What Should Be Modelled?

Balzer: This discussion is intended to set the stage for further discussions by soliciting AI, DB and PL views of "what should be modelled?" and the nature of models.

[This discussion suffered two problems. First, it was the initial attempt at the workshop by people in different areas to communicate on the same subject. Second, the issue of "what should be modelled?" was frequently confused with "How should it be modelled?" To aid the reader in understanding the issues that were confused, comments by Deutsch, Hayes, Levesque, Rich and Sibley have been extracted and placed at the beginning. ed.]

Kinds of Knowledge

Deutsch: There are three different kinds of knowledge each of which can be modelled.

- knowledge about the real world,
- knowledge about our representation of the real world, and
- knowledge about the operations of the system being used to manipulate these models of the real world.

The statement "Spot is a dog." clearly refers to entities in the real world. Since the real world cannot actually be put into a computer, we always use symbols (in the computer) to represent real world entities. When we make statements like "Spot is a dog." or "A dog has four legs." we deliberately confuse symbols with subjects in the real world. That is just fine in my opinion.

The second kind of model or knowledge concerns properties of a representation, e.g., the variable "color" can take on only one of the values "Red," "blue" and "green." This is not a fact of the real world; this is a fact about a representation of the real world. We must clearly distinguish between things that are means to represent the real world directly and things that are meant to only talk about representations of the real world.

The final kind of knowledge we model concerns models of how a system operates. What kind of operations can we perform? How does the system respond to various things that we do? What kinds of pragmatic information do we give a system so that it will complete its function.

[The following questions highlight some of the issues that were not always distinguished in the following discussion. ed.]

Hayes: Are we discussing the modelling of programs, which are representations of some world, or the modelling of the world itself?

Levesque: Are we talking about what is and what is not allowed in the real world or in the computer system?

Rich: We should distinguish features of the world to be modelled from techniques used to represent those features. Examples of modelling problems concerning the real world are what are objects (e.g., water, steam, the atmosphere, a table)? What objects can be created and destroyed? Can objects overlap? Corresponding to these features we require representation techniques such as message passing, hierarchical and relational database models, and pre- and post-conditions for operations.

Sibley: Are we talking about the problem or the solution?

[The discussion begins here. ed]

Properties of Models

Balzer: What are some desirable features of models? Should a model (e.g., specification) be operational, i.e., executable given a suitably powerful interpreter? This aids in analyzing the model and predicting its behaviour, probably at the expense of efficiency. Should a model support mappings, i.e., multiple views or perspectives? Although most of us would answer yes, there is little agreement on how to support maps. Finally, should a model contain meta knowledge, descriptions of the modelling mechanism itself, that can be queried and modified, e.g., What are the attributes of an object? What relationships can an object take part in? What constraints pertain to an object?

Properties of Objects

Balzer: Three fundamental components of models are objects, states and transitions. Properties of objects can be defined using structures, relations, operations and messages. These means can be used to define objects, relate objects and form associations of pieces of an object.

Hayes: Are these symbolic objects in a computer or are you talking about the real world?

Balzer: These are objects inside the machine used to represent our perception of aspects of real

world objects. In any problem domain there are identifiable objects which we choose to describe with one of the four mechanisms.

Weber: Since relations can be used to model objects and relationships, is there a duality?

Balzer: In my modelling system an object is denoted by a symbol and defined in terms of the relationships it participates in. Other systems provide a richer set of mechanisms to define objects. For example, a set of characteristics of an object is represented in a frame-based system by a set of slots. In addition frames can be related. An important question is whether there is one way or multiple ways of defining all you know about some objects.

Levesque: You seem to be talking about modelling mechanisms not "What should be modelled?" You use the term object to construct the model and not to define what you are modelling.

Balzer: To answer "What aspects of the world need to be captured?" I have proposed four ways of capturing two aspects: objects and their relationships.

Hayes: You are talking about particular ways of programming symbolic objects in the machine. That seems to have very little to do with the kinds of world we are describing. Again there are two different realms: the world we are trying to describe and the ways of describing it.

Balzer: You are right. We could focus on what should be in the model without describing the mechanisms employed to do it. I am addressing properties of models in the machine.

Properties of States

Balzer: The existence of objects results in states. One important question about states is: "Is all information global?" In a quiesed state is all information in the same space or is it somehow partitioned? It seems to me that for modelling we need a global state description.

Hendrix: What kind of dimensions do you have in mind? What is local as opposed to global?

Balzer: To me there is a difference between the system and the flow of information in the system. Assuming the system is quiesed, how do you get information out? Typically, one defines a mechanism that allows you to interrogate pieces of the system to get specific facts. What are the ways of retrieving information that are tied to the structure of the information about those objects? In a lot of specifications I see access path representations which is inappropriate since representation is an implementation concern. So, I am trying to identify whether at the information level access paths are part of the model or whether they are part of the implementation of the model. If we evaluate a model in terms of its expressive power then access paths should not be in specifications.

Other questions about global information are: Should you be allowed to use quantification to express queries over the entire state of the information model? Is there non-determinism in this information model? Should the information model include derived information, that is, inference rules? Should the model support privacy, viewpoints and perspectives?

Properties of Transitions

Balzer: Transitions are composed from primitive state to state changes. Modelling transitions concerns historical and future reference, transition prohibitions, atomicity of operations and agents of transitions, e.g., the environment (DB), the program (PL) or the system (AI). Currently there is considerable interest in modelling systems with multiple, concurrent agents with multiple activities per agent.

Constituents of Machine Models

Balzer: In summary, a model consists of objects, relationships, states, constraints (e.g., often represented using types), actions which cause state transitions, and agents which instigate actions.

AI, DB and PL Characatures

Balzer: AI, DB and PL have emphasized different aspects of models, hence have different answers for "What should be modelled?". To highlight the differences, I will overstate all three perspectives. AI focuses on primitives which the system (the agent) combines to achieve the defined goals. DB focuses on the data problem, i.e., changes in data structures, and ignores processes that cause change. DB people treat the agent as being outside the system they model. However, individual operations, e.g., insert, update and delete, are provided. The PL approach is to focus on operations which implicitly describe the data which is given no inherent structure of its own. This crude characterization leads to very different types of models. I would like to see the three perspectives closer together.

Extracting Constraints

Codd: There is some truth to your DB characterization, however, it is still misleading. In DB, we try to take out of application programs much of the meaning of data that is typically buried in those programs. Part of what is buried is called integrity constraints. Hence, we do not ignore the processes. We try to reduce them by extracting meaning that can and should reside in a description of data to be shared by all programs.

Balzer: There is more overlap than my characature allowed. The extraction of constraints that you mentioned for DB is becoming critical in each approach to describing what is to be modelled. Some examples are rules that operations and objects being modelled must obey, e.g., the laws of physics.

Modelling Freedoms

Balzer: If a feature of the real world, e.g., memory of past events, can be modelled directly using the systematized semantics of some modelling tools, the tools can be said to support modelling freedom for that feature. Model builders are free from constructing their own mechanisms. In asking "what should be modelled?" with respect to modelling tools, one asks "what modelling freedoms should be supported?"

One desirable modelling freedom is that constraints be understood and maintained automatically by the system.

A second freedom is the notion of an information model. It is desirable to have a uniform method of accessing information about a given state without having to specify explicitly how the model answers queries about itself.

A third freedom is an inferencing capability. DB provides mechanisms to retrieve information in the database and supports a small amount of derivation. AI permits arbitrary inference rules to be stated and provides a control mechanism, called heuristic search, to find ways of limiting the search needed to draw conclusions.

A fourth freedom is knowledge of past events. Should model builders be required to create auxiliary data structures to handle historical reference?

A fifth freedom is non-determinism. Should people have the freedom of being less than precise in selections thereby getting back multi-valued results?

A final freedom in my list is hypothetical activity, ways of trying things out to see what the effect is before actually making a design decision.

Rich: Another way to consider modelling freedoms is to determine what real world features are to be modelled, e.g., continuing processes, synchronization, emotional relationships between people, three dimensional geometry.

How Much to Model

Balzer: If we ignore implementation details we can focus on what constitutes a model of an application. How broad is the scope of the model: the application itself, the application environment, the system and its environment? These questions depend on the purpose of the model.

Purposes of Models

Balzer: The purpose for which a model is constructed determines what should be modelled. Four typical purposes of models are: simulation, understanding, specification and implementation.

Functional versus Representational Models

Balzer: I believe that models can be both functional and representational, i.e., there is a continuum between specification and implementation. It is not enough to focus merely on specifications or the WHAT. One can give a functional characterization of something and integrate more and more representation detail to get a full implementation. However, it is important at any point in making a description to understand whether an aspect is specificational or implementational. On one hand, there would be a large difference between an axiomatic specification and a corresponding implementation. On the other hand, given the appropriate "engine," some high level specifications could serve as implementations. Since such "engines" don't exist one can gradually translate specifications into more representational descriptions.

There are many places in the continuum at which you can make the tradeoff between a high level description and implementation efficiency. One problem with many current models and specifications is that they are hard to understand.

Hardgrave: Is a formalism which is easy to understand but somehow inconsistent, acceptable for specification?

Balzer: There is a lot of work directed at improving understandability without compromising precision. One method is to layer or modularize specifications so that you can understand a small piece at a time.