

Integration of a Cancer Research KnowledgeBase with a Cross-Hierarchy Modeling Platform

Roger Day

University of Pittsburgh Department of Biostatistics,
University of Pittsburgh Cancer Institute
Suite 325 Sterling Building
201 North Craig, Pittsburgh PA 15213
day@upci.pitt.edu

William Shirey

U. of Pittsburgh Cancer Institute
Suite 325 Sterling Building
201 North Craig, Pittsburgh PA 15213
shirey@upci.pitt.edu

ABSTRACT

There is a sense of promise that accelerating growth of knowledge about the molecular basis for the behavior of cancer cells, especially about the “cancer genome”, will lead, if not to the “magic bullet”, at least to much better treatments for patients. But there is a tension between pure empiricism pinning hopes on the sheer quantity of data and the use of biological reasoning to draw insights and improve predictions. Our premise is that at least some of the latter approach is essential, and a general cancer modeling system could be key to pulling the information together in a truly useful way. A comprehensive software-based facility to synthesize information, build models, simulate, and validate is in development. We have built a comprehensive modeling system for the cancer process, integrating the constituent multiple interacting processes at different scales, including the cancer cell, the patient, the oncologist, and the clinical trial. This is called the Oncology Thinking Cap (OncoTCap). This paper describes how the program achieves its generality, how it currently integrates with a knowledge base that serves as a model authoring tool, and how the knowledge base design will extend naturally to include an information-gathering tool. The gathering and structuring of research information lead to its formulation into model-building statements and model-validation goodness-of-fit criteria. With all the pieces in hand, a multitude of ambitious applications could be realizable.

1. INTRODUCTION

The explosion of biomedical research information presents a serious challenge: to harness this daunting and expanding set of resources to help prevent cancer cases or to treat patients more effectively. There are impressive efforts to catalogue the information as it grows in a way that allows different databases to interoperate. An especially important resource with this goal is the Entrez web-based family of tools¹⁻³. In the realm of cancer biology and cancer treatment, there is a sense of promise that accelerating growth of knowledge about the molecular basis for the behavior of cancer cells, especially about the “cancer genome”, will lead to major improvements in treatment. Many current efforts aim to identify new molecular targets for new therapies, to detect cancer earlier or classify or “stage” cancer more accurately for prediction and for prognosis. But there is frustration that the translation of new knowledge into treatment advances is not quicker.

There is a tension between pure empiricism pinning hopes on the sheer quantity of data and the use of biological reasoning to draw insights and improve predictions. Two arguments for pure empiricism are the vast quantity of data from high-throughput assays like microarrays and the sad history of plausible bright ideas refuted by clinical trials. Our premise is that the use of biological reasoning to draw insights and improve predictions is essential, and that the ability to visualize the consequences of biological assumptions or new biological discoveries may require conducting thought experiments with the help of a computer. The level of cancer cells is complex enough; molecular networks governing cell division, cell death, DNA repair, angiogenesis and others are individually complex but also share components. Then there are interactions between cancer cells and their neighbors (endothelia, immune cells, stroma). Effects on the patient lead to treatment measures that lead to changes in the tumor. Changes in the tumor are measured, feeding back and affecting the behavior of the oncologist. There seems to be enough complexity to justify a little assistance to the cancer thinker. Our goal, then is a comprehensive software-based facility to synthesize information, build models, simulate, and validate.

We have built part of this system. The Oncology Thinking Cap (OncoTCap) is a comprehensive modeling system for the cancer process, integrating the constituent multiple interacting processes. Different tumor cell subpopulations coexisting within the same tumor exhibit varied susceptibilities to antineoplastic agents and other treatment modalities. Therefore it is necessary to simultaneously take into account many types of heterogeneity. Gene dosage, gene expression, and mutated genes affect such critical processes as metastasis, local spread, apoptotic pathways, drug resistance mechanisms, angiogenesis, genetic repair mechanisms, cell cycle regulation, treatment-induced mutation, and immunogenicity. Also part of the whole picture are toxicity, nonlinear dose-response, and drug interactions.

This paper describes how OncoTCap achieves its generality, how it currently integrates with an “engine-driver” knowledge base that serves as a model authoring tool, and how the knowledge base design can extend naturally to complete the comprehensive system. The extensions are a cancer information resource with an information-gathering tool, and model-building/model-validation framework. The gathering and structuring of research information leads to its formulation into model-building statements and model-validation goodness-of-fit criteria. These

facilities are also structured as a knowledge bases coordinating with the “engine-driver”.

2. SIMULATION MODEL ASSUMPTIONS

The core of the OncoTcap computer program consists of a simulator engine that implements the multitype branching process model for cell populations. Thus there is one key assumption: that the cancer cell population divides into homogeneous subpopulations characterized by a set of attributes with discrete values. The attributes can describe for example the state of the genome (karyotype, amplifications, point mutations etc.), expression levels of important proteins, physical location, characteristics of the microenvironment. A basic Markov property is essential. Therefore “age”, or time since last cell division, is not such an attribute. Rather, each subpopulation is an age-mixed ensemble, and changes in the vector of cancer cell counts in successive short time intervals is determined by binomial sampling in each subpopulation.⁴ to select the number of cells which experience cell division, cell death, or whatever mutations or migrations are allowed. The probabilities of these events are functions of the individual attributes and systemic variables. For example, one kinetic event might be cell death through a particular apoptotic mechanism triggered by a chemotherapy drug. The hazard of cell death (probability per unit time) would depend on the presence of components of the apoptotic pathway, absence of amplification of blockers like bcl-2, and systemic concentration of the drug. A detailed model of individual cell changes through time, such as provided by **e-cell** or **virtual cell**, is not attempted.

The previous version of OncoTCap (Version 2) has two computational engines. The "PGF engine" provides analytic solutions to partial differential equations governing the joint probability generating function (p.g.f.) for the vector of counts of cells of different types. The "PS engine" generates patient simulations through stochastic simulation of individual histories. Extensive comparisons of the $N=\infty$ result from the PGF engine to the $N<\infty$ result from multiple runs of the PS engine validate the engines by cross-comparison and goodness-of-fit testing^{5,6}. Another advantage of having the two engines is the ability to discover principles using the PGF engine, then view specific patient histories from the PS engine to gain intuition and insight.

The PGF engine has been applied to combination treatment issues, and has identified a compelling novel principle for optimizing treatment^{7,8}. It is computationally extremely efficient, and can tackle general population structures. However, it has severe limitations, especially when parameters are dependent on the current state, which is rather fundamental to biology. For these reasons, stochastic simulation provided by the PS engine is an attractive practical alternative and supplement. In the latest version, OncoTCap 3, the stochastic simulator is the only engine at this time.

3. SIMULATION ENGINE-DRIVER

3.1 Overall Architecture

An overview of the coordination between the Engine Driver knowledge base and the OncoTCap 3 simulation engine is diagrammed in Figure 1.

The knowledge base provides the model author with the tools to select statements (of the regular English language variety)

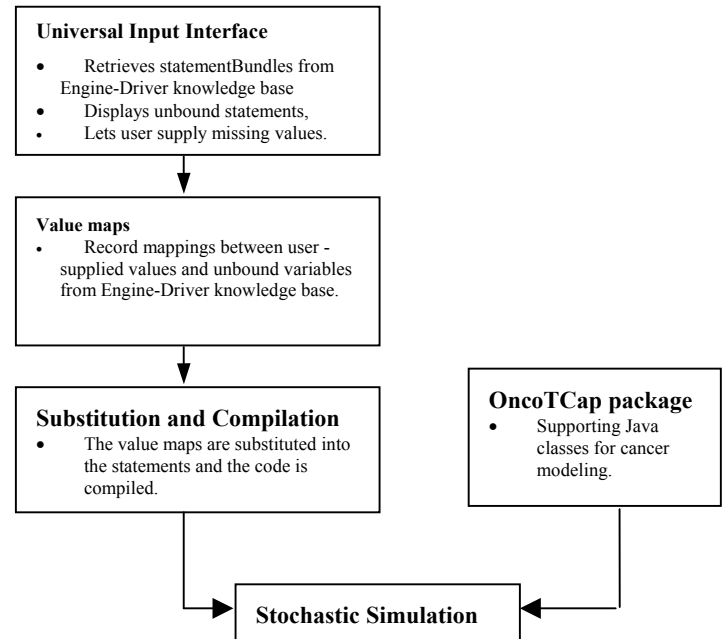


Figure1: Architecture for OncoTCap3

describing aspects of the behavior of cancer cells, patients and their organs systems, or oncologists, referred to here as the “behavioral processes”. When the OncoTCap initial screen is started, a particular set of these statements is extracted from the knowledge base, and loaded into the interface. Some statements are “fully bound”; no additional information is needed to use them. Others may be “unbound”, that is additional user decisions must be made. The interface supports these choices.

For example, in a recent application to teach fourth year medical school students about cancer clinical trials in relation to cancer biology, the bound statements included the initial tumor burden distribution, the rules governing tumor heterogeneity and evolution, sensitivity to treatments, and so on. The unbound statements included student choices as to clinical trial treatment regimen, study design, dose reduction rules and so on. The next figure shows an input interface screen, with a student in the process of choosing a treatment schedule.

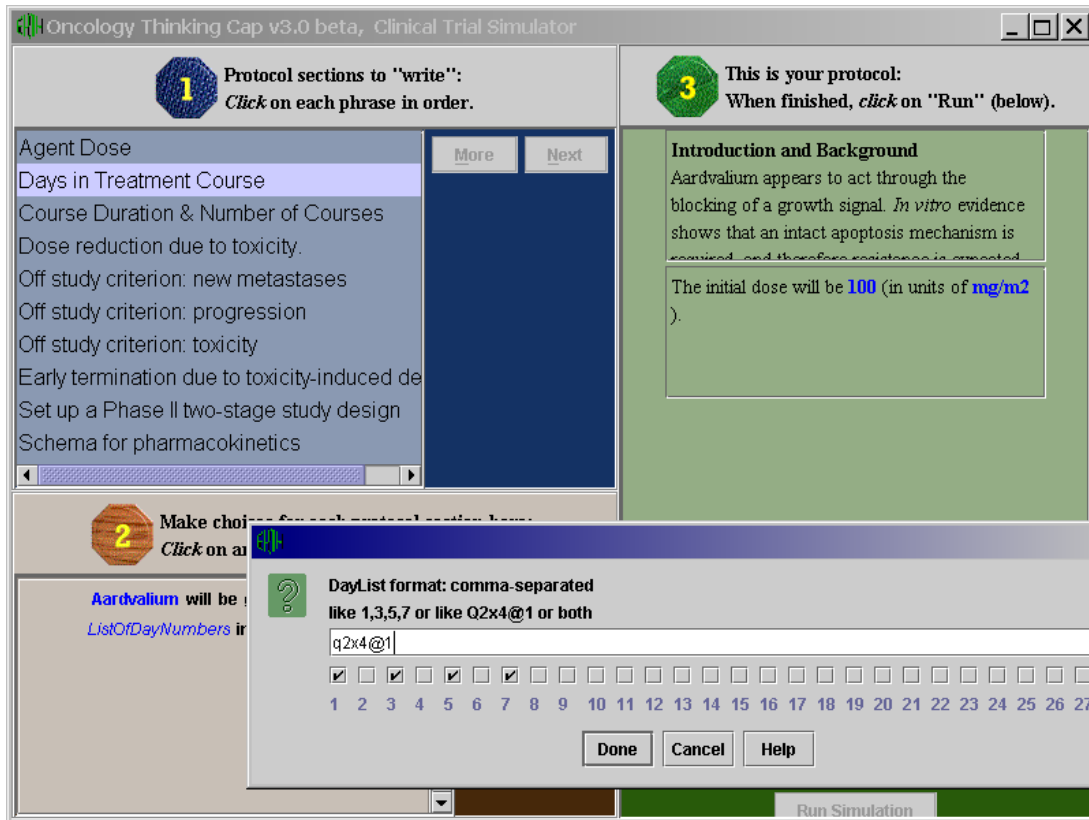


Figure 2: Scenario interface, with components: (1) Unbound statements, (2) Statement template for treatment schedule, (3) completed statements thus far, (Overlay) schedule input window.

When the decisions have been made, one can run a simulation from the model. The knowledge base contains Java program code for each of the statements. This code is combined with the Java package *edu.pitt.upci.oncotcap* to start the simulation; more details are provided below.

The main strength of this architecture stems from the complete isolation of the simulation engine itself. The architecture yields virtually complete flexibility in adding new features (whether in cancer biology, pharmacology, patient management, or clinical trial designs) without adding new permanent data structures to the computational engine, and almost eliminates the need for new screen development.

3.2 Authoring Of Cancer Models for Simulation

The components of cancer models are represented in a knowledge base. The mission of the knowledge base is to represent how statements about cancer are to be translated into computer code.

3.2.1 The Protégé knowledge management system.

Protégé 2000 is a general-purpose program written in Java which supports capture of knowledge in contexts specific to a domain

area. A user develops a frame-based ontology representing concepts in the knowledge domain, in the form of “classes” with

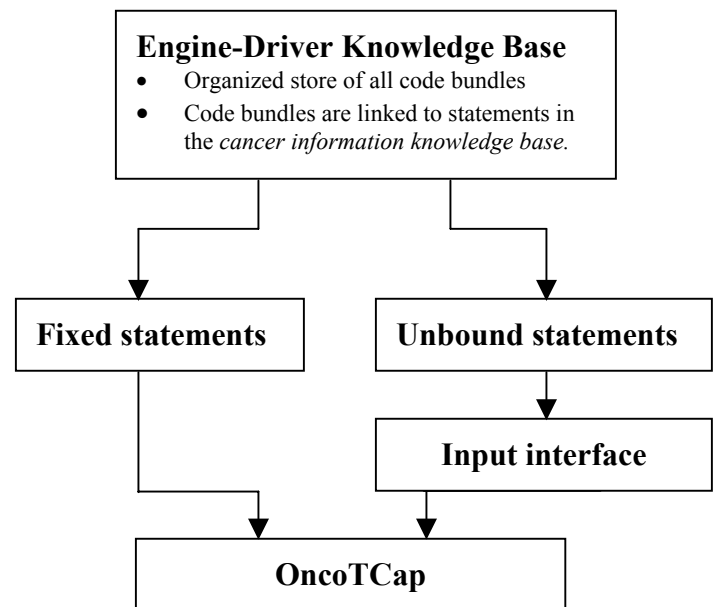


Figure 3: Engine-Driver Knowledge Base used for model building

defined content fields or “slots”. One then customizes on-screen forms for capturing the information, and proceeds to input “instances” of the ontology classes by filling in values for the slots. The ontology need not be fixed, but can change after instances have been created. The system has a rapidly growing base of users. Application areas are diverse, but several groups are intensely working on the development of clinical guidelines.

Protégé supports the addition of specialized plug-ins to the interface, and also supports external access to the data via a development kit. This is convenient for rapidly creating and grafting on tools to drive or interact with another program. A customized interface element called the WizardTab was developed, using the Protégé Development Kit, to support this authoring.

A growing community of developers is building a variety of sharable plug-ins. (We developed a specialize tab plug-in to facilitate the authorship process.) Knowledge bases may also be shared among groups. For example, the “Design-A-Trial”⁹ team at Imperial Cancer Research Fund is currently working to develop a Protégé-based “interlingua” that could accelerate the development of OncoTCap as well as other projects. Several groups are developing ways to include “problem-solving methods” so that the raw information can be subjected to various kinds of domain-specific reasoning.

3.2.2 Ontology for cancer model authorship.

The ontology developed for driving the OncoTCap program with a particular cancer model is oriented around the concepts of modeling-building rather than the cancer domain (see Section 5.2). From this viewpoint, a model is a *StatementBundlePackage*, that is a collection of statements that will (or could) be made. Each constituent *StatementBundle* contains prescriptions for the behavior of the fundamental processes in the model, which include *CancerCell*, *Patient*, *Oncologist*, and *ClinicalTrial*.

3.2.2.1 Close-up of the StatementBundle

The simulations performed and displayed are specified by sets of code which represent a single idea. Each set we call a *StatementBundle*, representing a single cognitive whole and labeled with a natural-language expression. A *StatementBundle* may govern the behavior of the cancer cells and the organ status variables as a function of the current state of the system (*biological-physiological*), or it may represent *decisions* made in regard to the treatment or experimental protocol.

A *StatementBundle* does not have to be fully specified. When OncoTCap is used in a model-fitting context, the undetermined degrees of freedom are within the *biological-physiological* rules, and are left up to model-fitting algorithm to fill in. When OncoTCap is used in interactive mode by a researcher or learner who is exploring the consequences of choices, the undetermined degrees of freedom correspond to decisions made by professionals planning research or managing patients, and the choices are made directly through the user interface. When the *StatementBundle* is not fully specified, it is represented by a visible template with

slots to be filled, as in Figure 1. Examples of *StatementBundle* templates describing *decisions* are:

- The initial dose will be *initial dose* (in units of *units*).
- If a *toxicity type* toxicity of grade *toxgrade* occurs, then reduce the dose by *ToxPctReduction* percent *ReductionDuration*
- Tumor evaluations will be scheduled every *tumor scan interval* days.

The underlined-italicized phrases are linked to user dialog boxes supporting the input of appropriate values for the unspecified slots. In an education setting, one *StatementBundle* contains the biology and physiology, and is not seen by the students, while the other *StatementBundle*'s, which represent choices to be made, are presented to the students in the interface. The choices in the hands of the students include setting doses and schedules for a treatment plan, making contingency plans for adverse events, specifying a clinical trial design, or setting schemas for assays, tests and scans. Each exercise defined by a set of *StatementBundle*'s we call a “problem-based scenario”.

3.2.2.1.1 Close-up of the CodeBundle

In general, a model (fully specified or not) consists of a set of *StatementBundle* objects. Each in turn consists of a set of *CodeBundle* objects. A *CodeBundle* is a collection of sections of Java code, each one associated with a particular Java class and a particular code section within that class. To illustrate this, consider a *StatementBundle* whose template reads as follows:

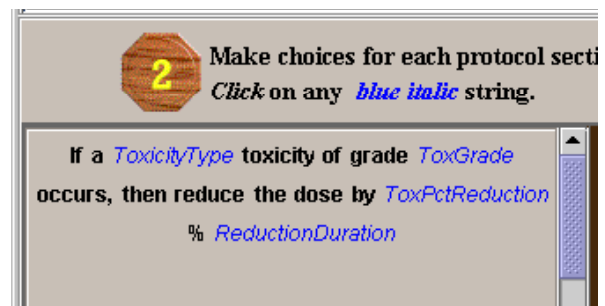


Figure 4: *StatementTemplate* as it appears in interface.

The corresponding *codeBundle* in the Protégé Engine-Driver is in Figure 5. The code will be inserted into the initialization section when the *Oncologist* class is created by subclassing *AbstractOncologist*.

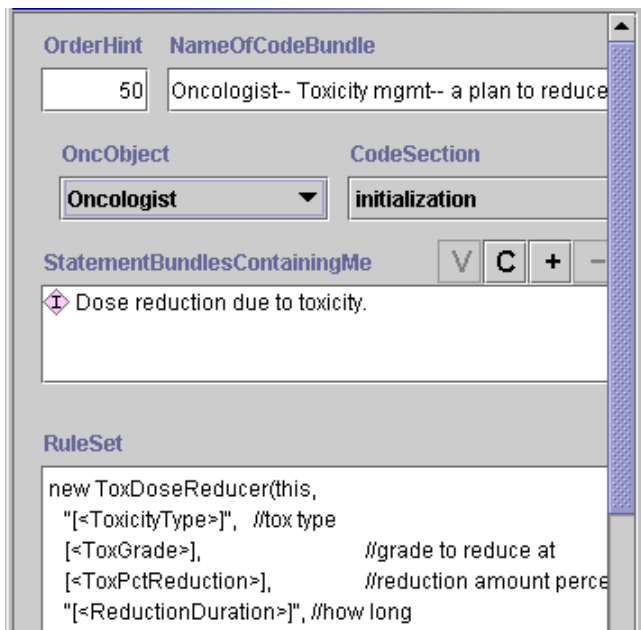


Figure 5: CodeBundle in Protégé Engine-Driver

The text delimited by “[<” and “>]” are replaced by the user’s choices stored in the StatementBundle’s valueMap.

The objects CancerCell, Patient, Oncologist, and ClinicalTrial are concretized subclassed versions of AbstractCancerCell, AbstractPatient, AbstractOncologist, AbstractClinicalTrial. Each

of these classes has the following sections:

- initialize(),
- updateActions(),
- updateVars(),
- updateFromObservable(),
- IDVariables,
- WorkingVariables,
- new Methods.

3.3 Runtime Assembly and Compilation

The assembly of the code proceeds as follows. Any code within a CodeBundle is first subjected to substitution of user-specified values; for example, the placeholder for the initial drug dose would be replaced by the user’s choice. Then the code for the object is assembled with appropriate wrappers. There are wrappers for each of the methods initialize(),updateActions(), updateVars(),updateFromObservable(), and there is a wrapper for the entire class. The other code sections are just copied into the body of the class.

With the code assembled, it is then compiled. This takes place during run-time. Finally, a starter object is instantiated (e.g. ClinicalTrial), which starts the simulation. The output display for our educational applications shows individual patient clinical histories, graphs of cell counts, pharmacokinetic parameters and organ function status, and a summary of the clinical trial result as it proceeds.

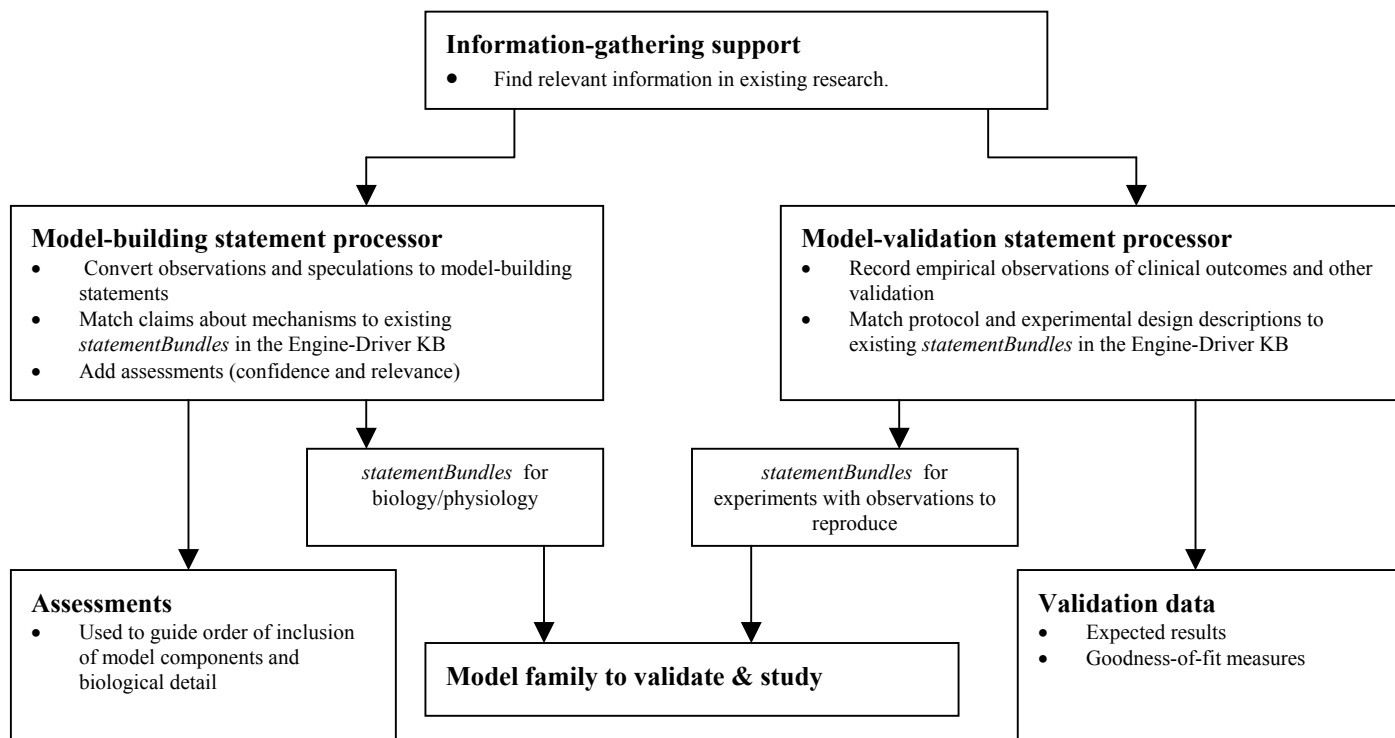


Figure 6: Cancer Information Genie

3.4 Modification for Expert Model-Specification

Cancer researchers should be able to use the same facility for thought experiments and modest model-building. The only modification required is to ensure that the repository of *statementBundles* and *codeBundles* is extensive enough, and organized in a navigable fashion; a searchable concept tree is appealing.

A tree of these *statementBundles* then replaces Section (1) of Figure 2. Thus, to fill in this tree and support the model-building process, a catalog of pre-coded *statementBundles* with tested *codeBundles* is required, as part of the knowledge base. Our current catalog is limited to the *statementBundles* describing decisions in clinical trial design, and to the specific *statementBundles* needed to represent the biology of cancer cell behavior and pharmacokinetics for the specific educational scenarios developed thus far. Because of the separation of the simulation engine from the knowledge base, this catalog can grow as needed, but its extension requires Java programming expertise, but its use in model-building does not.

The top-most broad categories of the catalog's tree structure include cancer cell behavior, behavior of neighboring cells, behavior of organs in response to tumor cells and to treatments, metabolism and clearance and fate of chemical species, patient management rules, generation of observations (imaging, blood tests, physical exam, pain reports etc), clinical trial management rules. The coverage of the catalog at the finer levels needs to include, for example:

- generic representations of cell division signaling pathways, apoptotic pathways, different kinds of mutation events, DNA repair and drug resistance pathways, capabilities for migration and metastasis and angiogenesis,
- representations of the roles that oncogenes and tumor suppressor genes in these pathways.
- different patterns for the pharmacodynamics of drug action on cancer cells and on function of physiologic systems
- a variety of clinical trial protocol statements including clinical trial designs.

With this design, one can plan a sensitivity analysis taking into account many complexities whose consequences are not transparent. In cancer research, examples of characteristics or events with multiple consequences, or interrelations among distinct subprocesses, abound.

- Cancer treatments are often mutagenic, with effects on tumor heterogeneity leading to further treatment resistance, metastatic potential, and other dangerous phenotypes.
- Cancer treatments often induce transient resistance mechanisms.

- A mutation which disables or cripples a DNA repair mechanism would affect mutation rates, but only of the type repaired by the mechanism.
- The expression of alleles of p53 affect cell cycle control, the effectiveness of DNA repair, and entry into apoptosis.
- Interferon alpha can have direct effects on cancer cells, effects on neighboring cells of the immune system, and cytotoxic effects on neighboring immature vascular cells.
- Overexpression of c-myc can lead to defective cell cycle control, but also to increased expression of VEGF, an angiogenic molecule.
- Treatment with one drug can change expression of metabolic enzymes in the liver, altering metabolism of the same drugs or other drugs.
- The her2-neu receptor, so important as a marker of high recurrence risk in breast cancer patients, also appears to mark heightened sensitivity of tumors to certain chemotherapy drugs.

4. MODEL-BUILDING AND MODEL-VALIDATION

The architecture described above can be expanded to incorporate knowledge-gathering as a preamble to model-building. Figure 6 describes an idealized picture of a supporting facility to build cancer models. The most important features are:

- support provided to the user or team in locating information,
- assistance in processing gathered information relevant to building a family of models to be studied,
- assistance in processing gathered information relevant to building a validation suite for the models,
- managing the execution of the model simulations,
- supporting the comparison of model results to the observed results of the validation suite

The elements in this list seem nearly indispensable in any future comprehensive simulation-based study of biological systems. This justifies adding a little detail for each element in turn.

4.1.1 Information-gathering support

Intelligent facilities to assist in the location of relevant research results will obviously be important. Efforts such as MedMiner¹⁰ are underway by many groups to go well beyond keyword searches, to base such facilities on contexts, relationships, and perhaps semantic content.

4.1.2 Model-builder statement processor

Research information articles, personal communications, etc.) is gathered, statements extracted and further processed. Some statements help describe the behavior of pieces of the system, for example output of molecular networks given a cancer cell's genome and immediate environment. A model-builder statement processor will be a software interface whose functions are:

- Help user translate original text into verbal prescriptions for the behavior of one of our processes (cancer cell, endothelial cell, organ system) under the conditions of the experiment.
- Help user to locate *statementBundles* in the Engine-Driver knowledgebase that represent the idea embodied by the prescriptions.
- Help user to score the prescription with a confidence assessment (assessment of the information according to one's confidence in its truth in the setting of its discovery)
- Help user to score the prescription with a relevance assessment (assessment of information according to its relevance to the setting of the problem under attack)

4.1.3 Model-validation statement processor

Research information articles, personal communications, etc.) is gathered, statements extracted and further processed. A model-builder statement processor will be a software interface whose functions are:

- Help user translate original text into verbal prescriptions for the experimental protocol (e.g. for clinical trials, the study design, sample selection, treatment plan, dose adjustments, etc.), and above all the outcomes of the experiment.
- Help user to locate *statementBundles* in the Engine-Driver knowledgebase that represent the prescriptions of the experimental protocol.
- Help user to score the outcome with a method for calculating a "goodness-of-fit" measure. This could be purely statistical, purely subjective, or a combination.

4.1.4 Model-execution

To execute the simulation of a model will require both the biological and physiological component from the model-building side, plus the experimental protocol from the model-validation statement processor. The engine-driver knowledge base which we currently use then comes into play as before, to assemble and compile the Java code and run the simulation.

4.1.5 Validation support

Finally, the outcomes of the simulations must be matched to the actual observed outcomes of the studies on which the "protocol" part of the model is based. The goodness-of-fit method is looked up and applied to generate comparisons results. Results of the comparisons are assembled and recorded.

4.1.6 Ontology for Model-building and Validation

A Protégé ontology representing these ideas has been prototyped and is seen in Figure 7. This knowledge base will coordinate with the engine-driver knowledge base described above that paves the way from the selected assumptions (*statementBundles*) to drive the modeling engine. Figure 8 shows how the knowledge gathered in this way can be represented diagrammatically within the knowledge base. Clicking on elements in the diagram, one can browse "up" to the literature sources behind the claims, or "down" into the *statementBundles* that represent them.

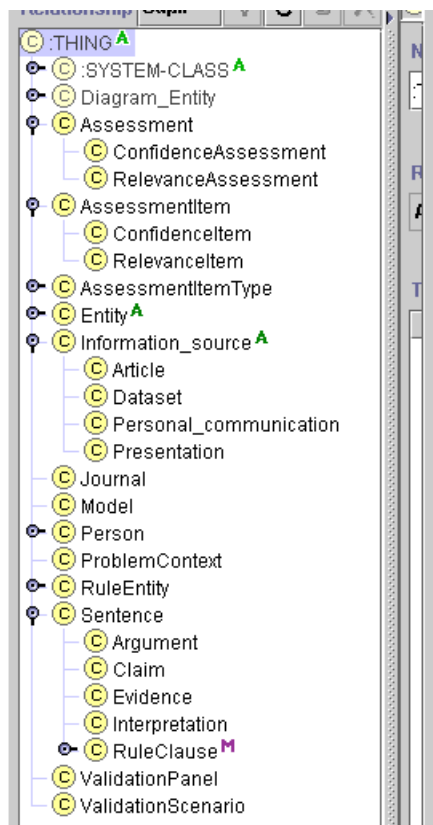


Figure 7: Model-Building Ontology

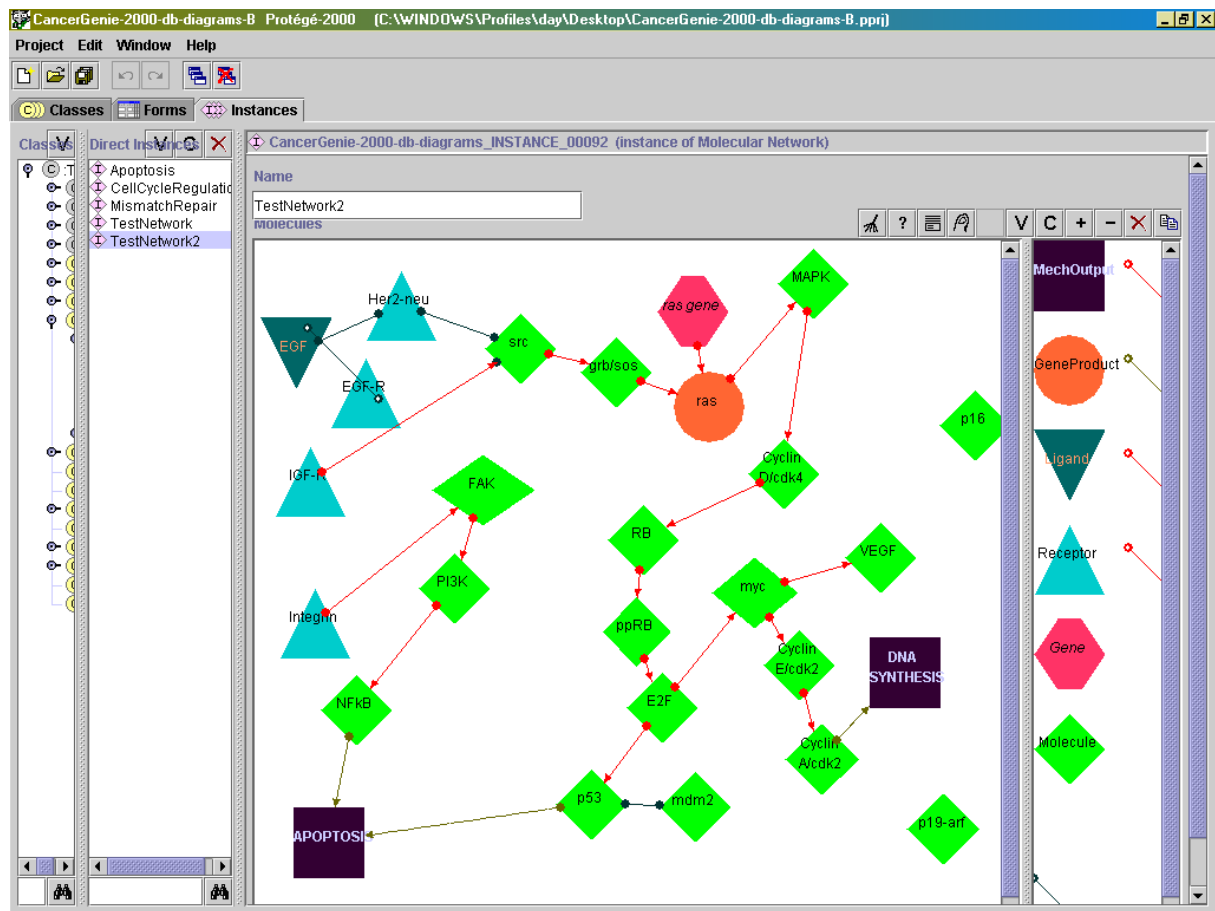


Figure 8: Diagram for browsing statement instances, Model-Building knowledgebase

5. THE FUTURE OF INTEGRATION

5.1 What about model-fitting?

Model-fitting would require that the goodness-of-fit measures from model validation be combined into an objective function to be maximized (or minimized). Then the value would in some way feed back to guide changes in the settings of individual model parameter values, or to insert or remove entire chunks of the model.

Nobody can fail to notice the immense challenge due to the difficulties of combining heterogeneous goodness-of-fit measures from different kinds of studies, the high dimensionality of the model spaces that will result from model-building, and the challenge of using assessments to guide the successive inclusion and exclusion of portions of the processes. The use of stochastic simulation also multiplies the computational demand by perhaps two orders of magnitude,

because each model yields a distribution of outcomes that usually can only be obtained by multiple runs.

The challenges have some partial answers. Starting from the very simple and adding detail to the model family slowly is an obvious essential strategy. Another strategy is to use the entire machinery for thought experiments in search of general principles. A successful search would be followed up by sensitivity analysis, to confirm its universality. This is not unprecedented. A study of optimal combination of two cancer treatments followed just this paradigm⁷.

In some cases the demands for a “good” fit may be limited because the purpose of the exercise is limited. For example, in education of cancer professionals, this system could support the development of automatic case generation, to give oncologists-in-training the opportunity to experience decision-making. There the criteria for a good-enough model may be achievable much more easily than a model with predictive power.

5.2 What about an ontology for cancer knowledge?

We have discussed a knowledge base for specifying how statements about cancer can be translated into simulation code, and we have discussed a knowledge base for specifying how research information can be translated into statements about cancer and processed for model-building and model-validation. Readers may be curious at the absence of discussion of an ontology for cancer knowledge itself. This would be akin to the knowledge bases developed for clinical guidelines; knowledge about the actual substance, not about statements about the actual substance. This would be daunting to create and maintain. Its “value-added” might include supporting the semi-automated parsing of the research statements to make it more efficient to find matching *statementBundles*. Perhaps it would even support engines for reproducing some of the reasoning processes of biologists.

6. SUMMARY

Molecular biology has brought an explosion of knowledge about many of the processes that transform a normal cell into a malignant one, cause malignant cells to become more aggressive, and allow malignant cells to resist anticancer treatment. A unifying idea is the development of heterogeneity within a tumor primarily through the loss of replication control. Natural selection acts upon a heterogeneous tumor cell population to change the tumor's composition over time, generally to the detriment of the patient or host. Comprehensive modeling of the evolution of tumor cell populations is likely to become increasingly important in the years to come. Cancer researchers need to reason from knowledge of molecular and cell biology to consequences for an entire tumor and the patient subjected to the tumor, for example to design better prevention and therapeutic strategies. However, intuition concerning cell population kinetics as applied to cancer treatment is generally poor. This deficiency, together with the information overload in cancer research, makes it difficult to make reliable and efficient use of available research information. At the same time, computational power and advances in software development open up possibilities to augment or replace informal reasoning and unreliable intuition by well-designed software. We think that the essential architectural elements of the scheme we describe here will play a role in 21st century medicine.

7. ACKNOWLEDGMENTS

Support for this work was provided in part by Grant # R25CA63548 of the National Cancer Institute. Sailesh Ramakrishnan and Qing-shou Huang contributed directly to the software development. Project. Colleagues who have provided invaluable guidance include Charles Friedman, Adam Brufsky, Joseph Baar, Merrill Egorin, George Sledge, Emil Frei III, William Peters, Stanley Shackney. Immense thanks go to the Protégé team, especially Mark Musen and Ray Ferguson.

8. REFERENCES

- [1] Rioux PA, Gilbert WA, Littlejohn TG. A portable search engine and browser for the Entrez database. *J.Comput Biol*, 1: 293-295.
- [2] Tatusova TA, Karsch-Mizrachi I, Ostell JA. Complete genomes in WWW Entrez: data representation and analysis. *Bioinformatics.*, 15: 536-543.
- [3] McEntyre J. Linking up with Entrez. *Trends Genet.*, 14: 39-40.
- [4] Day, R. S., Shirey, W., Ramakrishnan, S., and Huang, Q. Tumor biology modeling workbench for prospectively evaluating cancer treatments. 1998. Proceedings of the IEEE /CESA '98: Computational Engineering with Systems Applications, Hammamet, Tunisia.
- [5] Huang, Q. S., Day, R., and Bryant, J. Goodness-of-fit tests for simulators of multitype branching process based on the joint p.g.f. 157th Annual Joint Statistical Meetings .1996.
- [6] Day, R. S., Huang, Q. S., and Ramakrishnan, S. Pseudorandom number generators, goodness of fit tests, and stochastic simulation of heterogeneous tumors. 1999. Proceedings of the 160th Annual Joint Statistical Meetings (Biometrics Sect.). 1999.
- [7] Day RS. Treatment sequencing, asymmetry, and uncertainty: protocol strategies for combination chemotherapy. *Cancer Res*, 46: 3876-3885.
- [8] Norton L, Day RS. Potential innovations in scheduling cancer chemotherapy. In: DeVita VT, Jr., Hellman S, Rosenberg SA, editors. *Important Advances in Oncology 1991*, 1 ed. Philadelphia, J.B. Lippincott Company. 1991. 57-71.
- [9] Wyatt JC, Altman DG, Heathfield HA, Pantin CF. Development of Design-a-Trial, a knowledge-based critiquing system for authors of clinical trial protocols. *Comput Methods Programs Biomed.*, 43: 283-291.
- [10] Tanabe L, Scherf U, Smith LH, Lee JK, Hunter L, Weinstein JN. MedMiner: an Internet text-mining tool for biomedical information, with application to gene expression profiling. *Biotechniques*, 27: 1210-1217.