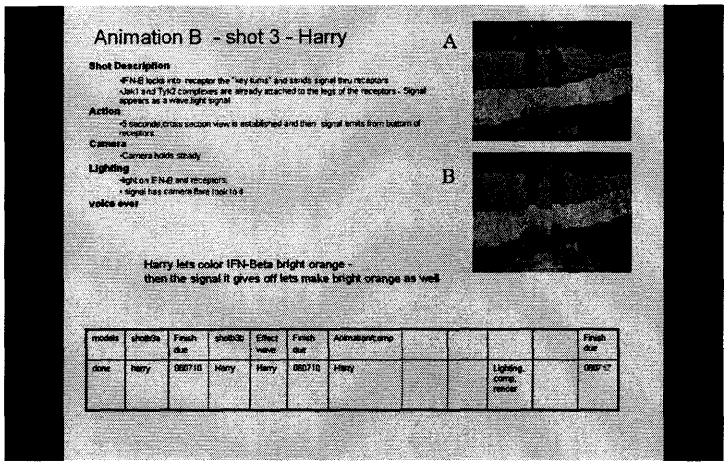
Camera
•Camera steady on close up of 2 particular receptors, others in background
•light on IFN-B and receptors

voice over

Harry lets color IFN-Beta bright orange

Geertjan will change the color and slightly modify
The receptors for the IFN-Beta

models	assign	Finish due	bkgds	assign	Finish due	Animation/comp	assign	anim due	comp	assign	Finish due
recept ors	Doug	060424	done	Harry	060418	5 sec IFN-B settles on receptor 1 and receptor 2	Geertje n.	060515	Lighting, comp, render	Geertje n.	JUNE 31



Animation B - shot 3 - Harry

Shot Description
•IFN-B locks into receptor the "key turns" and sends signal thru receptors
•Jak1 and Tyk2 complexes are already attached to the legs of the receptors - Signal appears as a wave light signal

Action
•5 seconds,cross section view is established and then signal emits from bottom of receptors

Camera
•Camera holds steady

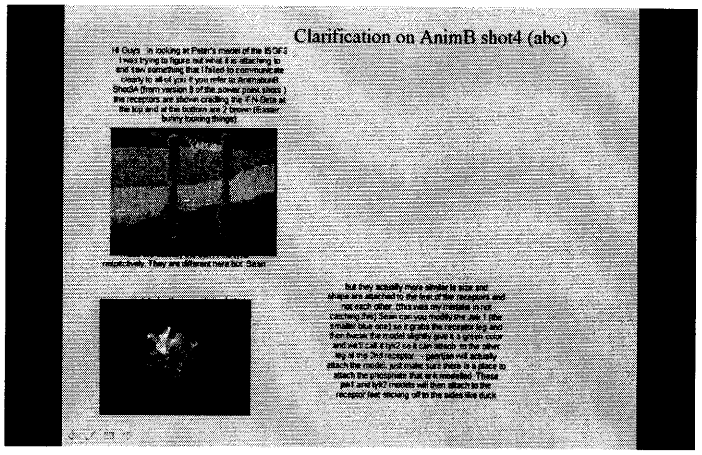
Lighting
•light on IFN-B and receptors
• signal has camera flare look to it

voice over

Harry lets color IFN-Beta bright orange -
then the signal it gives off lets make bright orange as well

models	shotb3a	Finish due	shotb3b	Effect wave	Finish due	Animation/comp					Finish due
done	harry	060718	Harry	Harry	060718	Harry			Lighting, comp, render		0607[illegible]

Figure 8-8: Storyboards for Scene 2.

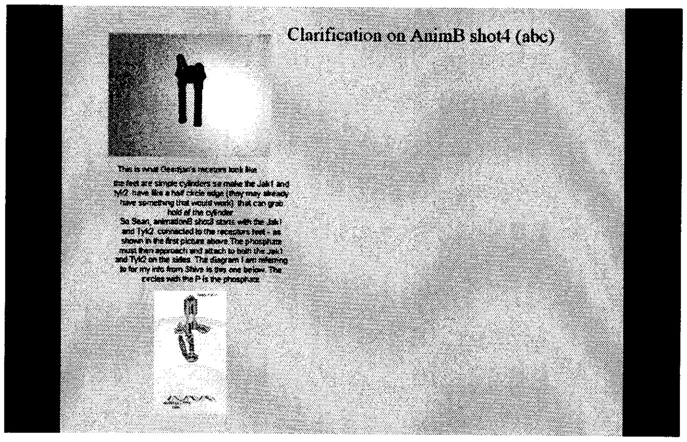


Clarification on AnimB shot4 (abc)

Hi Guys in looking at Peter's model of the ISGF3 I was trying to figure out what it is attaching to and saw something that I failed to communicate clearly to all of you. If you refer to AnimationB Shot3A (from version 8 of the power point shots) the receptors are shown cradling the IFN-Beta at the top and at the bottom are 2 brown (Easter bunny looking things)

respectively. They are different here but Sean

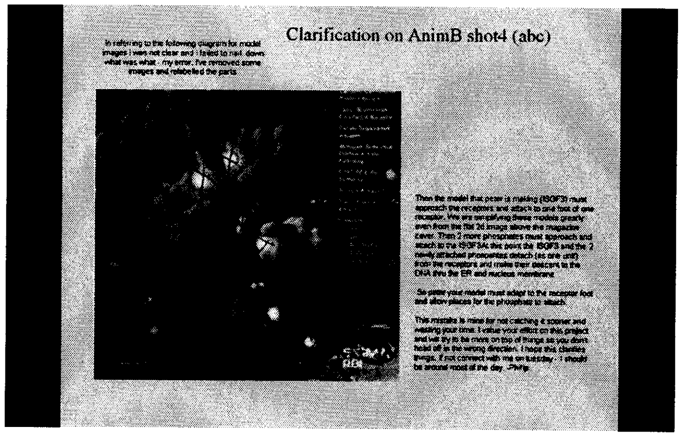
but they actually more similar in size and shape are attached to the feet of the receptors and not each other. (this was my mistake in not catching this) Sean can you modify the Jak 1 (the smaller blue one) so it grabs the receptor leg and then tweak the model slightly give it a green color and we'll call it tyk2 so it can attach to the other leg of the 2nd receptor. - georgian will actually attach the model, just make sure there is a place to attach the phosphate that erik modelled. These jak1 and tyk2 models will then attach to the receptor feet sticking off to the sides like duck



Clarification on AnimB shot4 (abc)

This is what Georgian's receptors look like

the feet are simple cylinders so make the Jak1 and Tyk2 have like a half circle edge (they may already have something that would work) that can grab hold of the cylinder
So Sean, animationB shot3 starts with the Jak1 and Tyk2 connected to the receptors feet - as shown in the first picture above. The phosphate must then approach and attach to both the Jak1 and TYK2 on the sides. The diagram I am referring to for my info from Shiva is this one below. The circles with the P is the phosphate



Clarification on AnimB shot4 (abc)

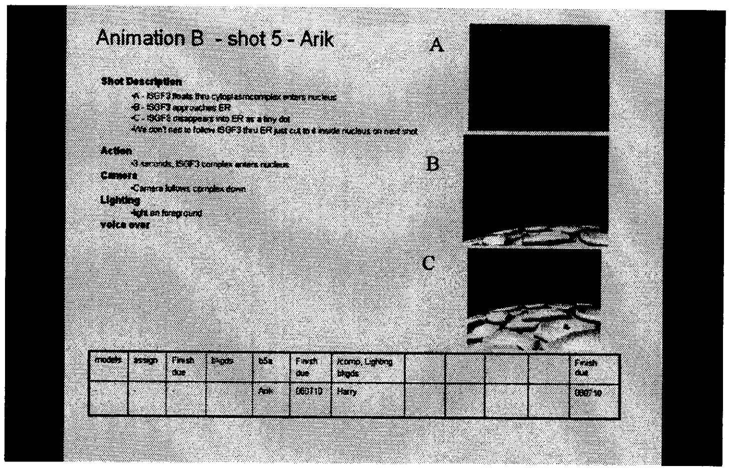
In referring to the following diagram for model images I was not clear and I failed to nail down what was what - my error. I've removed some images and relabelled the parts

Then the model that peter is making (ISGF3) must approach the receptors and attach to one foot of one receptor. We are simplifying these models greatly even from the flat 2d image above the magazine cover. Then 2 more phosphates must approach and attach to the ISGF3A. At this point the ISGF3 and the 2 newly attached phosphates detach (as one unit) from the receptors and make their descent to the DNA thru the ER and nucleus membrane.

So peter your model must adapt to the receptor foot and allow places for the phosphate to attach.

This mistake is mine for not catching it sooner and wasting your time. I value your effort on this project and will try to be more on top of things so you don't head off in the wrong direction. I hope this clarifies things, if not connect with me on tuesday - I should be around most of the day. -Philip

Figure 8-9: Storyboards for Scene 2.



Animation B - shot 5 - Arik

Shot Description
- •A - ISGF3 floats thru cytoplasm/complex enters nucleus
- •B - ISGF3 approaches ER
- •C - ISGF3 disappears into ER as a tiny dot
- •We don't ned to follow ISGF3 thru ER just cut to it inside nucleus on next shot

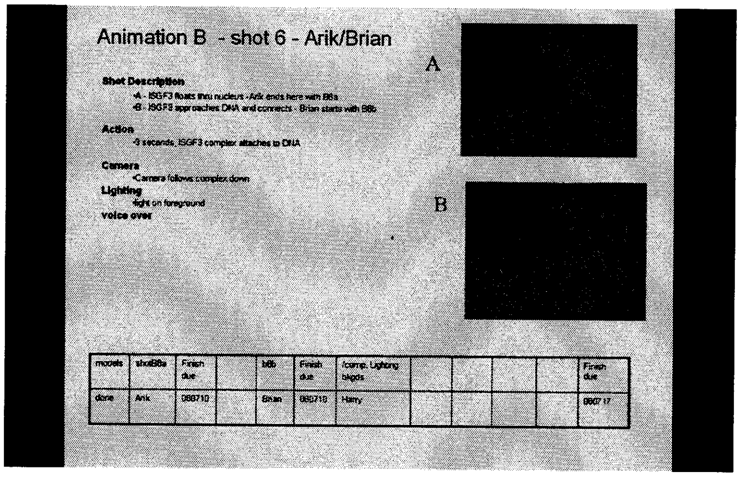
Action
- •3 seconds, ISGF3 complex enters nucleus

Camera
- •Camera follows complex down

Lighting
- •light on foreground

voice over

models	assign	Finish due	bkgds	b5a	Finish due	/comp, Lighting bkgds					Finish due
-	-	-		Arik	060710	Harry					060710



Animation B - shot 6 - Arik/Brian

Shot Description
- •A - ISGF3 floats thru nucleus - Arik ends here with B6a
- •B - ISGF3 approaches DNA and connects - Brian starts with B6b

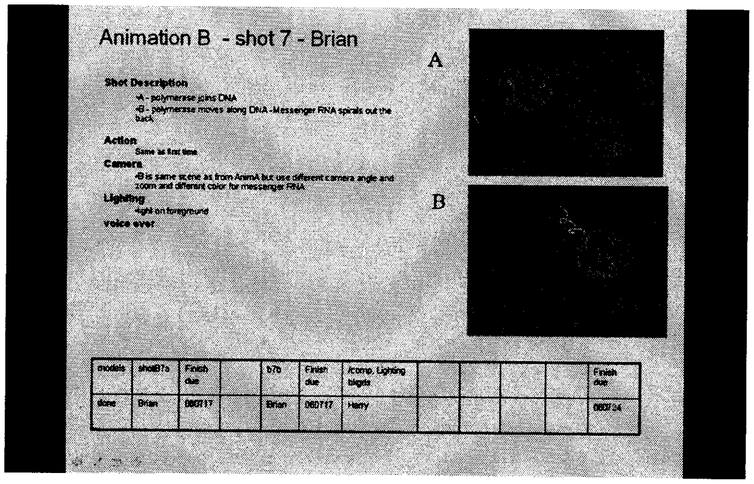
Action
- •3 seconds, ISGF3 complex attaches to DNA

Camera
- •Camera follows complex down

Lighting
- •light on foreground

voice over

models	shotB6a	Finish due		b6b	Finish due	/comp, Lighting bkgds					Finish due
done	Arik	060710		Brian	060710	Harry					060717



Animation B - shot 7 - Brian

Shot Description
- •A - polymerase joins DNA
- •B - polymerase moves along DNA - Messenger RNA spirals out the back

Action
- Same as first time

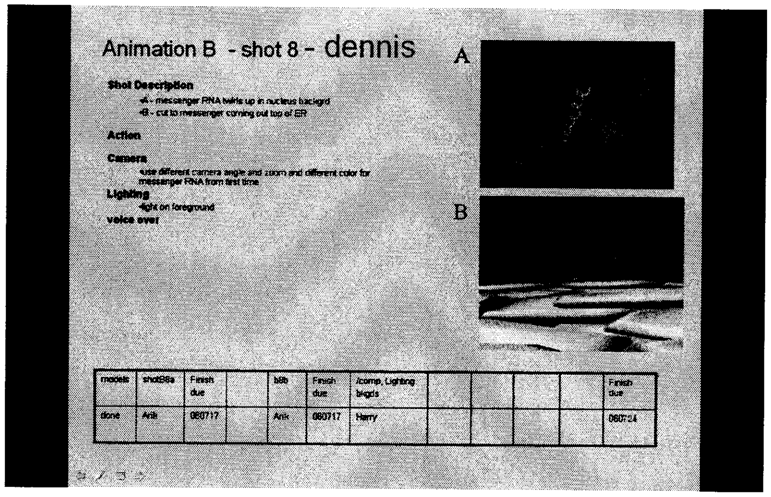
Camera
- •B is same scene as from AnimA but use different camera angle and zoom and different color for messenger RNA

Lighting
- •light on foreground

voice over

models	shotB7a	Finish due		b7b	Finish due	/comp, Lighting bkgds					Finish due
done	Brian	060717		Brian	060717	Harry					060724

Figure 8-10: Storyboards for Scene 2.



Animation B - shot 8 - dennis

Shot Description

- A - messenger RNA twirls up in nucleus backgrd
- B - cut to messenger coming out top of ER

Action

Camera

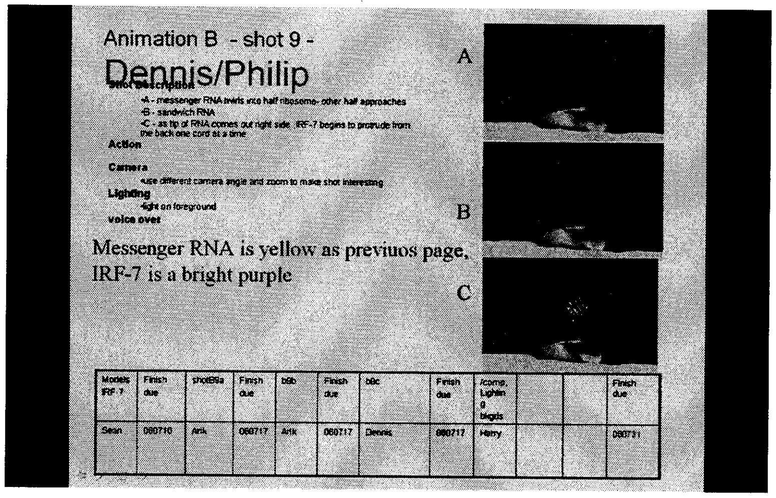
- use different camera angle and zoom and different color for messenger RNA from first time

Lighting

- light on foreground

voice over

models	shot8a	Finish due		b8b	Finish due	/comp, Lighting bkgds					Finish due
done	Arik	080717		Arik	080717	Harry					080724



Animation B - shot 9 - Dennis/Philip

Shot Description

- A - messenger RNA twirls into half ribosome- other half approaches
- B - sandwich RNA
- C - as tip of RNA comes out right side, IRF-7 begins to protrude from the back one cord at a time

Action

Camera

- use different camera angle and zoom to make shot interesting

Lighting

- light on foreground

voice over

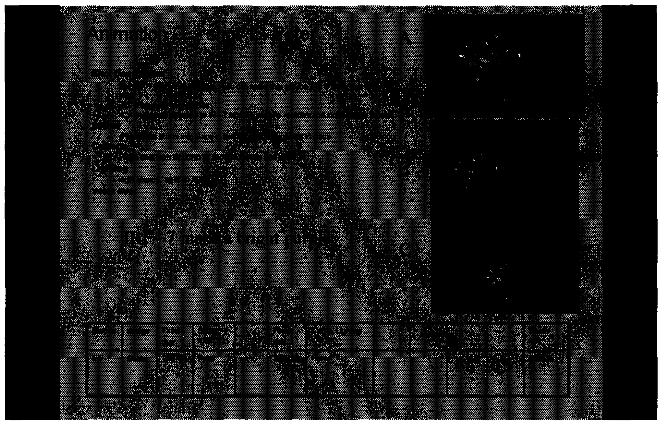
Messenger RNA is yellow as previuos page, IRF-7 is a bright purple

Models IRF-7	Finish due	shot9a	Finish due	b9b	Finish due	b9c	Finish due	/comp, Lightin g bkgds			Finish due
Sean	080710	Arik	080717	Arik	080717	Dennis	080717	Harry			080731

Figure 8-11: Storyboards for Scene 2.

8.3 Scene 3 - Up regulation of IFN-Alpha

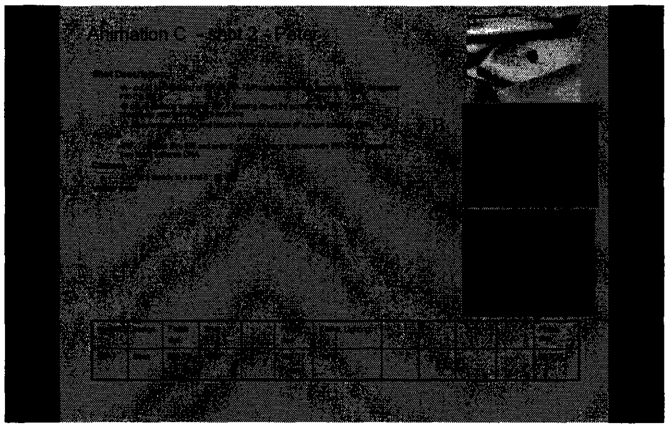
Scene 3 is the Up regulation of IFN-Alpha. Storyboards are provided in Figures 8-12 to Figures 8-13.


Animation C - Shot 1 - Peter

[illegible]

IRF-7 makes bright purple

[illegible]

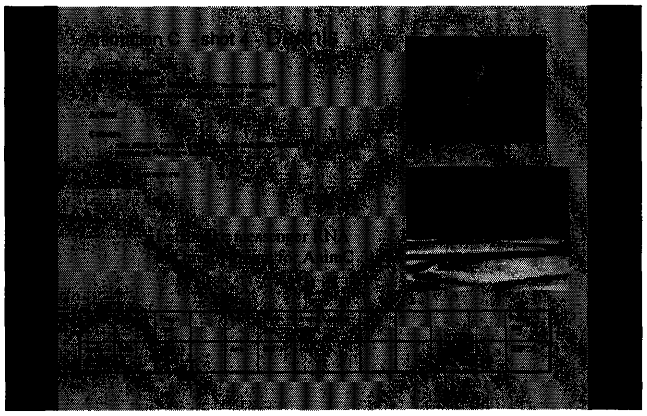

Animation C - Shot 2 - Peter

[illegible]

Animation C - Shot 3 - Brian

[illegible]

Figure 8-12: Storyboards for Scene 3.



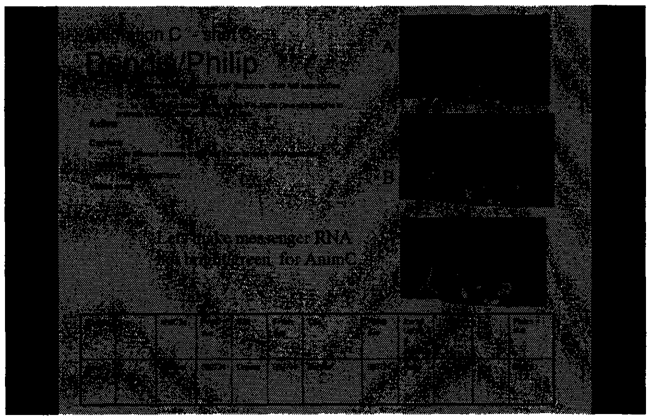
[illegible]ion C - shot 4 - Dennis

[illegible]

[illegible] messenger RNA

[illegible] for AnimC

[illegible]



[illegible]ion C - shot [illegible]

[illegible]/Philip

[illegible]

[illegible] make messenger RNA

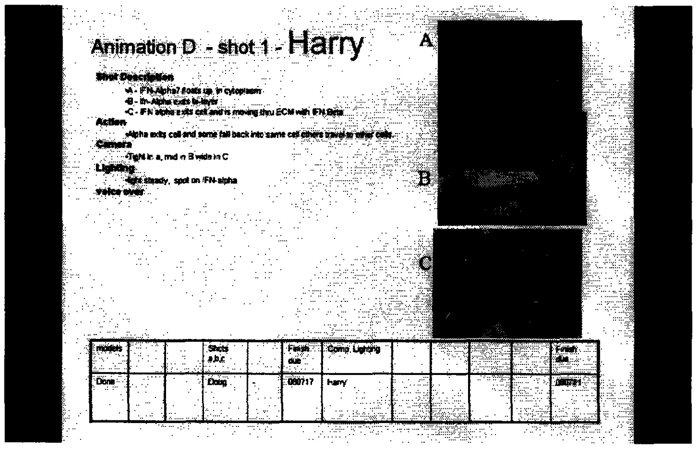
[illegible] for AnimC

[illegible]

Figure 8-13: Storyboards for Scene 3.

8.4 Scene 4 – Up regulation of IFN-Gamma

Scene 4 is the Up regulation of IFN-Gamma. As aforementioned, this scene is highly speculative but serves to show at a high-level how the up regulation of IFN-Gamma serves to provide a broad protection against viruses by IFN-Gamma serving to up-regulate numerous genes for protecting the cell from viral attack. Storyboards are provided in Figures 8-14 to Figures 8-18.



Animation D - shot 1 - Harry

Shot Description
- A - IFN-Alpha? floats up in cytoplasm
- B - In-Alpha exits bi-layer
- C - IFN alpha exits cell and is moving thru ECM with IFN Beta

Action
- Alpha exits cell and some fall back into same cell others travel to other cells.

Camera
- Tight in a, mid in B wide in C

Lighting
- light steady, spot on IFN-alpha

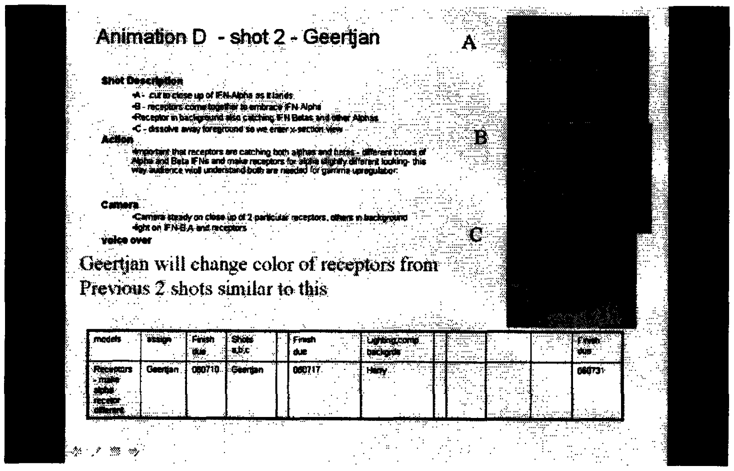
voice over

A

B

C

models			Shots a,b,c		Finish due	Comp. Lighting					Finish due
Done			Doug	-	060717	Harry					060731



Animation D - shot 2 - Geertjan

Shot Description
- A - cut to close up of IFN-Alpha as it lands
- B - receptors come together to embrace IFN-Alpha
- Receptor in background also catching IFN Betas and other Alphas
- C - dissolve away foreground so we enter x-section view

Action
- Important that receptors are catching both alphas and betas - different colors of Alpha and Beta IFNs and make receptors for alpha slightly different looking- this way audience will understand both are needed for gamma upregulator.

Camera
- Camera steady on close up of 2 particular receptors, others in background
- light on IFN-B,A and receptors

voice over

Geertjan will change color of receptors from Previous 2 shots similar to this

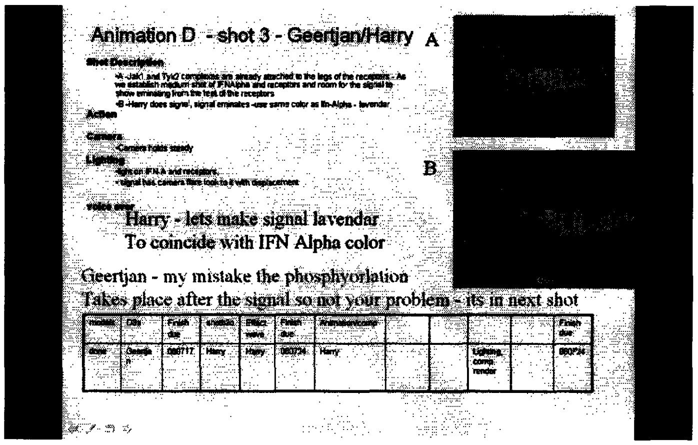
A

B

C

models	assign	Finish due	Shots a,b,c		Finish due	Lighting,comp backgrds					Finish due
Receptors - make alpha receptor different	Geertjan	060710	Geertjan		060717	Harry					060731

Figure 8-14: Storyboards for Scene 4.



Animation D - shot 3 - Geertjan/Harry

Shot Description

- A -Jak1 and Tyk2 complexes are already attached to the legs of the receptors - As we establish medium shot of IFNAlpha and receptors and room for the signal to show eminating from the feet of the receptors
- B -Harry does signal, signal eminates -use same color as Ifn-Alpha - lavendar

Action

Camera

- Camera holds steady

Lighting

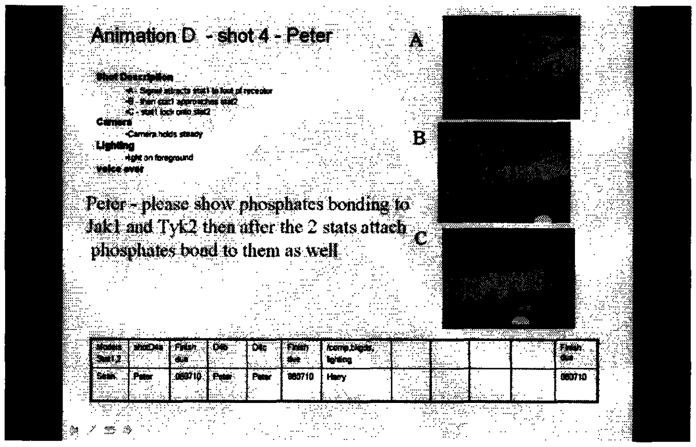
- light on IFN-A and receptors.
- signal has camera flare look to it with displacement

voice over

Harry - lets make signal lavendar
To coincide with IFN Alpha color

Geertjan - my mistake the phosphyorlation
Takes place after the signal so not your problem - its in next shot

models	D3a	Finish due	shot3b	Effect wave	Finish due	Animation/comp					Finish due
done	Geertjan	080717	Harry	Harry	080724	Harry			Lighting, comp, render		080724



Animation D - shot 4 - Peter

Shot Description

- A - Signal attracts stat1 to foot of receptor
- B - then stat1 approaches stat2
- C - stat1 lock onto stat2

Camera

- Camera holds steady

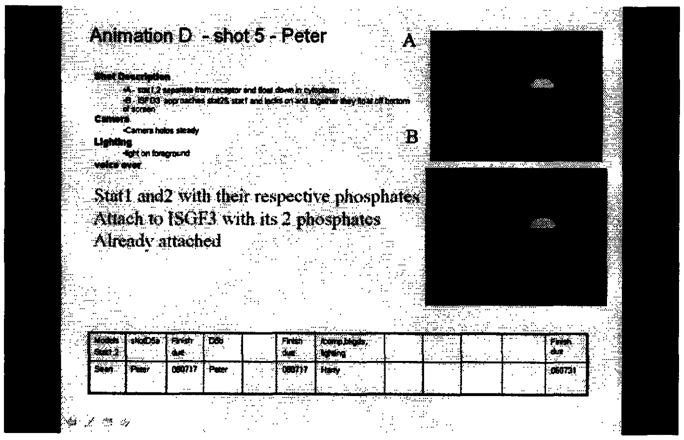
Lighting

- light on foreground

voice over

Peter - please show phosphates bonding to Jak1 and Tyk2 then after the 2 stats attach phosphates bond to them as well

Models Stat1,2	shotD4a	Finish due	D4b	D4c	Finish due	/comp,bkgds, lighting					Finish due
Sean	Peter	080710	Peter	Peter	080710	Harry					080710



Animation D - shot 5 - Peter

Shot Description

- A - stat1,2 separate from receptor and float down in cytoplasm
- B - ISFG3 approaches stat2& stat1 and locks on and together they float off bottom of screen

Camera

- Camera holds steady

Lighting

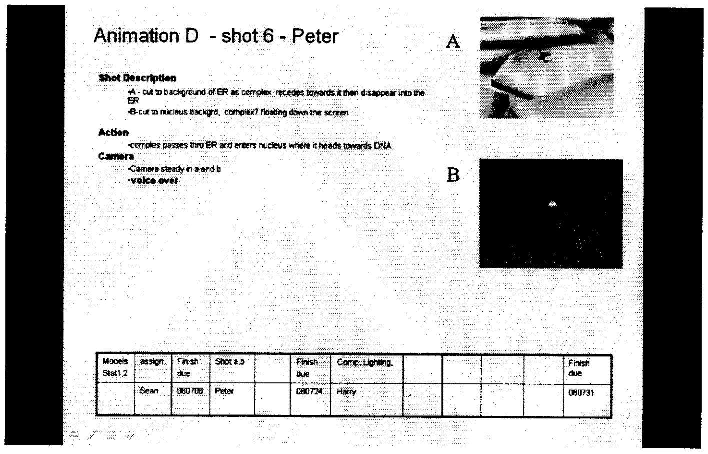
- light on foreground

voice over

Stat1 and2 with their respective phosphates
Attach to ISGF3 with its 2 phosphates
Already attached

Models Stat1,2	shotD5a	Finish due	D5b		Finish due	/comp,bkgds, lighting					Finish due
Sean	Peter	080717	Peter		080717	Harry					080731

Figure 8-15: Storyboards for Scene 4.



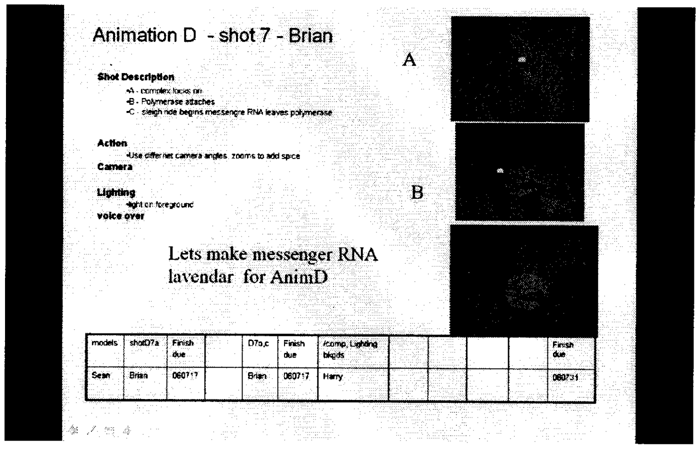
Animation D - shot 6 - Peter

Shot Description
- A - cut to background of ER as complex recedes towards it then disappear into the ER
- B-cut to nucleus backgrd, complex7 floating down the screen

Action
- comples passes thru ER and enters nucleus where it heads towards DNA

Camera
- Camera steady in a and b
- **voice over**

Models Stat1,2	assign	Finish due	Shot a,b		Finish due	Comp, Lighting,					Finish due
	Sean	080708	Peter		080724	Harry	.				080731



Animation D - shot 7 - Brian

Shot Description
- A - complex locks on
- B - Polymerase attaches
- C - sleigh ride begins messengre RNA leaves polymerase

Action
- Use differnet camera angles, zooms to add spice

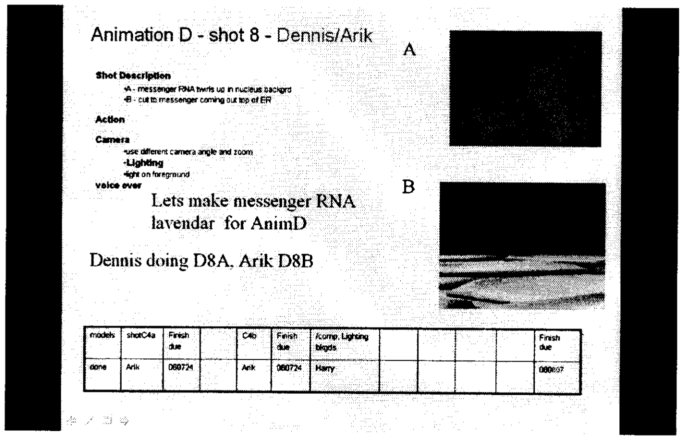
Camera

Lighting
- light on foreground

voice over

Lets make messenger RNA
lavendar for AnimD

models	shotD7a	Finish due		D7b,c	Finish due	/comp, Lighting bkgds					Finish due
Sean	Brian	060717		Brian	060717	Harry					080731



Animation D - shot 8 - Dennis/Arik

Shot Description
- A - messenger RNA twirls up in nucleus backgrd
- B - cut to messenger coming out top of ER

Action

Camera
- use different camera angle and zoom
- **Lighting**
- light on foreground

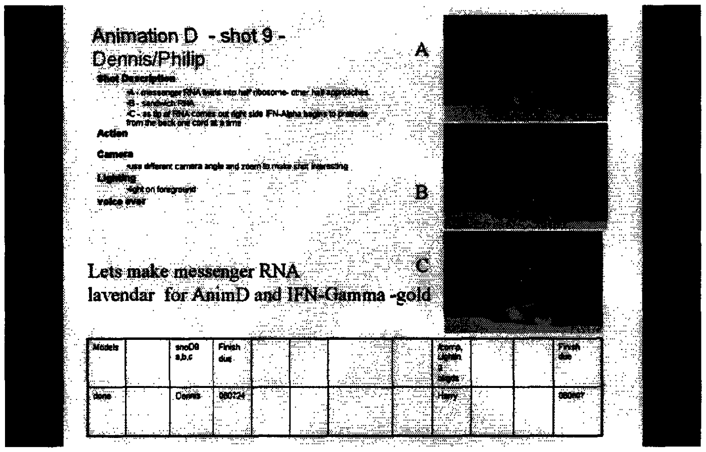
voice over

Lets make messenger RNA
lavendar for AnimD

Dennis doing D8A, Arik D8B

models	shotC4a	Finish due		C4b	Finish due	/comp, Lighting bkgds					Finish due
done	Arik	080724		Arik	080724	Harry					080807

Figure 8-16: Storyboards for Scene 4.


Animation D - shot 9 - Dennis/Philip

Shot Description
- A - messenger RNA feeds into half ribosome- other half approaches
- B - sandwich RNA
- C - as tip of RNA comes out right side IFN-Alpha begins to protrude from the back one cord at a time

Action

Camera
- use different camera angle and zoom to make shot interesting

Lighting
- light on foreground

voice over

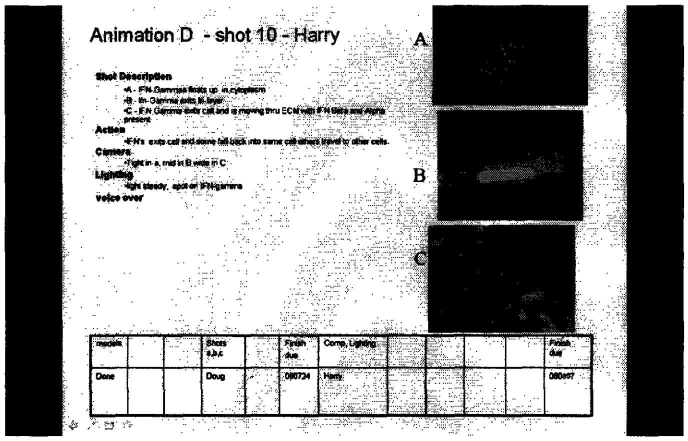
Lets make messenger RNA lavendar for AnimD and IFN-Gamma -gold

A

B

C

Models		snoD9 a,b,c	Finish due					comp, Lightin g bkgds			Finish due
done		Dennis	080724					Harry			080807


Animation D - shot 10 - Harry

Shot Description
- A - IFN-Gamma floats up in cytoplasm
- B - In-Gamma exits lli-layer
- C - IFN Gamma exits cell and is moving thru ECM with IFN Beta and Alpha present

Action
- IFN's exits cell and some fall back into same cell others travel to other cells.

Camera
- Tight in a, mid in B wide in C

Lighting
- light steady, spot on IFN-gamma

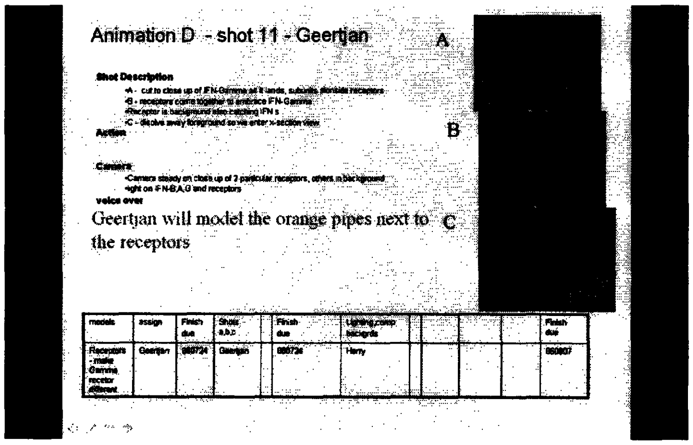
voice over

A

B

C

models			Shots a,b,c		Finish due	Comp, Lighting					Finish due
Done			Doug	-	080724	Harry					080807


Animation D - shot 11 - Geertjan

Shot Description
- A - cut to close up of IFN-Gamma as it lands, subunits outside receptors
- B - receptors come together to embrace IFN-Gamma
- Receptor is background also catching IFN s
- C - disolve away foreground so we enter x-section view

Action

Camera
- Camera steady on close up of 2 particular receptors, others in background
- light on IFN-B,A,G and receptors

voice over

Geertjan will model the orange pipes next to the receptors

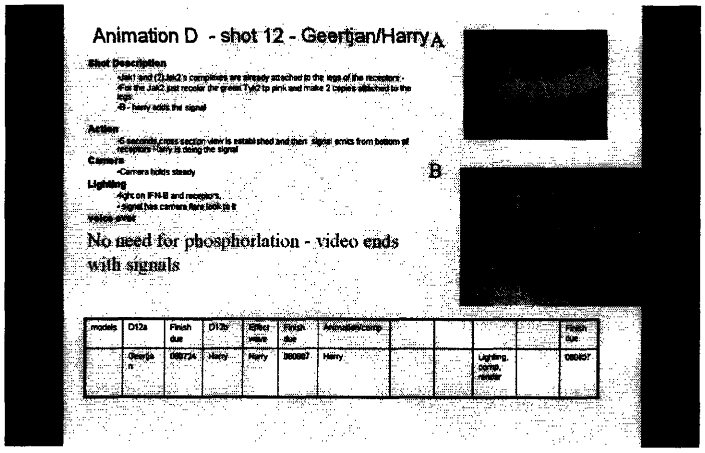
A

B

C

models	assign	Finish due	Shots a,b,c		Finish due	Lighting,comp backgrds					Finish due
Receptors - make Gamma recetor different	Geertjan	080724	Geertjan		080724	Harry					080807

Figure 8-17: Storyboards for Scene 4.



Animation D - shot 12 - Geertjan/Harry

A

B

Shot Description

- Jak1 and (2)Jak2's complexes are already attached to the legs of the receptors -
- For the Jak2 just recolor the green Tyk2 to pink and make 2 copies attached to the legs
- B - harry adds the signal

Action

- 5 seconds cross section view is established and then signal emits from bottom of receptors Harry is doing the signal

Camera

- Camera holds steady

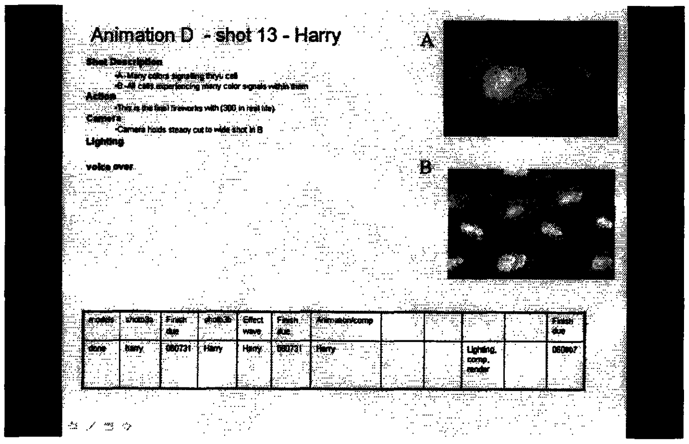
Lighting

- light on IFN-B and receptors,
- signal has camera flare look to it

voice over

No need for phosphorlation - video ends with signals

models	D12a	Finish due	D12b	Effect wave	Finish due	Animation/comp					Finish due
	Geertjan	080724	Harry	Harry	080807	Harry			Lighting, comp, render		080807



Animation D - shot 13 - Harry

A

B

Shot Description

- A - Many colors signaling thryu cell
- B - All cells experiencing many color signals within them

Action

- This is the final fireworks with (300 in real life)

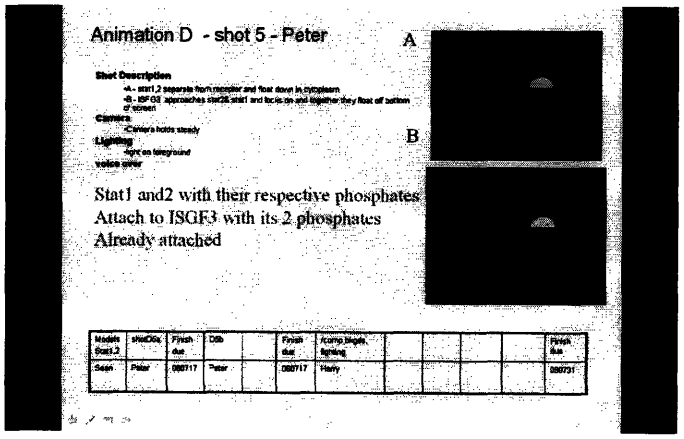
Camera

- Camera holds steady cut to wide shot in B

Lighting

voice over

models	shotD13a	Finish due	shotD13b	Effect wave	Finish due	Animation/comp					Finish due
done	harry	080731	Harry	Harry	080731	Harry			Lighting, comp, render		080807



Animation D - shot 5 - Peter

A

B

Shot Description

- A - stat1,2 separate from receptor and float down in cytoplasm
- B - ISFG3 approaches stat2& stat1 and locks on and together they float off bottom of screen

Camera

- Camera holds steady

Lighting

- light on foreground

voice over

Stat1 and2 with their respective phosphates Attach to ISGF3 with its 2 phosphates Already attached

Models Stat1,2	shotD5a	Finish due	D5b		Finish due	/comp bkgds, lighting					Finish due
Sean	Peter	080717	Peter		080717	Harry					080731

Figure 8-18: Storyboards for Scene 4.

8.5 Video Animation

This thesis also includes a full video of the scenes described above and is included herein with this thesis in a video DVD format. On the DVD Scene 1 can be found on the menu as Infection and IFN-Beta. Scene 2 can be found on the menu as IRF-7. Scene 3 can be found on the menu as IFN-Alpha. We did not include the SOCS1 regulation as that component was discovered after the making of the video. However, Scene 4 has a "bonus" conceptual scene of IFN-Gamma.

8.6 Summary

The video animation done for this thesis is an attempt to conceptually illustrate the various pathway mechanisms involved in the IFN response to virus infection. This video provides the key steps involved in this IFN system. We believe that videos such as these can be instructive in the classroom setting to convey complex pathway dynamics, currently represented using simple chalk and board or static diagrams using presentation graphics. At MIT, this video has been used in one of core systems biology courses BE440, a course which the author took himself in 2005 to illustrate the dynamics of the IFN pathway.

Students were asked in the lass to view the video and critique elements of the pathway demonstrate their understanding. We look forward to compiling this feedback and enhancing the video to produce more realistic depiction of the IFN response.

Chapter 9

Conclusions

The research objective of this thesis was to present a new architecture for integrating biological pathway models. We have met this objective by: 1. Defining and implementing an initial prototype of Cytosolve, a scalable computational architecture for integrating biological pathway models; 2. Testing the architecture and validating its accuracy by applying it to solve a known problem: the Kholodenko EGFR model; 3. Demonstrating the efficacy of the architecture by building an integrative model of the IFN response to viral infection by integrating multiple models created by different authors worldwide in varying formats, and 4. Presenting a new quantitative method for evaluating different architectures for integrating biological pathway models.

9.0 Key Findings

This research has resulted in the following key findings:

- First, using the new quantitative methodology for evaluating architectures developed in this, we have found that the distributed architecture provides the most optimal architecture for integrating biological pathway models, unanimously across the entire spectrum of stakeholders.
- Second, the initial literature review of experimental research substantiates the time scales predicted by the integrated model of IFN response, developed in this thesis. This integrative model of IFN immune response can now serve as a vehicle for further studies, alongside experimental research, to provide greater understanding of the IFN response mechanism.
- Third, the Cytosolve architecture has demonstrated through initial tests that it offers a far easier way to maintain an integrated model, in which various elements of that model may be constantly changing. This beneficial feature of Cytosolve, in addition to many others, makes it a scalable computational architecture for building modeling complex cellular functions, and eventually for modeling the whole cell.

9.1 Future Research

This thesis serves as a foundation for many new areas of research as itemized below:

- *Spatial scale variation.* In this thesis we have not considered changes in spatial scale. We believe that the architecture, based on its modular approach and support for multiple compartments, can support varying spatial scales. However, more testing will have to be performed to understand the computation times required to fully support such spatial variations.
- *Adaptive time stepping of the Controller.* Currently, all models are invoked using one constant time step, which is taken to be the fastest time step among the ensemble of models. This is not optimal, as some models may be varying slower than others. Additional effort is required to implement intelligent adaptive time stepping at the Controller level to observe the time scales of different models and invoke them only when necessary. Such an effort will result in improved computation time performance.
- Implementation and integration with emerging ontologies. The Cytosolve PID has support for integrating other ontologies such as MIRIAM; however, future research needs to be done to fully integrate

MIRIAM and other such ontologies. This effort will enable Cytosolve to support many more model formats with greater ease, leveraging standards that the systems biology community globally accepts.

- *Addition of new pathways to the IFN integrated model.* There is much scope for continued research in growing the current integrated IFN response model. This will involve finding new add-on models that affect the IFN system and then integrating those into the existing integrated model. Such additions will provide a mechanism for others to evaluate the performance of their own particular models when integrated into the IFN response system. In addition, linking the IFN integrated model to other systems such as TGF-Beta up regulation, for example, can provide insights and direction to support on going experimental research.
- *Web-enabled GUI.* One area of development will be to web-enable the current user interface so an individual scientist can more easily integrate their local model from their desktop to an integrated ensemble. The goal of this future research should be to open a formal public portal for worldwide use.
- *Video realism.* While a great deal of work was invested in creating the video, there are many elements that can be updated to the video to make it far more biologically realistic. Future efforts could include adjusting the relative spatial sizes of the different graphic objects (e.g. virus, membrane, etc.) to match biological reality. In addition, the time scales

are not well represented in the video. More accurate time scales would enable one to see the rate limiting steps in the pathway.

Appendices

A. WSDL Error Management

```
/**
 * ProcessCatalogServiceLocator.java
 *
 */

package pathwaySolver.ws;

public class ProcessCatalogServiceLocator extends org.apache.axis.client.Service
implements pathwaySolver.ws.ProcessCatalogService {

    public ProcessCatalogServiceLocator() {
    }


    public ProcessCatalogServiceLocator(org.apache.axis.EngineConfiguration config) {
        super(config);
    }

    public ProcessCatalogServiceLocator(java.lang.String wsdlLoc,
javax.xml.namespace.QName sName) throws javax.xml.rpc.ServiceException {
        super(wsdlLoc, sName);
    }

    // Use to get a proxy class for pathwaySolver
    private java.lang.String pathwaySolver_address =
"http://localhost:8080/axis/services/pathwaySolver";

    public java.lang.String getpathwaySolverAddress() {
        return pathwaySolver_address;
    }

    // The WSDD service name defaults to the port name.
    private java.lang.String pathwaySolverWSDDServiceName = "pathwaySolver";

    public java.lang.String getpathwaySolverWSDDServiceName() {
        return pathwaySolverWSDDServiceName;
    }

    public void setpathwaySolverWSDDServiceName(java.lang.String name) {
        pathwaySolverWSDDServiceName = name;
    }

    public pathwaySolver.ws.ProcessCatalog getpathwaySolver() throws
javax.xml.rpc.ServiceException {
       java.net.URL endpoint;
        try {
            endpoint = new java.net.URL(pathwaySolver_address);
        }
        catch (java.net.MalformedURLException e) {
            throw new javax.xml.rpc.ServiceException(e);
        }
        return getpathwaySolver(endpoint);
    }

    public pathwaySolver.ws.ProcessCatalog getpathwaySolver(java.net.URL
portAddress) throws javax.xml.rpc.ServiceException {
        try {
            pathwaySolver.ws.PathwaySolverSoapBindingStub _stub = new
pathwaySolver.ws.PathwaySolverSoapBindingStub(portAddress, this);
            _stub.setPortName(getpathwaySolverWSDDServiceName());
            return _stub;
        }
        catch (org.apache.axis.AxisFault e) {
            return null;
        }
    }

    public void setpathwaySolverEndpointAddress(java.lang.String address) {
        pathwaySolver_address = address;
    }

    /**
     * For the given interface, get the stub implementation.
     * If this service has no port for the given interface,
     * then ServiceException is thrown.
     */
    public java.rmi.Remote getPort(Class serviceEndpointInterface) throws
javax.xml.rpc.ServiceException {
        try {
            if
(pathwaySolver.ws.ProcessCatalog.class.isAssignableFrom(serviceEndpointInterface))
{
                pathwaySolver.ws.PathwaySolverSoapBindingStub _stub = new
pathwaySolver.ws.PathwaySolverSoapBindingStub(new
java.net.URL(pathwaySolver_address), this);
                _stub.setPortName(getpathwaySolverWSDDServiceName());
                return _stub;
            }
        }
        catch (java.lang.Throwable t) {
            throw new javax.xml.rpc.ServiceException(t);
        }
        throw new javax.xml.rpc.ServiceException("There is no stub implementation for
the interface:  " + (serviceEndpointInterface == null ? "null" :
serviceEndpointInterface.getName()));
    }

    /**
     * For the given interface, get the stub implementation.
     * If this service has no port for the given interface,
     * then ServiceException is thrown.
     */
    public java.rmi.Remote getPort(javax.xml.namespace.QName portName, Class
serviceEndpointInterface) throws javax.xml.rpc.ServiceException {
        if (portName == null) {
            return getPort(serviceEndpointInterface);
        }
        java.lang.String inputPortName = portName.getLocalPart();
        if ("pathwaySolver".equals(inputPortName)) {
            return getpathwaySolver();
        }
        else  {
            java.rmi.Remote _stub = getPort(serviceEndpointInterface);
            ((org.apache.axis.client.Stub) _stub).setPortName(portName);
            return _stub;
        }
    }

    public javax.xml.namespace.QName getServiceName() {
        return new javax.xml.namespace.QName("urn:pathwaySolver",
"ProcessCatalogService");
    }

    private java.util.HashSet ports = null;

    public java.util.Iterator getPorts() {
        if (ports == null) {
            ports = new java.util.HashSet();
            ports.add(new javax.xml.namespace.QName("urn:pathwaySolver",
"pathwaySolver"));
        }
        return ports.iterator();
    }

    /**
    * Set the endpoint address for the specified port name.
    */
    public void setEndpointAddress(java.lang.String portName, java.lang.String
address) throws javax.xml.rpc.ServiceException {

if ("pathwaySolver".equals(portName)) {
            setpathwaySolverEndpointAddress(address);
        }
        else
{ // Unknown Port Name
            throw new javax.xml.rpc.ServiceException(" Cannot set Endpoint Address for
Unknown Port" + portName);
        }
    }
    /**
    * Set the endpoint address for the specified port name.
    */
    public void setEndpointAddress(javax.xml.namespace.QName portName,
java.lang.String address) throws javax.xml.rpc.ServiceException {
        setEndpointAddress(portName.getLocalPart(), address);
    }

}
```

B. Implementation of SBML Solver

This Appendix provides details of the implementation of a distributed version of the SBML Solver.

Initializing the Solver

1) External computer sends 'createSolver' command to the WSDL web service. The 'createSolver' command has the parameters of model filename, amount of time that the model will be simulated, and the number of time steps that will be used to simulate the model. (Note that these last two parameters control how long each time step will be)

2) WSDL web service receives the 'createSolver' command, and creates an SolverController using the above parameters.

3) SolverController constructs an LSolver. Solver calls 'initSolver' in the Cytosolve_C code, which takes in the above parameters and loads the specified model, initializing everything that is needed for model simulation.

4) SolverController enters a wait loop that waits for an input file from the WSDL web service.

Simulating a single time step

1) External computer sends 'step' command. The 'step' command has the following parameters: species to change concentration values for, new concentration values for those species, and the species for which the external computer wants results.

2) WSDL web service receives 'step' command. The web service takes the parameters and writes them out to the input file that SolverController is waiting for. The WSDL web service then enters a wait loop, waiting for an output file.

3) The SolverController wait loop sees the new input file. It then opens the file, parses the parameters, and calls 'step' in Solver. The 'step' function in Solver then calls 'runSolver' in the Cytosolve_C code, which replaces species concentration values as specified, simulates over a single time step, then returns results back to the 'step' function in Solver. The 'step' function in SBMLSolver in turn returns the result to SolverController.

4) SBMLSolverController writes out results to the output file that the WSDL web service is waiting for. The SBMLSolverController also destroys its input file.

5) The WSDL web service sees the output file and parses the results. The results are returned to the external computer, and the output file is destroyed.

Destroying the Solver

1) External computer sends 'kill' command.

2) WSDL web service receives 'kill' command, writes out 'kill' to the input file that the SolverController is waiting for.

3) SolverController sees input command parses out 'kill'. It then calls Solver's 'cleanup' function, which calls 'destroySolver' in the Cytosolve_C code.

4) SolverController destroys input file, and stops execution.

C. Equations for IFN Solution Using Monolithic Approach

Reactions	Math
[virus]→[ssRNA]	$\frac{dRxn}{dt} = k_v \cdot [virus]$
[ssRNA] →[dsRNA]	$\frac{dRxn}{dt} = k_1 \cdot [ssRNA]$
[dsRNA] degradation	$\frac{dRxn}{dt} = k_2 \cdot [dsRNA]$
[VAK] activation	$\frac{dRxn}{dt} = k_3 \cdot [dsRNA]$
[VAK] degradation	$\frac{dRxn}{dt} = k_4 \cdot [VAK]$
[IRF-3]←→[IRF-3Pc]	$\frac{dRxn}{dt} = \frac{k_{f6} \cdot [IRF-3] \cdot [VAK]}{[IRF-3] + \frac{k_{r5} + k_{f6}}{k_{f5}}} - k_{r6} \cdot [IRF-3Pc]$
[IRF-3Pc]←→[IRF-3Pn]	$\frac{dRxn}{dt} = k_{f7} \cdot [IRF-3Pc] - k_{r7} \cdot [IRF-3Pn]$
[IFN-beta RNAn] production	$\frac{dRxn}{dt} = \frac{k_{8a} \cdot [IRF-3Pn]}{[IRF-3Pn] + k_{8b}} + \frac{k_{8a} \cdot [IRF-7Pn]}{[IRF-7Pn] + k_{8b}}$
[IRF-beta RNAn]←→[IFN-beta RNAc]	$\frac{dRxn}{dt} = k_9 \cdot [IFN-beta\_RNAn]$
[IFN-beta RNAc] degradation	$\frac{dRxn}{dt} = k_{10} \cdot [IFN-beta\_RNAc]$
[cIFN-beta] production	$\frac{\partial Rxn}{\partial t} = k_{11} \cdot [IFN-beta\_RNAc]$
[cIFN-beta] →[IFN-beta]	$\frac{dRxn}{dt} = k_{12} \cdot [cIFN-beta]$
[IFN-beta] degradation	$\frac{dRxn}{dt} = k_{32} \cdot [IFN-beta]$
[IFN-alpha RNAn] production	$\frac{dRxn}{dt} = \frac{k_{8a} \cdot [IRF-7Pn]}{[IRF-7Pn] + k_{8b}}$

Figure AC-1 – Differential equations for integrated monolithic approach.

[IFN-alpha RNAn]⟷[IFN-alpha RNAc]	$\frac{dRxn}{dt} = k_9 \cdot [IFN-alpha\_RNAn]$
[IFN-alpha RNAc] degradation	$\frac{dRxn}{dt} = k_{10} \cdot [IFN-alpha\_RNAc]$
[cIFN-alpha] production	$\frac{dRxn}{dt} = k_{11} \cdot [IFN-alpha\_RNAc]$
[cIFN-alpha] →[IFN-alpha]	$\frac{\partial Rxn}{\partial t} = k_{12} \cdot [cIFN-alpha]$
[IFN-alpha] degradation	$\frac{dRxn}{dt} = k_{32} \cdot [IFN-alpha]$
[JAK] & [IFNAR] association	$\frac{dRxn}{dt} = k_{,28} \cdot [JAK] \cdot [IFNAR] - k_{,28} \cdot [IFNARJ]$
[IFNARJ] & [IFN-alpha] association	$\frac{dRxn}{dt} = k_{,13} \cdot [IFN-alpha] \cdot [IFNARJ] - k_{,13} \cdot [IFN-alpha-bound]$
[IFNARJ] & [IFN-beta] association	$\frac{dRxn}{dt} = k_{,13} \cdot [IFN-beta] \cdot [IFNARJ] - k_{,13} \cdot [IFN-beta-bound]$
[IFN-alpha-bound] dimerization	$\frac{dRxn}{dt} = k_{,37} \cdot [IFN-alpha-bound]^2$ $- k_{,37} \cdot [IFN-alpha-bound\_2]$
[IFN-beta-bound] dimerization	$\frac{dRxn}{dt} = k_{,37} \cdot [IFN-beta-bound]^2 - k_{,37} \cdot [IFN-beta-bound\_2]$
[IFN-alpha-bound] phosphorylation	$\frac{dRxn}{dt} = k_{14} \cdot [IFN-alpha-bound\_2]$
[IFN-beta-bound] phosphorylation	$\frac{dRxn}{dt} = k_{14} \cdot [IFN-beta-bound\_2]$
[IFNAR2*] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{20} \cdot [IFNAR2^*] \cdot [SHP2]}{[IFNAR2^*] + \frac{k_{,19} + k_{20}}{k_{,19}}}$
[IFNBR2*] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{20} \cdot [IFNBR2^*] \cdot [SHP2]}{[IFNBR2^*] + \frac{k_{,19} + k_{20}}{k_{,19}}}$

Figure AC-2 – Differential equations for integrated monolithic approach.

[IFNAR2*] & [STATc*] association	$\frac{dRxn}{dt} = k_{f17} \cdot [IFNAR2^*] \cdot [STATc^*] - k_{r17} \cdot [IFNAR2^* STATc]$
[IFNBR2*] & [STATc*] association	$\frac{dRxn}{dt} = k_{f17} \cdot [IFNBR2^*] \cdot [STATc^*] - k_{r17} \cdot [IFNBR2^* STATc]$
[STATc] phosphorylation	$\frac{dRxn}{dt} = \frac{k_{14} \cdot ([IFNAR2^*] + [IFNBR2^*]) \cdot [STATc]}{[STATc] + \frac{k_{r13} + k_{14}}{k_{f13}}}$
[STATc*] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{22} \cdot [PPX] \cdot [STATc^*]}{[STATc^*] + \frac{k_{r21} + k_{22}}{k_{f21}}}$
[STATc] & [STATc*] association	$\frac{dRxn}{dt} = k_{f23} \cdot [STATc] \cdot [STATc^*] - k_{r23} \cdot [STATc^* - STATc]$
[STATc*] dimerization	$\frac{dRxn}{dt} = k_{f18} \cdot [STATc^*]^2 - k_{r18} \cdot [STATc^* 2]$
[STATc*2] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{22} \cdot [PPX] \cdot [STATc^* 2]}{[STATc^* 2] + \frac{k_{r21} + k_{22}}{k_{f21}}}$
[STATc*2] transport to nucleus	$\frac{dRxn}{dt} = k_{24} \cdot [STATc^* 2]$
[STATn*-STATn] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{26} \cdot [PPN] \cdot [STATn^* 2]}{[STATn^* 2] + \frac{k_{r25} + k_{26}}{k_{f25}}}$
[STATn*2] dissociation	$\frac{dRxn}{dt} = k_{r18} \cdot [STATn^* 2] - k_{f18} \cdot [STATn^*] \cdot [STATn^*]$
[STATn*] & [STATn] association	$\frac{dRxn}{dt} = k_{f23} \cdot [STATn] \cdot [STATn^*] - k_{r23} \cdot [STATn^* - STATn]$
[STATn*] dephosphorylation	$\frac{dRxn}{dt} = \frac{k_{26} \cdot [PPN] \cdot [STATn^*]}{[STATn^*] + \frac{k_{r25} + k_{26}}{k_{f25}}}$
[STATn] export to cytosol	$\frac{dRxn}{dt} = k_{27} \cdot [STATn]$
[STATn*-STATn-IRF9n] dissociation	$\frac{dRxn}{dt} = k_{r31} \cdot [STATn^* - STATn - IRF9n]$ $- k_{f31} \cdot [STATn^* - STATn] \cdot [nIRF - 9]$

Figure AC-3 – Differential equations for integrated monolithic approach.

[nIRF-9] export to cytosol	$\frac{dRxn}{dt} = k_{38} \cdot [nIRF-9]$
[cIRF-9] and [STATc*2] association	$\frac{dRxn}{dt} = k_{f31} \cdot [STATc*2] \cdot [cIRF-9] - k_{r31} \cdot [STATc*2-IRF9c]$
[STATc*2-IRF9c] transport to nucleus	$\frac{dRxn}{dt} = k_{29} \cdot [STATc*2-IRF9c]$
[STATn*2-IRF9n] dephosphotylation	$\frac{dRxn}{dt} = \frac{k_{26} \cdot [PPN] \cdot [STATn*2-IRF9n]}{[STATn*2-IRF9n] + \frac{k_{r25}+k_{26}}{k_{f25}}}$
[IRF-7 RNAn] production	$\frac{dRxn}{dt} = \frac{k_{8a} \cdot [STATn*2-IRF9n]}{[STATn*2-IRF9n] + k_{8b}}$
[IRF-7 RNAn]→[IRF-7 RNAc]	$\frac{dRxn}{dt} = k_9 \cdot [IRF-7\_RNAn]$
[IRF-7 RNAc] degradation	$\frac{dRxn}{dt} = k_{10} \cdot [IRF-7\_RNAc]$
[IRF-7Pc] production	$\frac{dRxn}{dt} = k_{11} \cdot [IRF-7\_RNAc]$
[IRF-7Pc] degradation	$\frac{dRxn}{dt} = k_4 \cdot [IRF-7\_Pc]$
[IRF-7Pc]⟷[IRF-7Pn]	$\frac{dRxn}{dt} = k_{f7} \cdot [IRF-7Pc] \cdot -k_{r7} \cdot [IRF7-Pn]$
[SOCS1 mRNAn] production	$\frac{dRxn}{dt} = \frac{k_{8a} \cdot [SOCS1\_mRNAn]}{[SOCS1\_mRNAn] + k_{8b}}$
[SOCS1 mRNAn] export to cytosol	$\frac{dRxn}{dt} = k_9 \cdot [SOCS1\_mRNAn]$
[SOCS1 mRNAc] degradation	$\frac{dRxn}{dt} = k_{10} \cdot [SOCS1\_mRNAc]$
[SOCS1] protein production	$\frac{dRxn}{dt} = k_{11} \cdot [SOCS1\_mRNAc]$
[SOCS1] degradation	$\frac{dRxn}{dt} = k_4 \cdot [SOCS1]$
[SOCS1] & [IFNAR2*] association	$\frac{dRxn}{dt} = k_{f34} \cdot [IFNAR2^*] \cdot [SOCS1] - k_{r34} \cdot [IFNAR2*-SOCS1]$

Figure AC-4 – Differential equations for integrated monolithic approach.

Bibliography

Aderem, A. (2005) Systems biology: its practice and challenges, *Cell*, **121**, 511-513.

Akarsu, E., Fox, F., Furmanski, W., Haupt. T. (1998) WebFlow-high-level programming environment and visual authoring toolkit for high performance distributed
computing. . *Proceedings of Supercomputing '98: High Performance Networking and Computing.* IEEE Computer Society, 1-7.

Alon, U. (2003) Biological networks: the tinkerer as an engineer, *Science*, **301**, 1866-1867.

Alvarez-Vasquez, F., Sims, K.J., Hannun, Y.A. and Voit, E.O. (2004) Integration of kinetic information on yeast sphingolipid metabolism in dynamical pathway models, *J Theor Biol*, **226**, 265-291.

Andersen, D.H. (1983) *Compartmental Modeling and Tracer Kinetics.* Springer, Berlin.

Arkin, A.P. and Fletcher, D.A. (2006) Fast, cheap and somewhat in control, *Genome Biol*, **7**, 114.

Asthagiri, A.R. and Lauffenburger, D.A. (2000) Bioengineering models of cell signaling, *Annu Rev Biomed Eng*, **2**, 31-53.

Bader, J.S. and Chant, J. (2006) Systems biology. When proteomes collide, *Science*, **311**, 187-188.

Baitaluk, M., Qian, X., Godbole, S., Raval, A., Ray, A. and Gupta, A. (2006) PathSys: integrating molecular interaction graphs for systems biology, *BMC Bioinformatics*, **7**, 55.

Bassingthwaighte, J.B., Chizeck, H.J., Atlas, L.E. and Qian, H. (2005) Multiscale modeling of cardiac cellular energetics, *Ann N Y Acad Sci*, **1047**, 395-424.

Bhalla, U.S. (2003) Understanding complex signaling networks through models and metaphors, *Prog Biophys Mol Biol*, **81**, 45-65.

Bocharaov (1994) Mathematical model of antiviral immune response III. Influenza A virus infection., *Journal of Theoretical Biology*, **167**, 323-359.

Brooks, F. (1975) *The mythical man month: essays in software engineering.*

Bulatwicz, T.F. (2006) SUPPORT FOR MODEL COUPLING: AN INTERFACE-BASED APPROACH. *Department of Computer and Information Science*. University of Oregon, 216.

Campagne, F., Neves, S., Chang, C.W., Skrabanek, L., Ram, P.T., Iyengar, R. and Weinstein, H. (2004) Quantitative information management for the biochemical computation of cellular networks, *Sci STKE*, **2004**, pl11.

Cannon, W.B. (1933) *The Wisdom of the Body*. Norton, New York.

Cella (1999) Maturation, Activation, and Protection of Dendritic Cells IInduced by Double-stranded RNA, *Journal of Experimental Medicine*, **189**, 821-829.

Cerami, E.G., Bader, G.D., Gross, B.E. and Sander, C. (2006) cPath: open source software for collecting, storing, and querying biological pathways, *BMC Bioinformatics*, **7**, 497.

Clements (2007) An Economic Model for Software Architecture Decisions. In IEEE (ed), *First International Conference on Economics of Software and Computation.*

Cooley, M. (1987) CYTOKINE ACTIVITY AFTER HUMAN BONE MARROW TRANSPLANTATION: Production of Interferons by Peripheral Blood Mononuclear Cells from Recipients of HLA-Identical Sibling Bone Marrow Transplants, *Journal of Immunology*, **138**.

Cuellar, A.A., Lloyd, C.M., Nielsen, P.F., Bullivant, D.P., Nickerson, D.P., Hunter, P.J. (2003) An Overview of CellML 1.1, a Biological Model Description Language, *SIMULATION*, **79**, 740-747.

Dabous, F.T. (2005) Estimating pattern consequences for the architectural design of e-business applications. *7th International Conference on Enterprise Information Systems* ICIES, Miami, USA, 248-254.

Davidson, M.W. (2007) Eukaryotic Animal Cell. Molecular Expressions, Tallahassee.

Dewey, C.F. (2006) Personal communication. In Ayyadurai, S. (ed). Cambridge, Personal communication.

Dhar, P., Meng, T.C., Somani, S., Ye, L., Sairam, A., Chitre, M., Hao, Z. and Sakharkar, K. (2004) Cellware--a multi-algorithmic software for computational systems biology, *Bioinformatics*, **20**, 1319-1321.

Duarte, N.C., Becker, S.A., Jamshidi, N., Thiele, I., Mo, M.L., Vo, T.D., Srivas, R. and Palsson, B.O. (2007) Global reconstruction of the human metabolic network based on genomic and bibliomic data, *Proc Natl Acad Sci U S A*, **104**, 1777-1782.

Endy, D. and Brent, R. (2001) Modelling cellular behaviour, *Nature*, **409**, 391-395.

Gianchandani, E.P., Brautigan, D.L. and Papin, J.A. (2006) Systems analyses characterize integrated functions of biochemical networks, *Trends in Biochemical Sciences*, **31**, 284-291.

Gilbert, D., Fuss, H., Gu, X., Orton, R., Robinson, S., Vyshemirsky, V., Kurth, M.J., Downes, C.S. and Dubitzky, W. (2006) Computational methodologies for modelling, analysis and simulation of signalling networks, *Brief Bioinform*, **7**, 339-353.

Glick, N. (2006) Interferon and its role in immune health. Center for Immune Research.

Gonzalez, P.P., Cardenas, M., Camacho, D., Franyuti, A., Rosas, O. and Lagunez-Otero, J. (2003) Cellulat: an agent-based intracellular signalling model, *Biosystems*, **68**, 171-185.

Hancioglu, B., Swigon, D., Clermont, G. (2007) A dynamical model of human immune response to influenza A virus infection, *Journal of Theoretical Biology*.

Hodgkin, J. (2001) What does a worm want with 20,000 genes?, *Genome Biology*, **2**, 1-4.

Hood, L., Heath, J.R., Phelps, M.E. and Lin, B. (2004) Systems biology and new technologies enable predictive and preventative medicine, *Science*, **306**, 640-643.

Hood, L. and Perlmutter, R.M. (2004) The impact of systems approaches on biological problems in drug discovery, *Nat Biotechnol*, **22**, 1215-1217.

Hornberg, J.J., Bruggeman, F.J., Westerhoff, H.V. and Lankelma, J. (2006) Cancer: a Systems Biology disease, *Biosystems*, **83**, 81-90.

Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., Arkin, A.P., Bornstein, B.J., Bray, D., Cornish-Bowden, A., Cuellar, A.A., Dronov, S., Gilles, E.D., Ginkel, M., Gor, V., Goryanin, II, Hedley, W.J., Hodgman, T.C., Hofmeyr, J.H., Hunter, P.J., Juty, N.S., Kasberger, J.L., Kremling, A., Kummer, U., Le Novere, N., Loew, L.M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E.D., Nakayama, Y., Nelson, M.R., Nielsen, P.F., Sakurada, T., Schaff, J.C., Shapiro, B.E., Shimizu, T.S., Spence, H.D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J. and Wang, J. (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models, *Bioinformatics*, **19**, 524-531.

Hunter, P., Borg, T. (2003) Integration from proteins to organs: the Human Physiome project, *Nature Reviews Molecular Cell Biology*, **4**, 237-243.

Hunter, P., Smith, N., Fernandez, J. and Tawhai, M. (2005) Integration from proteins to organs: the IUPS Physiome Project, *Mech Ageing Dev*, **126**, 187-192.

Hwang, D., Smith, J.J., Leslie, D.M., Weston, A.D., Rust, A.G., Ramsey, S., de Atauri, P., Siegel, A.F., Bolouri, H., Aitchison, J.D. and Hood, L. (2005) A data integration methodology for systems biology: experimental verification, *Proc Natl Acad Sci U S A*, **102**, 17302-17307.

Ideker, T. and Lauffenburger, D. (2003) Building with a scaffold: emerging strategies for high- to low-level cellular modeling, *Trends Biotechnol*, **21**, 255-262.

Isaacs, A., Lindenmann, J. (1957) Virus Interference. I. The interferon, *Proc. Roy. Soc. Lond. B Biol. Sci.*, **147**, 258-267

Kazman, R., Asundi, J. (2001) Quantifying the costs and benefits of architectural decisions. In IEEE (ed), *The 23rd International Conference on Software Engineering*. IEEE, 297-306.

Keller, E.F. (2007) A clash of two cultures, *Nature*, **445**, 603.

Kholodenko, B.N., Demin, O.V., Moehren, G. and Hoek, J.B. (1999) Quantification of short term signaling by the epidermal growth factor receptor, *J Biol Chem*, **274**, 30169-30181.

Kierzek, A.M. (2002) STOCKS: STOChastic Kinetic Simulations of biochemical systems with Gillespie algorithm, *Bioinformatics*, **18**, 470-481.

Kimmel, A.R., Parent, C. A. (2003) The signal to move: D. discoideum go orienteering, *Science* **300**, 1525-1527

Kitano, H. (2000) Perspectives on systems biology, *New Generation Computing*, **18**, 199-216.

Kitano, H. (2001) *Foundations of Systems Biology*. The MIT Press, Cambridge.

Kitano, H. (2002) Computational systems biology, *Nature*, **420**, 206-210.

Kitano, H., Funahashi, A., Matsuoka, Y. and Oda, K. (2005) Using process diagrams for the graphical representation of biological networks, *Nat Biotechnol*, **23**, 961-966.

Kitney, R., Dollery, C. (2007) Systems Biology: a vision for engineering and medicine. In Engineering, A.o.M.S.a.T.R.A.o. (ed).

Klipp, E. and Liebermeister, W. (2006) Mathematical modeling of intracellular signaling pathways, *BMC Neurosci*, **7 Suppl 1**, S10.

Klipp, E., Nordlander, B., Kruger, R., Gennemark, P. and Hohmann, S. (2005) Integrative model of the response of yeast to osmotic shock, *Nat Biotechnol*, **23**, 975-982.

Krueger, C.W. (1992) Software reuse, *ACM Computing Surveys (CSUR)*, **24**, 131-183.

Laue, M.v. (1913) Kritische Bemerkungen zu den Deutungen der Photoframme von Friedich und Knipping, *Physikalische Zeitschrift*, **14**, 421-423.

Lauffenburger, D.A. (2000) Cell signaling pathways as control modules: complexity for simplicity?, *Proc Natl Acad Sci U S A*, **97**, 5031-5033.

Lauffenburger, D.A. (2003) Four M's of Systems Biology. MIT, Cambridge.

Le Novere, N., Bornstein, B., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J.L. and Hucka, M. (2006) BioModels

Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems, *Nucleic Acids Res*, **34**, D689-691.

Le Novere, N., Finney, A., Hucka, M., Bhalla, U.S., Campagne, F., Collado-Vides, J., Crampin, E.J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B., Snoep, J.L., Spence, H.D. and Wanner, B.L. (2005) Minimum information requested in the annotation of biochemical models (MIRIAM), *Nat Biotechnol*, **23**, 1509-1515.

Le Novere, N., Finney, A., Hucka, M., Bhalla, U.S., Campagne, F., Collado-Vides, J., Crampin, E.J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B., Snoep, J.L., Spence, H.D. and Wanner, B.L. (2007) Minimum information requested in the annotation of biochemical models (MIRIAM), *Nat Biotechnol*, **23**, 1509-1515.

Le Novere, N. and Shimizu, T.S. (2001) STOCHSIM: modelling of stochastic biomolecular processes, *Bioinformatics*, **17**, 575-576.

Lindon, J.C., Holmes, E. and Nicholson, J.K. (2006) Metabonomics techniques and applications to pharmaceutical research & development, *Pharm Res*, **23**, 1075-1088.

Liu, E.T. (2005) Systems biology, integrative biology, predictive biology, *Cell*, **121**, 505-506.

Loew, L.M. (2002) The Virtual Cell project, *Novartis Found Symp*, **247**, 151-160; discussion 160-151, 198-206, 244-152.

Ma'ayan, A., Jenkins, S.L., Neves, S., Hasseldine, A., Grace, E., Dubin-Thaler, B., Eungdamrong, N.J., Weng, G., Ram, P.T., Rice, J.J., Kershenbaum, A., Stolovitzky, G.A., Blitzer, R.D. and Iyengar, R. (2005) Formation of regulatory patterns during signal propagation in a Mammalian cellular network, *Science*, **309**, 1078-1083.

Machne, R., Finney, A., Muller, S., Lu, J., Widder, S. and Flamm, C. (2006) The SBML ODE Solver Library: a native API for symbolic and fast numerical analysis of reaction networks, *Bioinformatics*, **22**, 1406-1407.

Mendes, P. and Kell, D.B. (2001) MEG (Model Extender for Gepasi): a program for the modelling of complex, heterogeneous, cellular systems, *Bioinformatics*, **17**, 288-289.

Mishra, J. and Bhalla, U.S. (2002) Simulations of inositol phosphate metabolism and its interaction with InsP(3)-mediated calcium release, *Biophys J*, **83**, 1298-1316.

Morgan, J.J., Surovtsev, I.V. and Lindahl, P.A. (2004) A framework for whole-cell mathematical modeling, *J Theor Biol*, **231**, 581-596.

Neteler, M., Mitasova, H. (2004) *Open Source GIS: A GRASS GIS Approach.*, Boston.

Noble, D. (2006) Systems biology and the heart, *Biosystems*, **83**, 75-80.

Oda, K. (2006) Map of the TLR signaling network, *Nature Molecular Systems Biology.*

Oda, K., Kimura, T., Matsuoka, Y., Funahashi, A., Muramatsu, M., Kitano, H. (2004) Map of the TLR signaling network, *AfCS Research Reports*, **2**, 1-12.

Oltvai, Z.N. and Barabasi, A.L. (2002) Systems biology. Life's complexity pyramid, *Science*, **298**, 763-764.

Palsson, B. (2004) Two-dimensional annotation of genomes, *Nat Biotechnol*, **22**, 1218-1219.

Palsson, B.O., Price, N.D. and Papin, J.A. (2003) Development of network-based pathway definitions: the need to analyze real metabolic networks, *Trends in Biotechnology*, **21**, 195-198.

Papin, J.A., Hunter, T., Palsson, B.O. and Subramaniam, S. (2005) Reconstruction of cellular signalling networks and analysis of their properties, *Nat Rev Mol Cell Biol*, **6**, 99-111.

Patwardhan, B., Warude, D., Pushpangadan, P., Bhatt, N. (2005) Ayurveda and Traditional Chinese Medicine: A Comparative Overview, *Oxford Journals Medicine Evidence-based Complementary and Alternative Medicine*, **2**, 465-473

Pecou, E. (2005) Splitting the dynamics of large biochemical interaction networks, *J Theor Biol*, **232**, 375-384.

Pennisi, E. (2003) A Low Number Wins the GeneSweep Pool, *Science*, **300**, 1484.

Pennisi, E. (2005) How will big pictures emerge from a sea of biological data?, *Science*, **309**, 94.

Peri, S., Navarro,J.D., Amanchy,R., Kristiansen, T.Z., Jonnalagadda,C., Surendranath, V., Niranjan,V., Muthusamy, B., Gandhi, T.K.B., Gronborg, M., Ibarrola,N., Deshpande, N., Shanker, K., Shivashankar, H.N., Pandey, A. (2003) Development of Human Protein Reference Database as an Initial Platform for Approaching Systems Biology in Humans, *Genome Research*, **13**, 2363-2371.

Pogson, M., Smallwood, R., Qwarnstrom, E. and Holcombe, M. (2006) Formal agent-based modelling of intracellular chemical interactions, *Biosystems*, **85**, 37-45.

Putnam, N.H., Srivastava, M., Hellsten, U., Dirks, B., Chapman, J., Salamov, A., Rokshar, D.S. (2007) Sea anemone genome reveals the gene repertoire and genomic organization of the eumetazoan ancestor. Lawrence Berkeley National Laboratory.

Quackenbush, J., Stoeckert, C., Ball, C., Brazma, A., Gentleman, R., Huber, W., Irizarry, R., Salit, M., Sherlock, G., Spellman, P. and Winegarden, N. (2006) Top-down standards will not serve systems biology, *Nature*, **440**, 24.

Raczynski, S. (1996) Differential inclusions in system simulation, *Transactions of the Society for Computer Simulation* **13**, 47-54.

Rajlich, V., Wilde, N. (2002) The role of concepts in program comprehension. *2002 International Workshop on Program Comprehension*. IEEE Computer Society Press, Los Alamitos, CA, 271-278.

Ray, L.B., Adler, E.M., Gough, N.R. (2003) Building a Case for Signaling, *Science*, **300**, 1523-1524.

Robinson, S., R. E. Nance, R. J. Paul, M. Pidd, and S. J. E. Taylor (2004) Simulation model reuse: Definitions, benefits and obstacles., *Simulation Modelling Practice and Theory*, **12**, 479-494.

Sato, M., Taniguchi, T., Tanaka, N. (2001) The interferon system and interferon regulatory factor transcription factors - studies from gene knockout mice, *Cytokine & Growth Factor Reviews*, **12**, 133-142.

Sauro, H.M., Hucka, M., Finney, A., Wellock, C., Bolouri, H., Doyle, J. and Kitano, H. (2003) Next generation simulation tools: the Systems Biology Workbench and BioSPICE integration, *Omics*, **7**, 355-372.

Sauro, H.M. and Kholodenko, B.N. (2004) Quantitative analysis of signaling networks, *Prog Biophys Mol Biol*, **86**, 5-43.

Schilstra, M.J., Li, L., Matthews, J., Finney, A., Hucka, M. and Le Novere, N. (2006) CellML2SBML: conversion of CellML into SBML, *Bioinformatics*, **22**, 1018-1020.

Schulz, M., Uhlendorf, J., Klipp, E. and Liebermeister, W. (2006) SBMLmerge, a System for Combining Biochemical Network Models. *Genome Informatics* 62-71.

Seeman, N.C., Belcher, A. M. (2002) Emulating biology: building nanostructures from the bottom up, *Proc Natl Acad Sci U S A*, **99 Supplement 2**, 6451-6455.

Shannon, P.T., Reiss, D.J., Bonneau, R. and Baliga, N.S. (2006) The Gaggle: an open-source software system for integrating bioinformatics software and data sources, *BMC Bioinformatics*, **7**, 176.

Silva, P. (2002) A general overview of the major metabolic pathways. In http://www2.ufp.pt/~pedros/bq/integration.htm (ed). Universidade Fernando Pessoa.

Slepchenko, B.M., Schaff, J.C., Macara, I. and Loew, L.M. (2003) Quantitative cell biology with the Virtual Cell, *Trends Cell Biol*, **13**, 570-576.

Snoep, J.L., Bruggeman, F., Olivier, B.G. and Westerhoff, H.V. (2006) Towards building the silicon cell: a modular approach, *Biosystems*, **83**, 207-216.

Stultz, C. (2007) Intractability of using atom-by-atom molecular dynamics for modeling biological pathways. In Ayyadurai, S. (ed). Cambridge, Personal Communication.

Subbarayappa, B.V. (1997) Siddha medicine: an overview, *Lancet*, **350**, 1841-1844.

Takahashi, K., Kaizu, K., Hu, B. and Tomita, M. (2004) A multi-algorithm, multi-timescale method for cell simulation, *Bioinformatics*, **20**, 538-546.

Takauji (2002) CpG-DNA-induced IFN-Γ production involves p38 MAPKdependent STAT1 phosphorylation in human plasmacytoid dendritic cell precursors, *Journal of Luekocyte Biology*, **72**, 1011-1019.

Taniguchi, T., Ogasawara, K., Takaoka, A., Tanaka, N. (2001) IRF family of transcription factors as regulators of host defense, *Annual Review Immunology*, **19**.

Tomita, M., Hashimoto, K., Takahashi, K., Shimizu, T.S., Matsuzaki, Y., Miyoshi, F., Saito, K., Tanida, S., Yugi, K., Venter, J.C. and Hutchison, C.A., 3rd (1999) E-CELL: software environment for whole-cell simulation, *Bioinformatics*, **15**, 72-84.

Vaidehi, N., Goddard, W.A. (2001) Atom-Level Simulation and Modeling of Biomacromolecules. In Bower, J.M., Bolouri, H. (ed), *Computational Modeling of Genetic and Biochemical Networks*. MIT Press, Cambridge, 161-163.

Wang, X.H., Connors, M., Wilson, D., Wilson, G., Nicholosn, G.M., Smith, R., Shaw, D., MacKay, J., Alexwood, P., Chrisite, M., King, G. (2001) Discovery and structure of potent and highly specific blocker of insect calcium channels, *Journal of Biological Chemistry*, **276**, 40306-403012.

Watson, J.D., Crick, F.H. (1953) Molecular structure of nucleic acids: A structure of deoxyribose Necleic Acid, *Nature*, **171**, 737-738.

Webb, K. and White, T. (2005) UML as a cell and biochemistry modeling language, *Biosystems*, **80**, 283-302.

Whelan, G., Castleton, K.J., Buck, J.W., Hoopes, B.L., Pelton, M.A., Strenge, D.L., Gelston, G.M., Kickert, R.N. (1997) Concepts of a framework for risk analysis in multimedia environmental systems (FRAMES). In Laboratory, P.N.N. (ed), *PNNL-11748*. Pacific Northwest National Laboratory, Richland.

White, J. (2007) Two protein interactions are intractable using molecular dynamics. In Ayyadurai, S. (ed). Cambridge, Personal Communication.

Wiener, N. (1948) *Cybernetics or Control and Communciation in the Animal Machine*. The MIT Press, Cambridge.

Wikipedia (2007) Interferon, *Wikipedia, The Free Encyclopedia*, **10**, <http://en.wikipedia.org/wiki/Interferon>.

Xia, L., Wang, L., Chung, A., Ivanov, F. (2002) Identification of Both Positive and Negative Domains with the Epidermal Growth Factor Receptor COOH-terminal

Region for Signal Transducer and Activator of Transcription (STAT) Activation, *Journal of Biochemical*, **277**, 30716-30723.

Yamada (2001) Computer Modeling of JAK/STAT Signal Transduction Pathway, *Genome Informatics*.

You, L., Hoonlor, A. and Yin, J. (2003) Modeling biological systems using Dynetica--a simulator of dynamic networks, *Bioinformatics*, **19**, 435-436.

Zi, Z., Cho, K., Sung, M., (2005) In silico identification of the key components and steps in IFN-gamma induced JAK-STAT, *FEBS*, **579**, 1101-1108.

Colophon

Endnote was used to format the bibliography format. It was primarily selected due its ease of use and integration with Microsoft Word.

This thesis was typeset using Microsoft Word, a decision that was agonized over long and hard. The alternative was to use FrameMaker or pdfLATEX. But FrameMaker has been largely abandoned by Adobe and pdfLATEX and only had indirect support EndNotes.

The text of this thesis was set in Book Antigua. This thesis has many screen shots. The Windows screen shots were created using the simple screen capture from Windows operating system.

This thesis was written and typeset to the music of Bruce Springsteen, Bob Dylan, Johnny Cash and from time to time music.

As of August 29, 2007, the PDF file that was used to print this thesis is approximately 20.3 megabytes in length.