Effective Description of Enterprise Processes

Roger Burkhart
Deere & Company
John Deere Road
Moline, IL 61265
roger@ci.deere.com

Position Paper
Workshop on Specification of Behavioral Semantics in Object-Oriented Information Modeling
Workshop on Object-Oriented Reflection and Metalevel Architectures
OOPSLA '93

DRAFT
July 31, 1993

1. Introduction

This paper proposes that a common set of primitive modeling concepts be adopted to express and specify the behavior either of enterprise objects (belonging to an application domain) or computational objects (objects built in software and residing inside a computer, but typically presumed to represent or correspond to some sort of external domain objects). The applied world of enterprise or business process modeling and the independent world of formal software specification each have significant elements to contribute to the other. The remainder of this paper identifies some of these elements, and outlines a unified approach for describing behavioral phenomena in either domain.

The world of business, including as major subdomains the organization of people and tasks, the management of economic abstractions such as money and contracts, and the technical analysis and realization of physical processes and artifacts, is extremely complex. Existing approaches to modeling a business capture only a fraction of the complexity that exists in any real business domain. Rather than describing or prescribing the real activities of the business, information modeling has typically retreated to defining only static structures of information that can be recorded as periodic snapshots over some limited portion of the business state.

The object-oriented perspective contributes a key element to the problem of business description: the principle of behavioral encapsulation. The only relevant characterization of an object, as viewed by agents external to that object, is one framed entirely in terms of observable behavior. A formal description of business objects, which assigns intended meanings to each of its terms and grammatical constructions, must ground itself in primitive phenomena manifested at the interfaces between objects. Software objects can be regarded as a special case of business objects, since technical artifacts built inside a computer are within the domain of objects a business deals with.

2. Describing Business Complexity

Extending the scope of behavioral description to the unbounded complexity of a real business domain is instructive for many reasons. In particular, it forces recognition that many elements difficult or intractable to deal with in existing formal frameworks are nevertheless essential, unavoidable realities that must be addressed by a general descriptive framework. Among these realities are the following:

- Concurrency. In any complex domain, many things are happening at once. A general framework must deal with concurrency and the autonomy of concurrent processes at its outset. Objects in general are active; they encapsulate autonomous processes that accept and generate threads of interactions across an encapsulation boundary. Objects can also be internally concurrent; their encapsulation boundary can surround a collection of concurrently interacting subobjects.

- Partial information. No part of the system can have information about any other part of the system, unless specific interactions have occurred to transfer that information (either directly or by inference out of information that has been transferred). Channels or mechanisms for information dissemination must themselves be described in the characterization of the total system. There can be no assumptions about global time or synchronization; synchronization must be mediated by mutual events that establish a common timeline.

- Visibility and connection. For reasons of scalability, a process can directly affect or query only a finite set of other processes. The ability to reference or affect other processes must be inferred out of this network of direct connections. This basic network can be extended by processes that serve as mediators for others, creating a network of mediated interactions that is typically much more densely connected than direct interaction. Modes and layers of interaction, together with all processes that enable them, are fundamental to understanding the potential for action within the system.

- Creation and destruction. A process not only has limited knowledge of other processes it can interact with, but the collection of these processes is not fixed. Processes can create or destroy other processes, and the ability of any process to communicate with others can change continually during its lifetime. A general framework for system description must account for the highly dynamic character of process configuration, and cannot assume any closure over the rest of the environment which any carved-out portion might eventually encounter.

- History. The action of an encapsulated process can depend in general on any part of its history, including every interaction in which a process has participated since its creation. An external observer cannot assume any restriction on the ability of an object to remember any part of its history using whatever forms of internal representation the object might employ. The ability of the past to affect the future cannot in general be captured by any finite state summary. The history of an object is bounded on one end in that all interactions are successors of a single creation event, but is unbounded on the other in that an object may continue to participate in further interactions as long as it exists.

- Reflection. The external behavior of an object can be a function not only of its history and an initial set of constitutive rules, but of internal processing that extends or evolves any initially predictable behavior. Objects can process representations not only of their history, but of their own capabilities, and they can compile and apply these representations to become effective over their own subsequent behavior. Monitoring and modification of behavior is a process that feeds on its own results, enabling a continual refinement of behavior into more complex forms. Indeterminism, in the most extremely surprising, creative, and novel ways, saturates the interior of a general object.

- Ontology. A formal language or framework says nothing without a commitment to the meaning or interpretation of its terms and formulas. The word ontology has recently been adopted to refer to the conceptualization, or term definitions, expressed by a particular formal system. As domains expand and begin to overlap, particular ontologies must be integrated with each other. Progressively deeper and more comprehensive ontologies are required to accomplish this integration. Ultimately an ontology is needed in the true philosophical sense of this term: a systematic account of everything that can possibly exist, including an ordering of inherent dependencies. Object-oriented description begins a commitment to one such ontology: it proposes that the world be understood in terms of the phenomena or appearances generated by active elements that observe, affect, and control each other, either directly or through their appearances.

One way of thinking about the problem of business description is that it must deal with the entire program of distributed artificial intelligence (DAI), which studies problem solving and general behavior in systems of intelligent agents. To cover a more complete domain, the particular focus on problem solving and intelligence must be extended to material worlds and apparently unintended phenomena. These too might be explainable in terms of interacting elements possessing only primitive behavior and yet leading to approximations of intelligence in greater or lesser degree. Across such levels, an open challenge is to develop a consistent account of intention or agency: to explain what it means for an object to act so as to seek or cause some outcome.

Object-oriented programming sometimes suffers a vague disrepute arising from a tendency to anthropomorphize objects. Perhaps this taint can be dissipated by asking instead what it is that animates an object. The ability of an object to act, and to participate in actions with other actors, is deeply important to the core of what an object is. It usually defines why we draw the boundaries of an object where we do and label "these" phenomena as belonging to "this" object.

3. Using Formal Models

While the business domain points the way to a tar pit of inextricable complexity, the world of formal systems, and formal software specification in particular, offers a more comforting world of carefully limited and mastered complexity. Both worlds are important, because many phenomena of business naturally limit themselves to special cases of the general model of complexity suggested above.

Mapping the features of a general model may well serve needs of both domains. A commitment to a comprehensive scheme for describing all possible phenomena, no matter how extreme or intractable the resulting complexity, results in a simpler and more adaptable foundation than approaches that just plunge into the middle of the scale of complexity without concern for eventually meeting up with other important entry points.

Formal systems, like almost any other artifact, are contained in the domain of business objects (business exploits anything potentially useful to its aims). Their effective application requires that they be understood in the context of their intended use, operation, and semantic interpretation. Deterministic machinery, and its possible interpretation as formal language, has many potential roles in the operation of a business. Business puts a premium on predictability and control over many of its subprocesses, and therefore puts in place many prescriptive structures that deliberately exclude any unplanned role for complexity or contingency.

A special category of deterministic machinery is designed to carry a weight of semantic interpretation, so that it can serve as formalized language. Formalized language is important for at least two basic reasons: to help people establish precision and avoid ambiguity (and perhaps achieve better understanding as a result), and to allow automated processing of language structures, either to perform inference and simulation, or to establish causal interaction between the formal system and some component of its environment it is intended to track or control.

First-order logic is a particular formal language that serves as a final arbiter of precision and avoided ambiguity. Its machinery is cumbersome for direct use in most applications, but any other language should be able to specify its mapping to the neutral form of first-order logic to stake its own claim to formal status. The selection of predicate logic as a basis for formalizing the conceptual schema of a database [1] or for knowledge interchange [2] is entirely appropriate, since no more complete or inherently capable method for formalizing the intended meaning of a static communication has ever been found. More esoteric forms of logic, such as modal or higher-order logics, can be interpreted by expanding an initially first-order universe to reflexively include components of the formal logical system itself, such as predicates, expressions, and constructs devised specially by an agent to organize and manage representations built within the formal system.

To ensure sufficient extensibility, the semantic framework of first-order logic needs to include not only the objects belonging to a universe of discourse, and the formulas in a language, but also a subject who constructs and interprets these formulas. Potentially this subject-oriented semantics might have as much or more to say about the process of knowledge acquisition by a subject (its purposes or expectations, the historical or contextual situation, effect on existing beliefs, etc.) as it has to say about the external objects about which knowledge is being acquired. This semantics would also address the process and mechanisms by which a knowledge-organizing agent establishes its presumed relation to the environment where it observes and acts.

4. Models for Specification

One salutary characteristic of first-order logic is its neutrality with respect to ontology: any scheme for describing a universe of discourse can be supported by suitable definition of terms. Once a commitment to basic ontology is made, however, one of its challenges it is to explain the subject-object interactions out of which logic and linguistic description arise. These include the prescriptive use of representations by subjects who impose plans and specifications on their environment.

An ontology of behavior and process, by situating logic and its semantic relations in a pervasive setting of interacting processes, points to a more general way of thinking about formal specifications of all kinds. Each such specification can be regarded as a blueprint for a machine, that if fed information about all interactions occurring in some domain, would continually maintain a single result value signaling that the interactions either do or do not conform to the specification. All forms of precise behavior specification could then be interpreted as indirect but equivalent ways of specifying such an observer machine. The machine need not actually be implemented to be useful; it could serve simply as a model by which to interpret statements of other specification languages.

An indirect specification of behavior, in terms not of the behavior itself but of a process designed to review and test behavior, allows the statement of rules and conditions that can be stated in no other way. Additionally, it can both specify and implement the interactions by which a specification can be made causally effective over the behavior it intends. Abstract models of computational processing can be employed to avoid overspecifying the details of a particular computational machine. Whether implemented or abstract, the process that has the most powerful ability to review behavior is one that can also simulate the kinds of behavior it expects to observe. The most general model for enterprise simulation would consist at least of a dynamically configuring community of interacting agents busily processing their histories and replacing their capabilities in a fully reflective way. Such machinery could establish running models of desired enterprise behavior (for example, complex problem-solving processes) to establish performance and quality criteria that could be established in no other way.

A simulation of interacting agents that includes the reflective ability to modify its own configuration and processing is the most expressive model by which to monitor and control enterprise behavior. An initial and very sketchy outline of such a model was offered in a previous paper [3]. That paper also briefly discussed and referenced some related work, such as the Actors model of distributed computation [4]. Details and major features of the proposed design continue to evolve, but the overall program - an abstract and general model of concurrent computation by which to specify enterprise behavior - continues to appear feasible. Whether it would ever succeed in mastering the real complexity of a business (which includes the routine confrontation of unprecedented circumstance) remains questionable.

References

[1] *Concepts and Terminology for the Conceptual Schema*, Technical Report 9007, International Organization for Standardization, Geneva, Switzerland, 1987

[2] Genesereth, M. R., Fikes, R. E., *Knowledge Interchange Format: Version 3 Reference Manual*, Logic Group Report Logic-92-1, Computer Science Department, Stanford University, Stanford, CA, 1992

[3] Burkhart, R., "Process-based Definition of Enterprise Models," in *Enterprise Integration Modeling: Proceedings of the First International Conference* (Charles Petrie, ed.), MIT Press, Cambridge, MA, October 1992.

[4] Agha, G., *Actors: A Model of Concurrent Computation in Distributed Systems*, MIT Press, 1987