

Object-Oriented Programming and Objectivist Epistemology: Parallels and Implications

Adam Reed

1. Introduction

Several textbooks (Booch 1994; Jacobson et al. 1992; Rumbaugh et al. 1991) on Object-Oriented programming (OOP) refer the reader interested in the epistemological foundations of this discipline to Ayn Rand's *Introduction to Objectivist Epistemology* ([1966–67] 1990). At the 1995 Summer Seminar of the Institute for Objectivist Studies (now The Objectivist Center), Robert Hartford presented a participant paper on Objectivist epistemology and OOP, focused on Ayn Rand's use of "Conceptual Common Denominators" and analogous application of the techniques of *inheritance* in OOP. Readily apparent correspondences between several aspects of OOP and Objectivist epistemology have led to extensive but not always well-informed speculation, in Internet Web sites and newsgroups, about their relationship. To resolve these issues, I asked Alan Kay and other pioneers of OOP about the historical facts. The results of my queries, and related observations about the cross-applicability of logical techniques based on Objectivist epistemology and OOP, are detailed below.

2. What is Object-Oriented Programming?

2.1. Programming Languages as Tacit Models of Causation

One of the formative events at the beginning of modern experimental science took place in 1588 when Galileo Galilei's father, musician and music theorist Vincenzo Galilei, asked the young Galileo about the relationship of tension on a string to its frequency of vibration. Galileo found differing formulas in the works of

Aristotle and Archimedes, and resolved the question by experimental measurement.¹ Archimedes turned out to have been right. That Aristotle was wrong, about a fact that could have been readily established by direct reference to reality, cast a long shadow on the potential value to science of Aristotle's ideas, including Aristotle's account of causality. In the words of Taylor (1976, 57), post-Galilean science had "a pronounced tendency to eschew Aristotelian concepts wherever possible. There have not been wanting scientists and philosophers who have insisted that the very concept of a cause is quite worthless, being 'anthropomorphic' in origin and in any case replaceable by such less esoteric concepts as concomitant variations, invariable sequences, and so on. Bertrand Russell declared the concept of cause a 'relic of a bygone age. . . .' Willard van Orman Quine wrote that 'the notion of cause would have no place in a suitably austere ontology.'" This was a standard view in post-Humean empiricism. Causation, like consciousness, became invisible to the very people who were studying it, hidden under cognitively abstruse but "philosophically correct" substitute labels.

Avoiding the vocabulary, if not the fact, of causation is particularly difficult in applicable science. Scientific engineering, including software engineering, is concerned with finding the optimal efficient causes for one's desired final cause. Even *stating what their discipline is about* becomes difficult if engineers try to avoid the concepts of causation. Paraphrasing Cheswick and Bellovin's 1994² observation about network security policies, everyone has a mental model of causation; those who think that they don't have one use theirs anyway; they just don't know what their model is.

The implicit model of causation used by scientists and engineers since Galileo, and possibly since Archimedes, might be called the event-state-event-state-event model. In this model, events change the state of reality; states of reality, in turn, trigger events, which change the state of reality etc. In some applications of the event-state-event model, the intervening states are made implicit, so that events are said to cause events that cause subsequent events in an endless chain. This is the event-event process model, also called the "billiard model" of causality. Or one might make events implicit, and model only the

transitions from one state of reality to the next. In applications of discrete mathematics this is called the "state machine" model.

2.2. Procedural, Applicative, and Declarative Languages

Every programming language, and every conscious methodology of solving software engineering problems, is based on one or more model(s) of causation. The processing units, from which computers are built, are designed to carry out a sequence of actions or steps, and early computer languages (such as Fortran, developed by John Backus at IBM between 1954 and 1957) specified abstract descriptions of event sequences. The state of data, used to represent the state of reality in computer programs, was used occasionally to decide what action was carried out next, but otherwise "state" was implicitly treated as a result, but not a cause, of sequenced events. The programmer used the programming language to specify the procedure—that is, the sequence of actions—meant to achieve the desired result. Computer languages based on the event-event implicit model of causation (Fortran, Algol, Basic, C, etc.) are called *procedural* languages.

As the range of problems tackled by software engineers expanded, it began to include situations in which computer programs emulated, replaced, or assisted humans in tasks for which human cognition uses a mental representation of the aspects of reality that the human is manipulating. Programming languages in which actions are applied to data representing the current state of reality are called *applicative* languages. Applicative languages explicitly implement in full the tacit event-state-event-state model of causation. Starting with the artificial intelligence list-processing language Lisp (developed by John McCarthy and his students between 1956 and 1961), applicative programming languages include the extended AI language Scheme, teaching language Logo, matrix-crunching language APL, machine and experiment control language Forth and others.

In specifying a task to a human assistant, the requestor seldom specifies the assistant's task in terms of events or actions, or even in terms of actions explicitly applied to states of reality. The assistant's task is normally described as a state of reality desired by the requestor

(e.g., “I want a report on accounts receivable”), perhaps relative to a previous state (e.g., “with updated billing information”). As computational algorithms to carry out desired changes in state were identified, it became possible to define computer languages in which the programmer, or user, declares only the desired changes in the data representing the state of some aspect of reality. The declared specification is then achieved algorithmically without additional human input. Such languages, which implement the “state machine” abstraction of the state-event-state tacit model of causation, are called *declarative* languages. Declarative programming languages include the report program generator RPG (IBM 1964), string processors (RegExp), formula languages such as Prolog, and markup languages (troff-mm, SGML, HTML, and XML).

The fourth category of programming languages rooted in the tacit event-state-event model of causation consists of *eclectic* languages, such as COBOL (1960) and PERL, which incorporate abstractions for performing different aspects of the same programming task with procedural, applicative and declarative sub-models. Eclectic languages are designed to incorporate new approaches as they evolve, and recent versions of COBOL and PERL also include the abstractions needed for object-oriented design, discussed below.

2.3. Aristotelian Causality and Object-Oriented Languages

In categorizing computer languages, the term “Object-Oriented” pertains to those languages in which states and actions do not appear by themselves, as primitive building blocks of processes and programs, but rather, and only, as attributes and actions of entities. The basic building block (and in strictly object-oriented languages the only stand-alone primitive abstraction) is the *Object*, which models an entity. Actions (called “methods”) are always the actions of an entity (or of a class of entities). In contrast to procedural, applicative, and declarative programming languages, which are based on the tacit event-state-event-state conception of causality, object-oriented languages implement the Aristotelian view of causality: that entities act according to their identity. OOP is programming in accordance with what Rand (1957, 954) called the law of causality: “The law of

causality is the law of identity applied to action. All actions are caused by entities. The nature of an action is caused and determined by the nature of the entities that act; a thing cannot act in contradiction to its nature.”

In spite of similar causal models, Rand did not actually influence the initial development of OOP (see History and Questions of Influence, below.) But the fact that OOP is “programming in accordance with the Law of Causality” has implications that include the applicability of the logic of OOP to the practice—“logic as the art of non-contradictory identification”—of Rand’s epistemology.

3. Matching Abstractions

When a reader familiar with Objectivist epistemology encounters OOP or when a programmer familiar with the abstractions of an object-oriented language such as Java encounters Objectivist epistemology, it is difficult not to notice an exact mapping between the foundational abstractions of the two systems (Table 1):

Table 1

Abstractions of Object-Oriented Programming	Abstractions of Objectivist Epistemology
Object Class	Concept
Object Instance	Unit
Member Method of an Object	Potential Action of an Entity
Invocation of a Member Method	Actual Action, Event
Arguments of a Method	Context of an Action
Member Variable of an Object	Attribute of an Entity
Value of a Member Variable	Measurement
Static Member	Characteristic
Interface	Conceptual Common Denominator

The correspondences above are architectural rather than ontological. Ontologically, all the abstractions of OOP refer to representations. Of the abstractions in Objectivist epistemology, only concepts are representations of reality in the mind. Units are objective in Rand's specific sense of a relation, analogous to Gibson's (1972; 1977) *affordances*,³ between human perception and its object in (usually external) reality. Finally, actions, attributes, measurements, etc. are out in the world. An ontological correspondence would match representations with representations, i.e., classes with concepts but instances with *knowledge* of units, methods with knowledge of actions, variables with knowledge of attributes, etc.

Just as in Objectivist epistemology a concept subsumes a class of units characterized by analogous actions and attributes, a class in OOP is characterized by a list of member methods and variables. For example, the concept of "chair" might be characterized like this:

```
class Chair
{
    float   heightOfSeat;
    float   depthOfSeat;
    float   widthOfSeat;
    int     numberOfLegs;
    bool    hasBack;
    // .... other member variables ....

    Chair()
    {
        // .... actions to make a new chair ....
    }

    support(Person sitter)
    {
        // .... how a chair supports a sitting person ....
    }

    tipOver(ForceVector impact)
    {
        // .... how a chair tips over when hit ....
    }

    // ....
}
}
```

Thus, the characterization of a class (concept) starts with

variables for the attributes (in this case heightOfSeat, depthOfSeat, widthOfSeat, numberOfLegs, hasBack, etc.) of an instance (unit) of that class, without specifying their specific values, which may vary from instance to instance. This works exactly like *measurement omission* in the Objectivist account of concept specification. The listing of member variables is followed by listing the "member methods" (actions) of which instances of the class (units of the concept) are capable, etc. This correspondence goes beyond the implicit models of causality and the relationships among fundamental abstractions. Most specific principles of Objectivist epistemology map exactly to the corresponding principles of OOP. For example, the epistemological principle that "each omitted measurement of a concept must exist, for each unit of that concept, in some quantity, but may exist in any appropriate quantity" (expanded from Rand [1966–67] 1990, 18) maps exactly to "each member variable of an object class has, in each instance of an object of that class, some specific value, but it may have any appropriate value." When this correspondence is not already exact, the statements of analogous principles of the two systems gain precision when qualified and expanded, as in this example, to match each other exactly. Interestingly, Rand anticipated the OOP terminology of "Object" and "Method" in reference to representations of entities and actions. In *Introduction to Objectivist Epistemology*, "object" refers to the counterpart, in reality, of the referent of a mental process such as emotion or cognition, e.g., "[t]he *object* may be a thing, an event, an activity, a condition or a person" (34). Similarly, "[c]oncepts of method designate systematic courses of action devised by men" [e.g., programmers] "for the purpose of achieving certain goals" [e.g., representing the course and results of some entity's actions] (35).

4. History and Questions of Influence

4.1. Pre-history

The history of OOP languages goes back to the design of SIMULA 1, Dahl and Nygaard's 1966 extension of a procedural language, ALGOL, to support simulation of systems with discrete

probabilistic events. Pre-OOP languages, including procedural languages such as ALGOL, supported the event-state-event model of causality embedded in classical physics from Galileo to the discovery of radioactivity by Becquerel in 1895. Becquerel's discovery marked the beginning of the end of classical (that is, event-state-event) models in physics. Radioactive nuclides, and other systems with discrete probabilistic events, proved surprisingly difficult to represent in the event-state-event model. To accommodate the description of systems with discrete probabilistic behavior, the preservation of event-state-event causality required the invention of "hidden" (that is, intrinsically unknowable) state variables that made system descriptions complicated and unwieldy, without any benefit in coherence or predictability.

The difficulties of fitting the discoveries of modern physics into the previously dominant (or "classical") event-state-event model of causation were explored by Whitehead (1929), who proposed an alternative ontology, complete with his own model of causation. While Kay mentions Whitehead in his correspondence (see Appendix A), Whitehead's complex ontology never found any actual application to the work of contemporary science. In any case, Whitehead's ontology, which posited events, and processes of events, as primary existents, is not likely to have influenced the development of a programming paradigm in which events are represented only as "methods of objects," that is, actions of entities.

To many physicists, the event-state-event model used in "classical" (that is, nineteenth-century) physics seemed coextensive with causality as such. Whitehead's identification of basic incompatibilities between modern physics and event-state-event causality led physicists to disclaim the very notion of causality, or to reject the knowability of causes. In the new, "acausal" physics, it was simply the nature of some nuclides to decay with a fixed probability per unit of time. In fact, the new physics did have an implicit model of causality; it was only that physicists did not recognize the model of causality they were now using. It was Aristotle's causality: the nature, fact and probability of the nuclear decay event, were caused and determined by the nature of the entity—the radioactive nuclide—whose action it was.

By conforming to the physicists' new way of thinking about systems with discrete probabilistic events, SIMULA actually implemented the representation and simulation of Aristotelian causality in a programming language.

4.2. SIMULA 1.

SIMULA 1 (1962–1965) added to ALGOL, an otherwise straightforward procedural language, two programming constructs required for modeling Aristotelian causality: the *class* and the *virtual procedure*, the latter being a precursor of what in modern object-oriented languages is called a *method*. Except for the availability of those two abstractions, SIMULA 1 was just ALGOL. The user of SIMULA 1 was expected to use the event-state-event causal model of classical physics for everything except modeling discrete event systems.

A SIMULA 1 "class" was actually what is called today a *class constructor*: a procedure to allocate storage for information about an entity (object instance) belonging to the specified class, and to perform actions necessary for the simulation to reach the state representing the existence and activation of the object instance. A "virtual procedure" was a member variable specifying the behavior of the object instance in a (deterministically or probabilistically) specified context; the value of this variable was an actual procedure defined elsewhere.

In 1967, Dahl and Nygaard used feedback from experience with SIMULA 1 to design SIMULA 67, which pioneered subclasses and inheritance from class to subclass. SIMULA was still, and remains, an object-oriented extension to a basically procedural language. It lacks explicit destructors (destructors, which are discussed later in this article in connection with scope tracking, would have been difficult to integrate with a procedural language such as ALGOL) but otherwise contains all the expected abstractions of subsequent "pure" object-oriented languages.

SIMULA 67 also has an interesting hint that Dahl and Nygaard knew they were enabling computer simulation of Aristotelian causality. While the names of all the other operators in SIMULA, as

in ALGOL, come from the English, one operator uses the Latin "qua" rather than the English "as." This recalls the neo-Aristotelian scholastics, who were fond of saying not merely that an entity has some characteristic, but that it has that characteristic *qua* an instance of a specific class. In SIMULA 67, the "qua" operator indicates that an attribute or action should be applied from its default value in the specified class, rather than from its specific value in the designated object instance. So if "American" is a subclass of the class "Man," and John is an instance of the class American, then one might write, in SIMULA,

```
John.values(baseball);
(John qua Man).values(freedom);
```

If Dahl and Nygaard were indeed aware of SIMULA's connection to the Aristotelian conception of causality, this awareness would have been very unusual for their time. In the 1960s, physicists still wrote of modern physics as "acausal." Our current identification of the causal model of modern physics with that of Aristotle only dates to Harré 1970 and Popper 1982. SIMULA 67, with minor additions and changes, evolved into the current standard SIMULA language.

4.3. Objects

Although SIMULA 1 is generally acknowledged as the precursor of today's OOP languages, no one connected with the SIMULA project used the term "object" until much later, after that term was made popular by Smalltalk. (Early SIMULA documents refer to what are now called "objects" as "records" or "structures" produced by invoking a class, the class being an initialization procedure; see above.) The first use of the term "object" in connection with software is due to Seymour Papert, at MIT, in the late 1960s.

Seymour Papert, a mathematician, worked with Jean Piaget at the University of Geneva from 1958 to 1963. He then came to MIT, where he collaborated with Marvin Minsky in research on mathematical psychology and artificial intelligence; they co-founded MIT's Artificial Intelligence Laboratory in the late 1960s. His interest in

extending Piaget's work on cognitive development led Papert (1980) to develop a highly simplified applicative language, LOGO, to experiment with the teaching of mathematical concepts through programming. In 1968–69, LOGO was used in place of the normal mathematics curriculum by a small seventh-grade class at Muzzy Junior High School in Lexington, Massachusetts (218). Although these students did manage to learn some concepts and write a few programs, the lack of a concrete referent for program abstractions was clearly an obstacle; it promised to be an even greater obstacle for younger children with whom Papert was planning to work in subsequent years. Drawing on experiences from his own childhood, when he discovered what he later found were basic mathematical concepts by playing with toy gears, Papert came up with the idea of "objects to think with"; he was experimenting with both actual objects under computer control, and visual representations of program-controlled objects on the computer screen (11). Papert eventually settled on a computer-controlled "turtle," and its graphical equivalent; children could now think of LOGO program instructions as messages sent to the turtle. In subsequent experiments, Papert and Radia Perlman (218) found that children as young as four years old could learn to control mechanical turtles; Cynthia Solomon (218) found that first-graders could learn to program by using schematic turtles on a computer screen as their "objects to think with."

4.4. Smalltalk

Alan C. Kay (1993) has written a comprehensive account of the early history of Smalltalk, the first consciously *object-oriented* programming language, and the source of the architecture of abstractions that underlies every OOP to date; an architecture identical (see above), except for labels, to that of Rand's Objectivist epistemology. The following summary extracts from Kay (1993) those influences that bear on independent but convergent development of OOP and Objectivist epistemology; for the broader context of these events the reader is referred to Kay's account.

As a graduate student at the University of Utah, which he entered in 1966, Kay studied an early release of SIMULA 1. In 1968, he

heard of Seymour Papert's work from Marvin Minsky, at a meeting on Education in Park City, Utah. Later that year, Kay visited Seymour Papert's laboratory and met Cynthia Solomon. Solomon's finding that first-graders could learn to program, if programming was presented in the form of messages to "an object to think with," was the second big insight for Kay: "As with SIMULA leading to OOP, this encounter finally hit me with what the destiny of personal computing *really* was going to be" (73).

After completing his dissertation in 1969, Kay moved to the Stanford Artificial Intelligence Laboratory, where he "spent more time thinking about notebook Kiddy Computers than AI" (74). When Xerox set up its Palo Alto Research Center in 1970, Kay began to consult for it; in 1971, he set up the Learning Research Group and wrote the first draft of Smalltalk. In 1972, Dan Ingalls, a member of Kay's group, wrote a working Smalltalk language processor.

Smalltalk was derived from SIMULA and LOGO by taking only their contributions to the object idea, and rigorously discarding everything else. In SIMULA, classes were an extension of procedural ALGOL. In LOGO, there was only one object, the Turtle, and sending messages to this object was an instructional metaphor used to teach an otherwise applicative language. In Smalltalk, everything is an object. All objects communicate exclusively by sending and receiving messages. Every object is an instance of a class, which encapsulates the shared behaviors of its instances. While classes in Smalltalk 72 still looked like (and could be used as though they were) functions in a procedural language, by 1976 all the features of a modern object-oriented language were in place. Smalltalk 76 included inheritance (discussed in a separate section below), and free-standing functions were eliminated (i.e., every event takes place when an object carries out one of its *methods*—that is, every event is an action of an entity that acts).

4.5. Philosophical Foundations of Smalltalk

Kay includes an appendix of acknowledgments, written in 1973, shortly after Smalltalk 72 started working. It begins: "Much of the philosophy on which our work is based was inspired by the ideas of

Seymour Papert and his group at MIT" (93). Papert was responsible for getting computer scientists to think of knowledge representation as an application of epistemology. Papert's philosophy—or rather epistemology—is the "genetic epistemology" of Jean Piaget (1950), with whom Papert had worked between 1958 and 1963.

Piaget was primarily a biological and psychological theorist. He regarded epistemology as contiguous with cognitive psychology and developmental biology; his innovation was to treat epistemology as a discipline of applied science—in Objectivist terms, an applied philosophical science based on the primacy of existence, rather than the primacy of consciousness that underlay prior "humanistic scholarship" in the philosophical disciplines.⁴

Before Piaget, epistemologists based their prescriptions for knowledge representation on abstractions that had no specific grounding in how human minds actually represent knowledge. Piaget was the first to insist that epistemological prescriptions be based on what was actually known about the human mind, its biological function, its genetic evolution and its development, in reality.⁵

Kay's rethinking of programming along explicitly object-oriented lines started with his application of scientific findings about how children actually develop their representations of knowledge, brought by Seymour Papert from Jean Piaget's psychological laboratory to the MIT Artificial Intelligence Lab.

Kay (1993) mentions two other philosophical ideas that guided the development of Smalltalk. One is Leibniz's *monadic metaphysics*: "To get everything out of nothing, you only need to find one principle" (74). Kay divides programming languages into those that "seem to be an agglutination of features—or a crystallization of style . . . LISP, APL, and Smalltalk are the second kind. It is probably not an accident that the agglutinative languages all seem to have been instigated by committees, and the crystallization languages by a single person" (69–70). While Kay is less extreme than his Leibnizian rhetoric might suggest—he lists a half-dozen principles, not just one, as the foundation of Smalltalk—it was this Leibnizian minimalism that drove the requirement for the simplest structure of abstractions that was both consistent with how human children learn and

sufficient to represent all knowledge.

The third is Plato's principle (*Phaedrus* 12:265e) "of dividing things into classes where the natural joints are, and not trying to break any part after the manner of a bad carver." Kay (1993) quotes a modern abbreviation of this idea, "dividing nature at its joints" (71). Dan Ingalls, the implementer of Smalltalk, mentions the same principle in his 1981 interview in *BYTE*. This means that the designer of a programming language should not just invent a structure of abstractions from aesthetics and whimsy; rather, the abstractions must mirror the already existing structure of the problem the designer is solving, even if that problem is as broad as the representation of knowledge itself.⁶

4.6. The Question of Rand's Influence

Rand first published her *Introduction to Objectivist Epistemology* in eight consecutive monthly issues of *The Objectivist*, from July 1966 through February 1967. She re-published it as a monograph from *The Objectivist* later in 1967. Between 1969 and 1971, she hosted a seminar on her work in epistemology with some dozen philosophers, mathematicians and physicists. The abstractions Rand proposed to represent knowledge in her work, and the relations among those abstractions stand, as noted above, in an exact one-to-one relationship to those of OOP as developed by Kay between 1971 and 1976. Their architectures of abstraction for the representation of knowledge are identical; the only differences, apart from ontology, are in labels.

In view of the temporal sequence, it seemed plausible to ask whether Rand's work had any actual influence on Kay and his colleagues at Xerox PARC. In October 2001, I sent electronic mail messages to Kay, and to all the members of his original Smalltalk team whose e-mail addresses were available on the Internet, asking this question. The replies I received were unanimous: Ayn Rand's epistemology was not discussed by or known to any of them. Kay kindly sent me an extended reply, reproduced in Appendix A, in which he mentions Carnap as the source of the neo-Aristotelian conception of causality (each action is caused and determined by the nature of the entities that act) underlying the object-oriented

paradigm. Other parallels between the philosophical influences noted in Kay 1993, and similar ideas in the work of Rand, are discussed below.

4.7. Parallels to Objectivist Epistemology

As noted by Kay (1993), the inclusion of facts about how children learn into the epistemological foundations of Smalltalk came from Seymour Papert (who in turn credits most of it to Jean Piaget). In this respect, Rand developed her epistemology not very differently from Piaget. The first foundation of Rand's epistemology was primacy of reality. The philosopher's job was not merely to invent a normative scheme for how knowledge ought to come about, but to describe how knowledge is actually built, by actual human minds, in reality. But our knowledge of how this happens comes from the findings of developmental psychology. Campbell (1999, 126–27) writes:

Jean Piaget also considered the means by which children acquire new knowledge to be crucial to an account of concept formation (or, for that matter, of any other problem area in epistemology). But for that very reason he rejected all attempts to assert that philosophy has priority over psychology, and all attempts to wall off philosophical inquiry from the theories and findings of psychology. Rand's understanding of the critical role of cognitive capacity and limitations and her interest in cognitive development should have led her to regard epistemology as a cognitive science (as Ó Nualláin does), or even as a developmental science (à la Piaget, or Feldman). A truly systematic and integrative conception of knowledge of the sort that Rand aimed at would overcome another dichotomy, the dichotomy between epistemology and psychology. Instead, her attempts to isolate philosophy from the sciences have obstructed the assimilation of cognitive psychology by most Objectivists. Even David Kelley, who once taught in a cognitive science program, and has made his own extensive use of ideas from

cognitive psychology, still accepts the formulations of Quine, a philosopher who never gave up his allegiance to behaviorism, as circumscribing the potential of “naturalized epistemology,” and continues to insist on one-way traffic between philosophy and psychology. I am convinced that Rand’s antipsychological metatheory (a set of declarations more than a little at odds with her actual practice) has significantly inhibited the further growth of the Objectivist epistemology, within the areas sketched by Rand as well as beyond them.

In actual practice, Rand’s structure of abstractions for the representation of knowledge was, like Piaget’s, Papert’s, and Kay’s, driven largely by the facts of developmental psychology. Because Piaget, unlike Rand and Kay, did not consider the architecture of knowledge representation as the central problem of epistemology, his epistemological system is quite different from theirs. In particular, Piaget’s basic unit of human knowledge is the scheme, not the concept. Because the scheme does not map in a simple or elegant way into an architecture of representation, Kay went on to add other principles to his foundations.

The second principle mentioned by Kay—“To get everything out of nothing, you only need to find one principle” (1993, 74)—corresponds to something Rand advocated as a basic principle of design. One of the first things that Howard Roark, speaking for Rand, says in *The Fountainhead* is: “Nothing can be reasonable or beautiful unless it’s made by one central idea, and the idea sets every detail” (Rand 1943, 18). It is not surprising to find Rand using this principle, not only in fiction about architecture, but also in her design for an architecture of knowledge representation. Rand and Kay also share a visceral distaste for collectivized design by committees (Kay calls it “agglutinative”).

Kay’s third principle (1993, 71; from Plato, *Phaedrus* 12:265e), that his architecture of knowledge representation should let programmers “divide things into classes where the natural joints are” is even more basic for Rand, who developed her epistemology from the premise that correct mental classification, while not intrinsic in reality (contra

realists) needs to be coherent with reality (contra subjectivists or nominalists). Binswanger (1989, lecture 3, tape 1, side B) uses the same metaphor from Plato, to explain Rand’s concept of “objective abstraction”:

Objectivism holds, in this regard, that nature is like a roast chicken. It’s true that the roast chicken is not already cut up, and it doesn’t already have dotted lines on it saying “cut here.” So, god doesn’t tell us where to cut the chicken. On the other hand, it doesn’t mean that we can take the knife and just cut anywhere. If you want to avoid making a mess, you have to find the joints, then cut it at the joints. So the joints are there, they’re not labeled; they’re not marked “cut here,” and they’re not physically separated. But there is a right place to divide phenomena and a wrong place.

These last two principles also correspond to Rand’s ([1966–67] 1990, 72) twin “razors” of epistemology: “*concepts are not to be multiplied beyond necessity . . . nor are they to be integrated in disregard of necessity.*”⁷

In sum, the two structures of abstractions match not because of mutual influence, but because they were driven by adherence to the same constraints: (1) Aristotelian causality; (2) developmental psychology; (3) minimalism; and (4) knowledge as an objective relation between representation (for Rand, representation in consciousness; for Kay, representation in programs) and reality. Independent discoveries of the same structure are, of course, frequent in the natural sciences; the possibility of several scientists independently discovering the same set of facts follows naturally from the nature of their task: to discover, objectively, the facts of the same reality. The two structures of abstractions match because they were discovered, as is done in the sciences, not invented or created out of their authors’ consciousness, as might have been the case in a “humanistic” discipline (as philosophy was traditionally held out to be). This bears out a point noted by Merrill (1997; see note 4): Historically, the specialized sciences separated from philosophy when their practitioners switched from the “humanistic” focus of philoso-

phy (what Rand calls “primacy of consciousness”) to the discovery of objective fact (“primacy of reality”). Rand’s principle of “primacy of reality” means that in objective philosophical science, as in any science, the topic is reality, not ideas. If the philosophical sciences become driven by the primacy of reality,⁸ then independent discovery of the same aspect of reality, by two or more scientists, sometimes will happen in objective philosophy as it already happens in every positive science.

One consequence of Rand’s principle of the primacy of existence is that the proper division of philosophical knowledge must be based on the objectively identified structure of fundamental facts of reality, rather than on agreement with any philosophical school or with a specific philosopher’s system of thought. In deference to Rand’s request that the term “Objectivism” be reserved for her own philosophical system, I use the terms “objective logic” and “objective epistemology” (rather than “Objectivist logic,” etc.) for the content of philosophical sciences based on the primacy of existence (and on additional principles grounded in the primacy of existence) as those sciences are extended beyond the scope of Rand’s own work.

5. Logic: Scope and Inheritance

Objective epistemology and OOP attract very different numbers of active researchers. While there are at most a few dozen philosophers and logicians exploring the implications of Rand’s epistemology, computer scientists working on OOP languages and software development methodologies number in the tens of thousands. New logical techniques for object-oriented languages and methodologies are being constantly discovered, developed and tested, and have been since 1972. In view of the correspondence between Objectivist epistemology and the foundational abstractions of OOP, it may be productive to examine the potential utility, for objective epistemology, of logical techniques developed in the latter.

5.1. Epistemology and Logic

“Centuries ago, the man who was—no matter what his errors—

the greatest of your philosophers, has stated the formula defining the concept of existence and the rule of all knowledge: A is A . . . I am here to complete it: Existence is Identity, Consciousness is Identification” (Rand 1957, 934). With those words, Rand staked her claim as an innovator—not in metaphysics, where her identification of existence with identity had been anticipated by others, most closely (Kelley 1986, 29 n. 31) by Sartre (1943)—but in epistemology. If logic is “the art of (non-contradictory) identification” (Rand [1966–67] 1990, 46) and epistemology is the study of knowledge—that is, the study of identification—then logic is the art of putting epistemology into practice. To erase doubt about her departure from the epistemology of Aristotle and the Aristotelians, Rand begins her *Introduction to Objectivist Epistemology* (on the very first page of the original Foreword in *The Objectivist*) by listing “the Moderate Realists, whose ancestor (unfortunately) is Aristotle” (2) among those offering defective “solutions” (3)—“solutions” in Rand’s own quotation marks—to what is “known as ‘the problem of universals’” (1). While dissenting from Aristotelian epistemology, Rand certainly admired Aristotelian contributions to logic; the three parts of *Atlas Shrugged* (Rand 1957) are subtitled “Non-Contradiction,” “Either-Or,” and “A is A.” Yet her own contributions to epistemology led Rand (e.g., [1966–67] 1990, 79) to dismiss some traditional Aristotelian arguments as “intrinsicist.”

From the perspective of Objectivist epistemology, Aristotelian logic is right but incomplete. Objectivist epistemology requires a rich logic that includes not only syllogisms, but also error recovery procedures, starting with “when you come to a contradiction, check your premises” (Rand 1957, 188); scope tracking (below, section 5.2 of this article); and simplification tools such as inheritance (below, 5.3) that are necessitated by the two “Razors,” Ockham’s and Rand’s ([1966–67] 1990, 72).

In science, broadly understood as the discipline of acquiring knowledge in accordance with the primacy of existence (Merrill 1991, 155), theory often advances by understanding and integrating methods discovered through practice. Where Rand’s innovations in epistemology have analogues in OOP, logical techniques from OOP

can illuminate issues in objective epistemology. In particular, *scope tracking* facilitates the identification of *scope violations* (errors that Rand and Peikoff ascribe to “intrinsicism,” see section 5.2 below), while *inheritance* (Hartford 1995) is useful in detecting violations of Rand’s and Ockham’s “Razors.”

5.2. Scope

The reason for Rand’s scare quotes around what is known as the “problem of universals” is that she did not regard concepts as “universals” (Rand [1966–67] 1990, 53):

The extreme realist (Platonist) and the moderate realist (Aristotelian) schools of thought regard the referents of concepts as intrinsic, i.e., as “universals” inherent in things . . . as special existents . . .

While most Aristotelians gave our perception of the facts of reality a role in the recognition of “universals,” they held that once recognized, “universals” could be used universally, without regard to the scope of the facts of reality that led men to recognize them. Therefore, Aristotle’s logic had no use for scope-tracking. The Aristotelian view of concepts as “universals” led to a logic that was mainly a matter of plugging terms into syllogisms and grinding out conclusions. It is thanks to this legacy of Aristotle that formal logic is normally understood to be maximally decontextualized. Since concepts were assumed to be valid universally, the validity of carrying a valid conclusion from one syllogism to the next was not questioned, and not considered questionable.

In accordance with the principle of the primacy of reality over consciousness, and with her identification of concepts as units of general knowledge *in consciousness*, Rand explicitly rejects the notion of concepts as “universals”—as special existents (“essences”) intrinsic in reality and universal in scope. In contrast to Aristotle, Rand regards concepts as mental references to classes either of perceptible units or of simpler concepts; in either case, the boundary of the class subsumed under the concept is itself necessitated by relevant facts

about the concept’s constituent units, and by those facts’ relationships to other concepts and facts. Thus, the super-category of “concept” is: a mental reference to (an open, possibly infinite set of) facts of reality. But many (even open, possibly infinite sets of) facts of reality have limited, non-universal scope. Outside the bounds of the facts of reality to which it refers, a concept refers to—nothing. Rand’s epistemology implies that a concept is valid only within the scope of the facts to which it refers. This unavoidably bears on the practice of logic. Logical operations—such as carrying a term from one syllogism to the next—are only valid insofar as they operate on concepts *within the bounds of the facts of reality on which those concepts are based*. Outside those bounds, concept labels are mere “floating abstractions” (Rand [1966–67] 1990, 43).

The requirement of using each concept *only within the scope of the facts of reality on which that concept is based* is the basis of Rand’s (1964a, 13–14) explicit delimitation of the scope of ethics. The first step is the dependence of the concept of “value” on the concept of “goal,” which in turn requires a process of choosing among alternatives. Therefore, “where no alternative exists, no goals and no values are possible” (13). Ethics, which classifies the goals and values of an agent as good or evil for that agent, applies only to agents whose continued existence depends on what goals and values they choose. This further restricts the scope of ethics to living beings: “It is only to a living entity that things can be good or evil” (16). And the scope of the concept of “living entity” in the context of ethics is not simply a biological organism, but *an organism whose life is conditional on its own choices among alternative goals*. “To make this point fully clear, try to imagine an immortal, indestructible robot, an entity which moves and acts, but which cannot be . . . injured or destroyed” (16). The scope is not simply life in the biological sense: Rand uses a biological robot (in technical terminology, an android); it could not be “immortal” if it were not a living thing to begin with. But because the “robot” is not subject to death or injury, it is outside the scope of the conditionality of life on one’s choice of right action. Therefore, it is outside the scope of ethics.

Unfortunately, Rand did not explicitly identify the need to stay

within the scope of each abstraction, as requiring an extension of logic beyond Aristotle's syllogism machine. This led some writers (e.g., Den Uyl and Rasmussen 1984, 18) to discount Rand's explicit opposition to Aristotelian intrinsicism, claiming that Rand's epistemology is simply Aristotelian, and opposed only to "Platonism among proponents of Aristotelianism."⁹

When Rand ([1966–67] 1990, 79) encountered the practice of using a concept outside its scope, she identified this as a consequence of "intrinsicism," that is, of adherence to the notion of concepts as "universals" intrinsic in reality. Yet, in most cases, the "intrinsicist" is only using, albeit somewhat mechanically, the (Aristotelian) logic she or he learned in school. For example, Johnson and Rasmussen (2000) make a straightforward—and, except for scope violation, syllogistically impeccable—argument for ascribing rights to a human fetus. But Rand already dealt, in the "immortal robot" example, with exactly this specific scope violation. The fetus' life is supported vegetatively by the womb, and to the extent that it depends on anyone's choice of values and goals, it depends only on the choices of the pregnant woman. The fetus' life, like that of Rand's "immortal robot," does not depend in any way on its own choice of actions. The fetus, like the "immortal robot," is outside *the scope of the conditionality of one's life on one's choice of right action*, and therefore outside the scope of ethics.

Some OOP languages were created by gradual adaptation of procedural languages. One of these, C++, was an object-oriented adaptation of the widely used procedural language C. Among the programming constructs inherited by C++ from C were "pointers" (reified references). Many defects of software written in C and C++ are traceable to "rogue pointers"—references used outside the scope of the facts to which they refer. The "rogue pointer" is analogous to what Rand calls a "floating abstraction," and the defects resulting from the use of "rogue pointers" are parallel to errors that Rand and Peikoff ascribe to "intrinsicism" (see, for example, Peikoff's 1991 discussion of the errors of scholastic Aristotelians in dealing with the virtue of justice (284) or "fair price" (400)). Modern OOP languages, including the otherwise C-based language JAVA, do without pointers,

largely to eliminate the rogue-pointer problem. In these languages, the data structure that describes an instance of an object is explicitly created when the program enters the bounds of the object's existence, and explicitly destroyed when the program leaves. Except for "constructors" that create representations of instances, attempts to invoke a method that belongs to a given object class *outside the bounds of any object of that class* are automatically identified as "scope violation errors."¹⁰

The logic of "scope tracking" may be outlined as follows:

(1) Explicitly identify each general fact of limited scope (that is, a fact that exists only within the bounds of some less-than-universal context) when this fact is first used. In implementations of OOP languages, this is done by marking an object class as active (or "initialized") when the first data structure representing an instance of the object is created—that is, when a program enters the boundary, or scope, of the facts that give rise to the object. In the corresponding logic of claims about reality, it is necessary to note the (possibly bounded) context of every asserted, non-axiomatic generalization.¹¹

(2) Explicitly verify that every use of a concept is within the scope, as identified in step (1), of all the facts of reality to which that concept refers. In OOP languages, this means that whenever a method (other than the new instance constructor, when used to initialize an instance of a previously uninstantiated class) is invoked, it is invoked either with respect to a specific object instance, or with respect to a class with one or more instances that have been created and have not been destroyed. In syllogisms, this means that one must identify at each step the context, if any, of every term in the syllogism—and reject any would-be syllogism which lacks a context that lies simultaneously within the contexts of all its terms. And then explicitly delimit, as in step (1), the scope of the result.

As an example of how these steps are applied, consider Rand's (1964c, 92) argument in the "logical transition from the principles guiding an individual's actions to the principles guiding his relationship with others." This is the argument relating rights in the individual context, that is "conditions of existence required by man's nature for his proper survival," (Rand 1957, 976) to rights as "moral

principles defining and sanctioning a man's freedom of action in a social context" (Rand 1964c, 93). The starting point is (A) the fact that a rational man will only act in ways that safeguard the conditions of existence required by his nature for his proper survival. Therefore, (B) he will benefit from cooperation and trade with others only if those with whom he deals respect the conditions of his proper survival. He knows that (C) those with whom he can deal for mutual benefit are rational men like himself, and therefore (D) they will deal with him on condition that he respect *their* conditions of proper survival. Therefore, (E) a rational man, in order to benefit from cooperation and trade with others, will respect the conditions of proper survival—the rights—of other men.¹²

How is scope tracking applied to this argument? Proposition (A) is bounded by the requirement that there *be* conditions of existence in which one's proper survival is possible. Proposition (B) is bounded by the requirement that the context permit one to benefit from cooperation and trade. These bounds exclude, from the scope of the principle of respecting rights, the so-called "lifeboat situations" often invoked in discussions of other ethical systems. These bounds on the scope of rights are a logical necessity, and not an objection: as Rand (1964b, 49) wrote, "men do not live in lifeboats." Proposition (C) adds another bound: that the other, the person one is dealing with, is rational. Thus, for example, it is not a violation of the principle of respecting rights to prevent a depressed friend from killing herself: it is outside the scope of principle (E) because of this bound (and is a selfish action, because one stands to benefit from dealing with the saved friend when she is rational again).

In OOP languages, scope-tracking is automatic. It relieves the programmer of the need to watch for rogue pointers, and makes her task less laborious. In conscious syllogistic reasoning, scope-tracking imposes on the thinker a greater need for persistent mental focus than would be the case without it. But it is necessary to keep the result of a chain of syllogisms true of reality—that is, to keep it true.

5.3. Inheritance

The following discussion of the applicability of inheritance to

logic that implements Rand's epistemology is based on Robert Hartford's presentation at the 1995 Summer Seminar of the Institute for Objectivist Studies. In OOP, "inheritance" is the operation of what Rand, in her *Introduction to Objectivist Epistemology*, called "Conceptual Common Denominators."

A *common denominator*, in arithmetic, is a technique for adding or subtracting fractions with different denominators as though they had the same denominator. By analogy, Rand's "Conceptual Common Denominator" (CCD) is the set of characteristics, which is the set of actions or measurements (in OOP, member methods or variables), that permits mental integration and similar mental operations on entities from different concepts (object instances from different classes) as though they were instances of the same concept or class. Some OOP languages have an abstraction (called "interface" in JAVA) that corresponds exactly to Rand's CCD; most implement CCDs as sets of members (characteristics) "inherited" from a wider class by more specific classes.

Inheritance is the application of knowledge and procedures from a wider class or concept to a narrower one. For example, a member of the class/concept "chair" inherits the measurements and actions of "furniture," so that unless we have found out otherwise, we can apply what we know about furniture when asked about a chair. Of course, a chair is not in all respects like other furniture: it has a seat, so it may tip over differently than furniture without a seat, etc. But inheritance provides the cognitive economy of being able to apply to chairs *any measurement or action of furniture* that is not known to be specifically different for chairs. The main benefit of inheritance techniques is cognitive economy: that which is already known about instances of one concept by virtue of its CCD with a second, may be re-applied to the second concept rather than developed from scratch.

A pure CCD, or in programming languages an *interface*, permits the application of the same cognitive economy between concepts/classes that share a subset of measurements and actions without belonging to the same broader concept or class. For example, appliances and furniture have some of the same measurements and actions (supporting surface, number of legs or supports, height,

width, depth, tipping over when hit if higher than wide, etc.). They may, however, differ systematically in some of those measurements and actions. In the case of appliances, for example, “supporting surface” is a sortal variable; it may be the floor or a tabletop, bench top, etc. In the case of furniture, the supporting surface is a constant characteristic—it is always the floor. The CCD permits one to figure out *just once* how to avoid tipping over instances of either concept, move them around, etc.

Rand’s use of CCDs, like OOP’s use of inheritance, is pervasive. In *Introduction to Objectivist Epistemology*, Rand uses CCDs in her explanations of concept formation (ch. 2), abstraction from abstractions (ch. 3), consciousness (ch. 4), definitions (ch. 5), and axiomatic concepts (ch. 6). In her *Epistemology Workshops*,¹³ Rand explained CCDs at length and used them in discussions of abstraction as measurement-omission (seminar A), the role of words (seminar D), meta-abstraction (seminar F), thought and emotion (seminar G) and the make-up of entities (seminar J). The use of CCDs is one of Rand’s two most salient innovations in epistemology, as basic as measurement omission and much more generally applicable. Unfortunately, writings by others about Rand’s epistemology seldom reflect the importance and ubiquity of CCDs in Rand’s thought; even Peikoff (1991) only mentions CCDs twice, both times in relation to concept-formation in his chapter 3.

The techniques that have been developed by object-oriented programmers under the heading of inheritance are readily applicable to logic based on Objectivist epistemology; Rand’s own extensive use of CCDs coincides with several of those techniques. The foundation for inheritance techniques is explicit identification of CCDs. The CCD of a group of concepts consists of their common measurements (which, of course, may have different values for different instances of either concept) and corresponding (but not necessarily identical) actions. Depending on the number and importance of the corresponding actions and measurements in the CCD, the underlying concepts may be either integrated into a wider concept (if it makes sense to consider the characteristics included in the CCD as epistemologically essential for this wider concept) or a stand-alone CCD

(analogous to “interface” in JAVA). While not all OOP languages implement multiple inheritance, multiple inheritance is clearly a fact of reality and therefore a principle of objective logic—for example, “girl” *in fact* inherits from both “child” and “woman.”

After one or more wider concepts or stand-alone CCDs have been specified, it is no longer efficient to duplicate their characteristics in mental operations on the concepts that “inherit” them. Instead, one need only refer, in mental operations on the more specific concept, to the wider concepts or stand-alone CCDs it “inherits from,” specifying only those of its inherited actions that differ from the norm inherited from wider concepts and CCDs, and measurements and actions beyond those inherited. In the *Epistemology Workshops* cited above, Rand does this repeatedly. Her seminar on “thought vs. emotion” (Rand [1966–67] 1990, 223–25) is an extended illustration of this technique: it starts with the question, “What is the CCD uniting thought and emotion?” and then uses this CCD to illuminate the relation between the two.

6. Summary

The parallels between Objectivist epistemology and OOP illuminate the boundary that Rand bridged¹⁴ from philosophical systems created by and in human consciousness, to objective philosophical sciences based on the primacy of reality. In objective sciences, which now include epistemology and logic, knowledge is discovered rather than created. The same discovery can be made independently by different minds, because it is a fact of the same reality that is being discovered. There is no reason for any user of logic to refrain from using facts discovered by others—and in the case of logic derived from analogous principles in Object-Oriented programming and Objectivist-Objective epistemology, there is much to be gained by using techniques discovered by both.

7. Appendix: Alan Kay’s Response

Date: Fri, 2 Nov 2001 00:02:51 -0800
 To: Adam Victor Reed <areed2@calstatela.edu>
 From: Alan Kay <Alan.Kay@squeakland.org>

Subject: Re: Query - History of OOP

Adam -

LOGO was based on two previous languages: JOSS and LISP. JOSS was the first great language designed for end-users, and LISP was the first truly great computer language, period. These languages, as with almost all computer languages, had some root in mathematical languages, most of which are "extensional" in nature—that is, there is an underlying notion (or at least a whiff) of operations on sets. Piaget's metaphors for the thinking of children were drawn from his interest in mathematics, and these were extensional as well: in terms of operations.

In the "Early History of Smalltalk" that I wrote (in *History of Programming Languages II*, ed. Bergin, Addison-Wesley, ca 1996 [the actual reference is Kay 1993—AR]), I described how my conception of objects arose from:

- seeing "almost object" ideas in Sketchpad, SIMULA, B5000, etc.
- my background in Biology, particularly cell biology
- my background in Mathematics, particularly pure math, Algebras, etc.
- that I was a grad student in an ARPA project, and we were thinking about doing the ARPAnet

This was an "intensional" view, in which the "operations" are actually "intrinsic" "properties" or "behaviors" of entities. The terms "extensional" and "intensional" I got from Rudolf Carnap, a logician, who had created a system of logic that was intensional. I think his book was *Meaning and Necessity*.

In my case, I was trying to understand SIMULA I, had just seen Sketchpad, and it occurred to me:

- that SIMULA was dealing with the same kinds of entities that Sketchpad was, but procedurally (not as pretty, but more pragmatically exploitive).
- that cell biology already did something nicer, and in a very comprehensive and big way. That this could be a great road to handle exploding complexity and introduce extreme scaling.

- that this would be a great architecture for the ARPAnet, in which there were to be eventually millions of machines able to cross communicate.
- that you might be able to get millions of "software machines" with the same robust properties if enough SW engineering were done.
- that you could generalize procedural attachment a little and you would get something that was like an algebra (where you have only a few names for lots of similar operations).
- that this would be intensional (per Carnap) but not a logic so much as a computational scheme.
- that you could do pattern matching on incoming messages, and this would also give you a way to extend both the syntax and the semantics of the language.
- etc.

In other words, Nov. 11th 1966 was a very "happy" day for me.

Ayn Rand's philosophy had no bearing on these ideas. However, some of the notions intertwined in relativity theory did, particularly the ones regarding multiple reference frames and the difficulties of synchronization. This POV definitely leads to a conception of the universe in which control is distributed amongst the entities. We also see this in bio, a few levels of org higher, and then in ecologies, a few levels of org still higher. The insights about how the metaphors of relativity could help guide the design of object-based systems did not occur in the first flashes of the first day, but crept in over the next several months of thinking about the ramifications of the ideas.

In sum, one doesn't have to go further than science and math in the last 150 years or so to get many of the primary POVs that helped lead to the invention of objects. BTW, a deeper philosophical view than Rand's can be found in a variety of books by A. N. Whitehead (including *Process and Reality*). These books were based on a deeper understanding of modern physics than Rand apparently had. I don't agree with all of Whitehead's notions, but I'm sure they contributed one way or another to the thinking I was doing in the late sixties.

Notes

1. Galileo tied one end of a string to a nail on a board, and attached weights to the string's other end, which hung from the board's edge. He found that to change the pitch by an octave (factor of 2 in the frequency of vibration) required 4 times the original weight, not 2 times as claimed by Aristotle (Drake 1970; 1975; 1978). Long (2000) argues that Aristotle, unlike Rand, regarded "what is evident to everyone or to most people"—the collective wisdom of mankind—as *endoxa*, premises that required no additional validation from sense data. If Long is right, it could be said that Aristotle lost favor with post-Renaissance scientists because he was not an Objectivist.

2. "Every network manager has a security policy. Those who say they don't have one just don't know what theirs is" (Steven Bellovin, AT&T Network Security Lectures, based on Cheswick and Bellovin 1994).

3. For more on the relation of Rand's philosophical epistemology to Gibson's psychological epistemology, see Kelley 1986, 66–75.

4. Merrill (1997) identifies Rand's distinction between "Primacy of Existence" and "Primacy of Consciousness" with the boundary between science and "humanistic scholarship" in the sense of Snow (1993). According to Merrill, science does *not* differ from humanistic scholarship primarily because it uses different methods of thinking. Merrill argues that those who distinguish science by its methodology—that is, by what goes on in the scientist's consciousness—are looking at science from the external perspective of the primacy-of-consciousness culture, and miss the essential difference: that, in science, unlike pre-Objectivist philosophy, the topic is reality, not ideas. Scientists, depending on the aspects of reality being dealt with, use any appropriate methodology such as experimentation, observation, induction, deduction, etc. What defines the real scientific method is not any one way of thinking, but rather the commitment to "go by the evidence" of reality in judging what is true, without regard for the effect of doing so on the state of anyone's feelings, beliefs, or claims to wisdom.

Philosophy, self-defined from Thales on as "love of wisdom"—that is, the pursuit of a wise consciousness—has been, historically, the paradigmatic discipline of "humanistic scholarship." At extremes, its commitment to primacy of consciousness was expressed in positions implying that there was no reality external to consciousness; or that external reality, if it existed at all, was not knowable; or that we can know our minds without knowing the external world; or that knowledge of some aspects of reality should not be sought, because obtaining it would compromise the state of consciousness esteemed as wisdom. But primacy of consciousness appears implicit also in less extreme positions, such as Aristotle's occasional acceptance of a belief that was intuitively plausible, and did not contradict anything he already knew, without doing what to Galileo, and to any modern scientist, would be a simple and obligatory test against reality. Merrill notes that what Rand called the "special sciences," from physics to psychology, gradually cut their ties to the primacy-of-consciousness tradition as they became part of real, primacy-of-existence objective science. Although Rand herself did not claim this explicitly, Campbell (1999) shows that Rand, like Piaget, tried to incorporate available scientific knowledge—particularly from developmental psychology—into the structure and substance of her epistemology. Piaget did this because he had worked for decades collecting data on human development; Rand did this because, at the time (see note 14 below), she was still committed to the integration of philosophy with knowledge available from other primacy-of-reality disciplines.

5. Besides Piaget (1950), Papert (1980, 217) recommends Boden (1979) and a "short list" of nine other books.

6. Of course, if the designer is used to formulating the problem in terms of procedural languages or other non-OOP approaches, there is a significant developmental transition that the programmer needs to undergo before he or she will think in fully object-oriented terms. See Campbell et al. 1992.

7. It may seem ironic to credit Plato, the originator of the "shadows on the far wall of the cave" metaphor for perception, with influence parallel to Rand's "primacy of existence." Yet, in modern terms, Plato's metaphor may have been intended to motivate reasonable caution in interpreting perceptions—a point vindicated by scientific knowledge of perceptual and cognitive illusions—and not to deny the existence of an actual objective reality shaped by universal "forms" (the latter corresponding, roughly, to "laws of reality"). The idea that perception, not being diaphanous, cannot be trusted to produce reliable identification of reality *even when interpreted and corrected by reason* was introduced later, by Jewish and Christian Platonists, notably Philo of Alexandria. Philo argued that Reason could apprehend Forms without mediation by "shadows" (perception and the senses), an interpretation that promoted the identification of Form with God. In that sense, Plato himself was not necessarily a Platonist.

8. Merrill (1997) notes that once the primacy-of-reality principle is accepted, all that obscures the contiguity of the philosophical sciences with the rest of science is the current formulation of their boundary in terms of methodology—*itself* a symptom of primacy-of-consciousness among would-be "philosophers of science." Hence the need to recognize axiomatics as a valid methodology of positive science. See also Merrill 1995.

9. While Den Uyl and Rasmussen (1984) reject "Porphyrean essentialism," the most intrinsic reading of Aristotle's notion of essence, they still dismiss Rand's insight on the finite scope of concepts.

10. In C++, it was possible to retain and "make use" of a pointer to an object that had been destroyed when the program left its scope. The place in storage that was previously occupied by information about the destroyed object was sometimes allocated to other objects by the time the pointer was misused—and in that case the rogue pointer corrupted information about unrelated objects, causing defects that were extremely difficult to trace. For a more extended discussion of the havoc created by rogue pointers, see Cline and Lomow (1995, FAQs 304, 306 and 308).

11. In some contexts, particularly when the measurements omitted in the induction of a generalization are sortal rather than scalar, parts of the exact location of the boundary may be optional. This is analogous to Rand's ([1966–67] 1990, 72–74) discussion of the "borderline cases" of a concept. Thus, when using a general fact to answer some specific question, it may be enough to place the underdetermined parts of the boundary in any convenient place, subject to the requirement that its placement must be coherent with known facts of reality, and made explicitly clear.

12. While Rand wrote that "Man is not a lone wolf and he is not a social animal. He is a *contractual* animal" ([1972] 1979, 2–3), she was not a *contractarian* in the sense of holding that rights are the result of a "social contract." In Rand's ethics, the decision to respect the rights of others is made by the rational man unilaterally, as a *precondition* for contractual cooperation and trade.

13. The 1990 edition of *Introduction to Objectivist Epistemology* includes excerpts from the epistemology workshops for philosophers and scientists that Rand conducted between 1969 and 1971.

14. After Rand's break with Nathaniel Branden in 1968, she de-emphasized, and in some respects repudiated, their previous integration of philosophy with the inductive sciences. By 1973, Rand was asserting a hierarchical, one-way relation between philosophy and "special sciences" such as developmental and

cognitive psychology (Rand 1973, 278):

Philosophy studies the fundamental nature of existence, of man, and of man's relationship to existence. As against the special sciences, which deal only with particular aspects, philosophy deals with those aspects of the universe which pertain to everything that exists. In the realm of cognition, the special sciences are the trees, but philosophy is the soil which makes the forest possible.

Rand's integration of psychology and biology with philosophy during the period of her collaboration with Branden, and her subsequent de-emphasis and repudiation of that integration, deserve the attention of a separate study.

References

- Binswanger, Harry. 1989. *Consciousness as Identification* (audiotape set). Second edition. Second Renaissance Books/Ayn Rand Bookstore.
- Boden, Margaret. 1979. *Piaget*. London: Harvester Press.
- Booch, Grady. 1994. *Object-Oriented Analysis and Design with Applications*. Second edition. Reading, Massachusetts: Addison-Wesley.
- Campbell, Robert L., Norman R. Brown, & Lia A. DiBello. 1992. The programmer's burden: Developing expertise in computer programming. In *The Psychology of Expertise: Cognitive Research and Empirical AI*, edited by Robert R. Hoffman. New York: Springer, 269-94.
- . 1999. Ayn Rand and the cognitive revolution in psychology. *The Journal of Ayn Rand Studies* 1, no. 1 (Fall): 107-34.
- Carnap, Rudolf. 1989. *Meaning and Necessity: A Study in Semantics and Modal Logic*. Second edition. Chicago: University of Chicago Press.
- Cheswick, William R., and Steven M. Bellovin. 1994. *Firewalls and Internet Security*. Reading, Massachusetts: Addison-Wesley.
- Cline, Marshall P., and Greg A. Lomow. 1995. *C++ FAQs (Frequently Asked Questions)*. Reading, Massachusetts: Addison-Wesley.
- Dahl, O.-J., and K. Nygaard. 1966. SIMULA: An algol based simulation language. *Communications of the ACM* 9, no. 9: 671-78.
- Den Uyl, Douglas J., and Douglas B. Rasmussen. 1984. Ayn Rand's realism. In *The Philosophic Thought of Ayn Rand*, edited by Douglas J. Den Uyl and Douglas B. Rasmussen. Urbana: University of Illinois Press, 3-20.
- Drake, Stillman. 1970. Renaissance music and experimental science. *Journal for the History of Ideas* 31: 483-500.
- . 1975. The role of music in Galileo's experiments. *Scientific American* 232 (January-June): 98-104.
- . 1978. *Galileo at Work: His Scientific Biography*. Chicago: University of Chicago Press; Dover 1995 (paperback).
- Gibson, James J. 1972. A theory of direct visual perception. In *The Psychology of Knowing*, edited by J. R. Royce and W. W. Rozeboom. New York: Gordon and Breach, 215-27.
- . 1977. The theory of affordances. In *Perceiving, Acting, and Knowing*, edited by Robert E. Shaw and John Bransford. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 67-82.
- Harré, Rom. 1970. *The Principles of Scientific Thinking*. London: Macmillan.
- Hartford, Robert. 1995. *Object-Oriented Programming and Objectivist Epistemology*. Participant presentation, Summer Seminar of the Institute for Objectivist Studies, Madison, Wisconsin.
- Ingalls, Daniel H. H. 1981. Design principles behind Smalltalk. *BYTE Magazine* 6, no. 8 (August): 286-98.
- Johnson, Gregory R., and David Rasmussen. 2000. Rand on abortion: A critique. *The Journal of Ayn Rand Studies* 1, no. 2 (Spring): 245-61.
- Jacobson, I., M. Christerson, P. Jonsson, and G. Overgaard. 1992. *Object-Oriented Software Engineering*. Reading, Massachusetts: Addison-Wesley.
- Kay, Alan. 1993. The early history of Smalltalk. *ACM SIGPLAN Notices* 28, no. 3: 69-88.
- Kelley, David. 1986. *The Evidence of the Senses: A Realist Theory of Perception*. Baton Rouge: Louisiana State University Press.
- Long, Roderick. 2000. *Reason and Value: Aristotle versus Rand*. Poughkeepsie, New York: The Objectivist Center.
- Merrill, Ronald E. 1991. *The Ideas of Ayn Rand*. LaSalle, Illinois: Open Court.
- . 1995. Axioms: The eightfold way. *Objectivity* 2, no. 2: 1-15.
- . 1997. *The Radicalism of Objectivism*. Lecture at Reed College (26 April) <http://www.monmouth.com/~adamreed/Ron_Merrill_writes/Articles/RadicalismOfObjectivismhtm.htm>.
- Papert, Seymour. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.
- Peikoff, Leonard. 1991. *Objectivism: The Philosophy of Ayn Rand*. New York: Dutton.
- Piaget, Jean. 1950. *Introduction à l'épistémologie génétique* (3 vols.). Paris: Presses Universitaires de France.
- Plato. 1914. *Phaedrus*. Translated by Harold North Fowler. On the web at <<http://plato.evansville.edu/texts/fowler/phaedrus.htm>>. Stephanus numbering refers to the chapter, page and section in the Henri Estienne (Stephanus) 1578 edition of the original Greek text.
- Popper, Karl R. 1982. *Realism and the Aim of Science*. London: Hutchinson.
- Rand, Ayn. 1943. *The Fountainhead*. New York: Bobbs-Merrill.
- . 1957. *Atlas Shrugged*. New York: Random House.
- . 1964a. The Objectivist ethics. In *The Virtue of Selfishness: A New Concept of Egoism*, by Ayn Rand. New York: Signet (page numbers in paperback), 13-35.
- . 1964b. The ethics of emergencies. In *The Virtue of Selfishness: A New Concept of Egoism*, by Ayn Rand. New York: Signet (page numbers in paperback), 43-49.
- . 1964c. Man's rights. In *The Virtue of Selfishness: A New Concept of Egoism*, by Ayn Rand. New York: Signet (page numbers in paperback), 92-100.
- . [1966-67] 1990. *Introduction to Objectivist Epistemology*. [First published in *The Objectivist* 5, no. 7: 1-11; 5, no. 8: 1-7; 5, no. 9: 1-8; 5, no. 10: 1-7; 5, no. 11: 1-7; 5, no. 12: 1-6; 6, no. 1: 1-11; 6, no. 2: 1-8. Page numbers refer to expanded second edition, edited by H. Binswanger and L. Peikoff. New York: Meridian.
- . [1972] 1979. A nation's unity. *The Ayn Rand Letter* 2, no. 1 (9 October); no. 2 (23 October); no. 3 (6 November): 121-32. In *The Ayn Rand Letter*, Volumes I-IV (1971-1976), published in 1979. Palo Alto: Palo Alto Book Service.
- . [1973-74] 1979. Philosophy: Who needs it? *The Ayn Rand Letter* 3, nos. 7-8 (31 December; 14 January): 277-84. In *The Ayn Rand Letter*, Volumes I-IV (1971-1976), published in 1979. Palo Alto: Palo Alto Book Service.
- Rumbaugh, J., M. Blaha, W. Perelmani, F. Eddy, and W. Lorensen. 1991. *Object Oriented Modeling and Design*. Englewood Cliffs, New Jersey: Prentice Hall.
- Sartre, Jean-Paul. 1943. *L'Être et le Néant: Essai d'ontologie phénoménologique*. Paris: Gallimard.
- Snow, C. P. 1993. *The Two Cultures*. Rede lecture 1959. With introduction by Stefan Collini. New York: Cambridge University Press.

Taylor, Richard. 1967. Causation. In *The Encyclopedia of Philosophy*, editor-in-chief, Paul Edwards. New York: Macmillan, Vol. 2, 56–66.

Whitehead, Alfred North. 1929. *Process and Reality*. New York: Macmillan.