THE SCHOLAR'S COMPANION® WHITE PAPER

"Imagination is more important than knowledge."

Albert Einstein

and, speaking of imagination...

"I think there is a world market for maybe five computers."

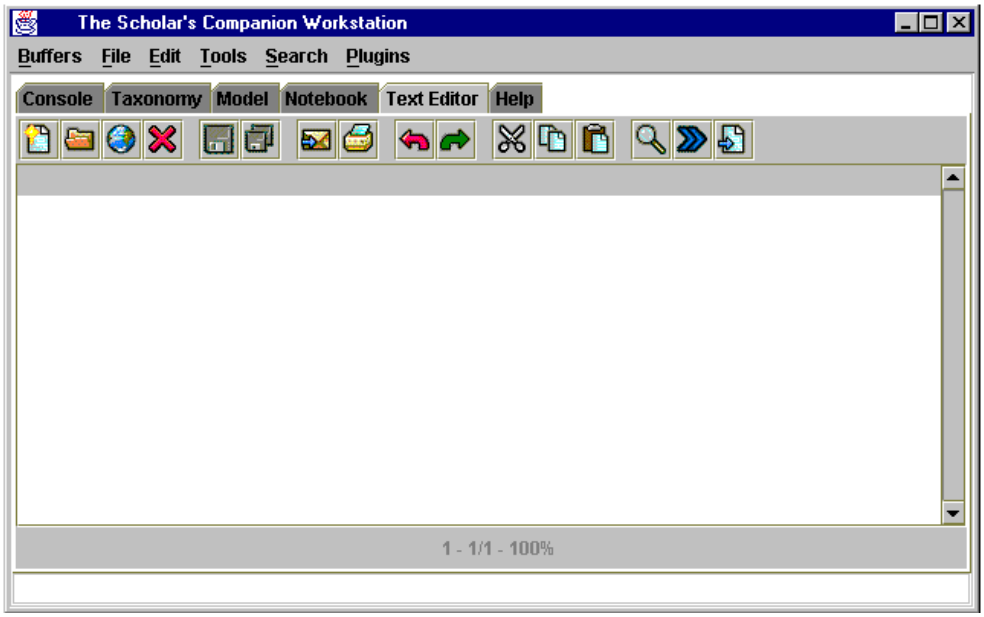
Thomas Watson, chairman of IBM, 1943

Abstract

The Scholar's Companion (TSC) is a software project in two parts: an expert system built with a qualitative reasoning architecture, and a user-centered graphical user interface (GUI). Early versions of TSC have been used in such diverse domains as K-12 learning, Ph.D. research in materials engineering, and medical research in the field of immunology. TSC, serves as an "intelligence amplifier" for a scholar, providing several benefits in the conduct of research.

Introduction

The Scholar's Companion is a project that has been inspired by the notion of providing support for scholarship. Scholarship involves any individual, from a student of the sciences to those whose studies pursue the arts, literature, or philosophy. Support involves a software tool that couples personal document expression with the resources of educational institutions and those

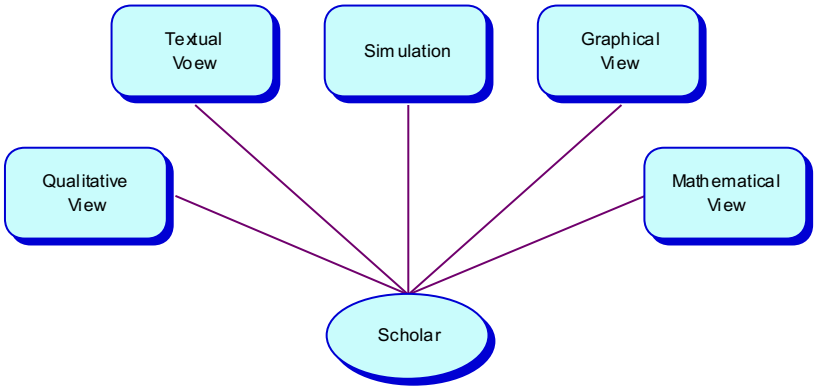


available on the World Wide Web. This support comes in the form of a GUI (see the following screen shot) coupled with an expert system that we discuss in some detail below.

Our model of a scholar indicates that personal documentation typically occurs in a *journal*. The journal collects notes, references, and experimental data including results. The program Mathematica® by Wolfram Research Inc. involves such a journal, oriented specifically toward

mathematicians[1]. The Scholar's Companion is created to expand upon that work, supporting scientists, engineers, and thinkers in a wide range of disciplines. Indeed, the program in a prototype form was used by a student to conduct her research in Materials Engineering. In that project, the program was configured to control a simulation of a polymer curing exercise. Configuration of TSC meant that the student wrote a process model (which we discuss in some detail below), and created a software environment within TSC such that this process model could be applied to a simulation of a live process. Simulation with process rules allowed for the discovery of improved process control rules. The results of that research lead to her Ph.D. at the University of Dayton.

TSC provides scholars with *multiple views* of their study universe. It is through the generation of the initial view by programming and the study of additional views that the scholar is empowered by TSC to



gain insight not previously possessed. Views can appear as logs of system behavior, graphical illustrations of models built (see screen shot below) or taxonomies compiled, or in textual form in "natural language" discussions. Views can also involve linking to tools outside TSC, such as Mathematica®.

Basic Nomenclature

TSC is an Expert System. An Expert System is a kind of computer program intended to perform *reasoning* tasks. For example, the simplest reasoning task is expressed in an **IF-THEN Rule** such as:

```
IF person has food
THEN person can eat
```

Now, if we observe that a person has some food, we can immediately reason that this person can eat. That is an extremely simple example of *deductive reasoning* applying the *data driven method*. Our observation gave us the data, we arrived by deduction to the conclusion.
Suppose we just observe a person, but do not (at first) observe any food. We form an hypothesis we would like to test:

```
Person can eat.
```

---

[1] For the next draft, I will try to past in a screen shot from one.

To test this hypothesis, we bind it with our rule. To do so, we apply a *hypothesis driven method*; we find that a person can eat IF the person has food. We are applying the rule in a backward direction, a process called *abductive reasoning*. In so doing, our inference process will now be forced to find evidence of the existence of food. Depending upon how we configure the program, it could use sensors to make the necessary observation, or it could simply pose the question to us :

```
Does this person have food?
```

The question forces us to perform the observation; just one of the many ways TSC can be programmed to perform its own reasoning tasks. Another way TSC is typically programmed is to arrange some actors on a stage (programmatically speaking). Arranging actors means, for example, that one imagines (and programs) a person to be situated near some food. Then, TSC can notice that this person has food, so our example rule above will be satisfied and the person can eat.

Along the way, in the process of constructing rules, we are dealing with **Concepts**. In our tiny example, we have the following Concepts:

- Person (an object)
- Food (an object)
- Have (possession, containment, near, process)
- Can (capability, enablement)
- Eat (process)

The Scholar's Companion represents all concepts and rules uniformly as **Frames**. A frame is a convenient way to visualize small database tables, one for each concept. Each frame contains **Slots**. Slots represent kinds of **relationships** associated with each concept. Consider the concept of a Human, as illustrated here in the TSC representation:

```
c: Human
      subOf     ( Mammal )
      hasSubs   ( FemaleHuman MaleHuman )
```

Let us consider this example frame in enough detail to get a glimpse into the nature of knowledge representation in TSC.

We first notice the **c:** (pronounced see-colon). This is a TSC *keyword*. It is a *compiler directive*. All compiler directives are discussed in detail later. This particular compiler directive, when read during an Include session, directs the compiler to begin the construction of a Frame with the name set to the word that follows. Once the new frame has been constructed, the compiler automatically begins looking for slots. The compiler always looks for slots as a single word, followed by a left parenthesis, followed by words and parentheses, and ended with a balanced right parenthesis. By looking for balanced parentheses (same number of right parentheses as there are left ones), we are able to continue slot values on multiple lines of source code.

Our example frame indicates that we intend to represent a Human as a member of the set Mammal. We also intend to represent that the set Human includes two sub members: FemaleHuman and MaleHuman. Notice that *slot names* are a single word, and they are case sensitive. Notice that each slot will contain a *set* of *slot values* enclosed inside of a set of parentheses.

We have seen that TSC deals with concepts, some of which can be rules, some of which can be relations, some of which can be objects. Now, what can we do with rules, objects, relations, and so forth? First, and foremost, we can use TSC to construct **Models**. By using the word models, we are implying models of behaviors of some universe of discourse.

Consider, for example, the universe associated with the lives (ecosystem) of phytoplankton and zooplankton in our oceans. We know that zooplankton and phytoplankton are objects, one a member of the set of animals, and one a member of the set of plants. We know that zooplankton will eat phytoplankton. We therefore suspect that it should be possible, given that TSC can build models, to build a model of the ecosystem in which these two objects exist.

Elements of TSC Knowledge Representation

Consider the following discussion. The following example derives from a project constructed by my (then) 7-year old daughter Nefer when she took a profound interest in taxonomic studies, mostly because of their graphical presentations. For the textual representation (the only kind of representation a 7-year old might be expected to implement), she wrote a simple story, as follows:

```
An animal is a living_thing. A mammal is an animal.  A mammal
has hair.  A mammal makes_milk. A reptile is an animal.
A human is a mammal. Nefer is a particular human.
```

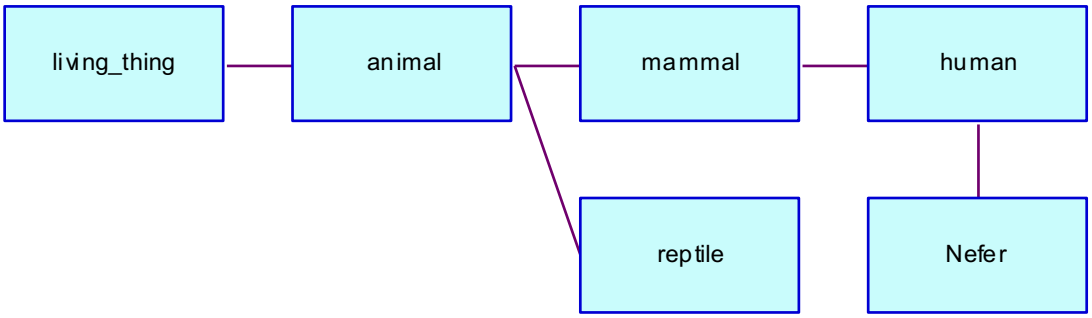
This story was entered in a text editor and presented to TSC for parsing. TSC read and parsed the story, constructing the following internal representation:

```
c: living_thing
    hasSubs      ( animal )
c: animal
    subOf         ( living_thing )
    hasSubs      ( mammal reptile )
c: reptile
    subOf     ( animal )
c: mammal
    subOf     ( animal )
    hasSubs   ( human )
    makes     ( milk )
c: human
    subOf     ( mammal )
    hasInstances ( Nefer )
c: Nefer
   instanceOf ( human )
```

The "`c:`" in each frame is TSC's way of indicating the frame represents a *concept.*. It is added in the `prettyPrint` output method.

TSC uses this internal representation for several purposes, including archival, reasoning, and graphical representation. For that, TSC has an interpreter that can present this knowledge as a

taxonomic tree organized along just about any relationship (slot) requested. I have recreated this knowledge here for illustration, as a taxonomic tree along subs and instances relations.

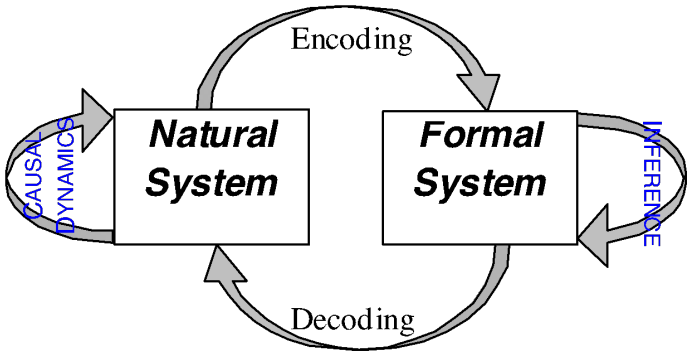


Qualitative Modeling

TSC is specifically a tool for building and studying qualitative models of some universe of discourse. Universes of discourse include the fields of medicine, biology, geology, and so forth. In any universe, the events in an environment are observed. These observations are *mapped* (encoded) by the scholar into a TSC model; the model is studied, and it may be further mapped, by TSC, to a simulator which recreates the dynamics of the universe.

How does TSC support modeling?

A model is constructed by mapping some observations from a natural system into a computer representation called a formal system. To imagine how TSC fits into this scheme, first consider an arbitrary domain, as illustrated next:

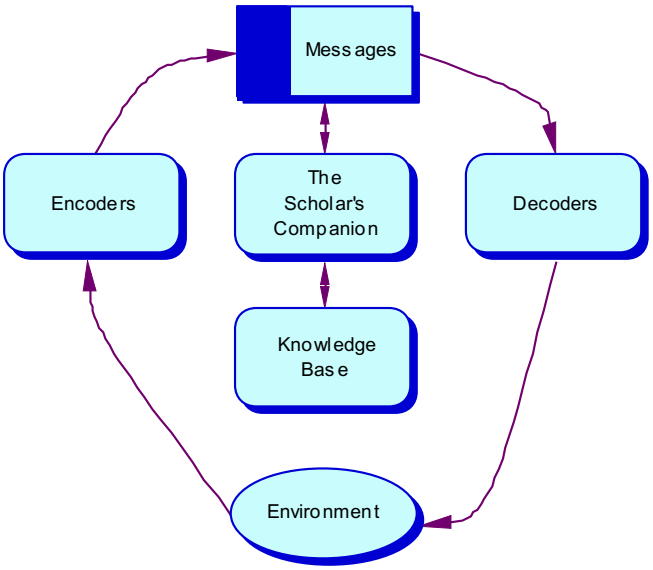


The natural system can be represented by a database comprised of measurements; observations from somewhere in the universe. TSC provides the necessary *encoder* and *decoder* facilities for communication with the environment. TSC further provides the knowledge base facilities necessary to hold the formal system, the model, and to help in its creation. The TSC architecture itself is illustrated next.

We are interested in constructing a formal system – a *model*, as illustrated. Any model is a *formal system* that is to be used as a representation of a *natural system.*

To provide a unified way to think about the construction of a model, TSC adds to this architecture a uniform way to create models, based on the notion of actors on a stage behaving according to a script. This is a *stage metaphor* discussed next. Users build a particular kind of model, a *qualitative model*.

The following figure shows how TSC can be configured when needed to study sensed information from an external environment (e.g. simulator).



What is a qualitative model?

We need to make the distinction between the so-called Qualitative Model, and other kinds of models. First, however, let's consider: just what is a *model*, anyway?

Like the child's model airplane, any model we discuss is a tiny recreation of the "real thing." It may be a physical reproduction (albeit scaled down in size, as in the model airplane) or it may be some kind of descriptive reproduction, as in the mathematical models created by engineers and scientists.

In the qualitative model, we wish to ask questions which are not to be answered with specific quantitative values. We ask: is something changing? Which direction is the change going? In the quantitative model, we ask questions that are to be answered with specific quantitative values: how much? when? and so forth.

Any model built by a scholar considers physical entities, their relations to each other and to their environment, and their states. To facilitate easy consideration of these issues, TSC implements a style of programming based on a stage metaphor.

Modeling Architecture

In QP Theory – one of several variants in the qualitative reasoning universe – we take the view that all *events* (e.g. transactions) may be considered (e.g. represented and reasoned about) as

physical ***processes***. Given this view, we now define any process as comprised of the following descriptors based on a theatrical metaphor discussed in more detail below:

- Actors
- Relations
- States

The variant of QP Theory[2] discussed here accepts the theatrical metaphor: all processes take place on a *stage*, in the same manner as a theatrical production. Thus, actors may exist in different varieties:

- Static Actors, akin to the props on stage
- Dynamic Actors, akin to the players on stage
- Agent Actors, akin to players that do something to other actors
- Patient Actors, players that receive the actions of agents

TSC Actors are different: *that there exists no script with which the actors perform*. Instead, in our model, the actors are guided in their behaviors by process rules, which we discuss below. Relations include all of the sensible relations that exist among all actors on stage. A typical relation might be abutment, implying physical contact between two actors (e.g. a player sitting on a chair).

States are those which exist on the stage. A typical state of, say, an actor might be a mood (e.g. anger, or elation).

The theatrical metaphor is used fairly literally here to illustrate the following point: any physical process can be described and can be the subject of logical reasoning through the use of QP Theory. Logical reasoning implies some form of rules of logic. A script for a play forms the analogous function; the behaviors of players on stage are dictated by two factors: a script, and the various interpretations placed on the script by actors and directors. By analogy, the behaviors of a transaction are dictated by process rules involved in transactions.

From this, we infer the following:

- All processes within the TSC framework are represented in the QP Theory framework.
- The designer (read: Knowledge Engineer, scientist, student, etc.) of any process has the task of *setting the stage*.
- Libraries of QP rules for TSC can be pre-constructed.
- QP rules form the primitive behaviors of TSC.
- Advanced behaviors are constructed by *composition* of QP rules.
- Any process involves one or more of the following:
  1. Change within actors
  2. Change within relations
  3. Change within states

[2] By *variant*, we refer, perhaps, to the notion of a *dialect* of some *language*.

Rules

A rule, in this case, a Process rule or QP Rule, is best visualized as an IF-THEN construct. Many methods exist by which the IF and the THEN constructs are represented. I prefer an "english-like" approach for tutorial purposes. Consider this example:

```
IFActors include Buyer and Seller, and Product, and Money
IFRelations include (Buyer has Money), and (Seller has
Product), and (Buyer wantsToBuy Product), and (Seller
wantsToSell Product)
IFStates include (Buyer isEager) and (Seller isEager)
THENRelations (Buyer has Product) and (Seller has Money)
```

This example illustrates a classical process called a BuySell Transaction. Here, the stage includes two dynamic actors, a Buyer and a Seller, and two otherwise static actors, a Product, and some Money. The actors have relationships and states. The transaction involves changes in the relationships; no change of actors is represented. For completeness, this rule should also change the states, given that each dynamic actor is *satisfied*, perhaps no longer in the *eager* state. It is reasonable to have a group of daemon rules that notice that if the actors are satisfied then change the eager state to false, thus deleting it from the stage.

The Stage Metaphor

Qualitative Modeling with TSC applies the theater as a metaphor for thinking about the modeling process. This simply means that, as one thinks about a qualitative model of some universe (e.g. an ecosystem, a biological system, etc), one imagines that universe being "acted out" on a stage. By applying a stage metaphor, we are now free to refer to the various objects in the universe as *actors*. We are also now free to discuss these actors and their various states and relations to other actors. We can *set the stage* as we discuss next.

Scholars set a stage in order to build a model. Setting the stage means describing the initial conditions, those actors on stage, valid relations and states.TSC is then given a task to construct the qualitative model, given the library of process rules and the initial conditions. That model is called an *envisionment*.

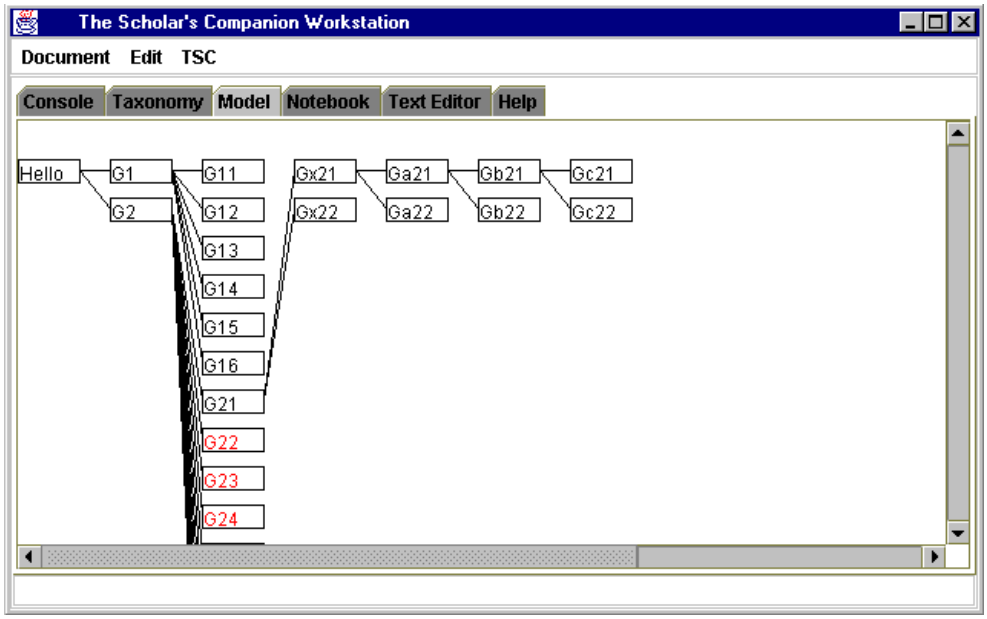
Envisionments

In the big picture, an envisionment is a directed graph that presents the steps a process takes from beginning to end. It can be viewed as a kind of *script* from the stage.

TSC's basic behavior in building an envisionment is as follows:

```
given a source episode
        collect process rule related to the episode
        for every rule which can fire
            clone the episode and update the clone
            spawn a task to expand on the new episode
```

The following screen shot shows an envisionment built as a model based on initial conditions



expressed in the node named "Hello." In some sense, the envisionment represents a *concept map*, one that has been expressed by the firing of process rules. If the rules express the nature of processes in some conceptual framework, then the envisionment maps those rules to their temporal expression within that framework. One begins to realize that concept maps can be built and modified by the construction and tuning of process rules.

Applying The Scholar's Companion to Research

The basis for existence of TSC is simply the support of intellectual exploration. The thesis underlying its creation is that the *apparent intelligence* of a pair (TSC and scholar) can be greater than that of the scholar acting alone. This so called IQ enhancement obtains from two separate benefits provided by interaction with TSC:

- Rapid construction and study of domain models
- Enhanced views

Rapid construction and study of domain models means that the scholar, working from within the journal (normally kept in hard copy form but now a part of the model itself), achieves a more rapid pace of exploration and documentation of the research in progress. Journal notes, references, model design, and model results are all presented together in a single unified document.

Enhanced views refers to the fact that the scholar is now able to present domain information in a qualitative way and see it returned in a variety of other ways, such as trees, graphs, model computations, and so forth. One of the more interesting observations made while teaching TSC to others has been the "Aha" response: "Geeze, they never taught us to think that way in med school."

Applying TSC to a research project simply means doing what one normally does: create a journal, develop a hypothesis, a research plan, sketches related to various approaches to research

methods, and, eventually, model design. Once model design is accomplished, a task is developed which TSC will be asked to perform. Tasks include the construction of envisionments where the research involves the construction of a qualitative model, database mining for evidence in support of a theory or model, or even database mining in search of new knowledge. Tasks have even included the active control of a process simulator while searching for enhanced process control rules.

We like to think that every scholar needs companions. We believe that our project, The Scholar's Companion, is suited to fill one of those needs.