An Aristotelian Understanding of Object-Oriented Programming

Derek Rayside
Electrical & Computer Engineering
University of Waterloo
Waterloo, Canada
drayside@acm.org

Gerard T. Campbell
Department of Philosophy
St Jerome's University
Waterloo, Canada
gcampbel@watarts.uwaterloo.ca

ABSTRACT

The folklore of the object-oriented programming community at times maintains that object-oriented programming has drawn inspiration from philosophy, specifically that of Aristotle. We investigate this relation, first of all, in the hope of attaining a better understanding of object-oriented programming and, secondly, to explain aspects of Aristotelian logic to the computer science research community (since it differs from first order predicate calculus in a number of important ways). In both respects we endeavour to contribute to the theory of objects, albeit in a more philosophical than mathematical fashion.

1. INTRODUCTION

Both the programmer and the logician wish to reason *with order, with ease, and without error* [2] for the sake of achieving *conceptual integrity* [11] in their intellectual expressions. The importance of *rigorous formalism* to conceptual integrity is well understood in programming and in logic. However, other facets of conceptual integrity, such as the use of language, seem to be less well understood. Our enquiry explores some of these other facets of conceptual integrity in programming by comparing object-oriented programming with Aristotelian logic — a logic that is concerned both with formalism and with meaning. By this we hope to make contributions to the theory of object-orientation and thereby facilitate better programming and better programming languages.

Our enquiry follows the order often suggested for those programming in an object-oriented programming language: first to identify the objects and classes, then to examine the relations between them, and finally to reason about them. This is also the order of study in Aristotelian logic, which, according to Thomas Aquinas, consists of three operations: *definition*, *predication*, and *inference* [2].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
OOPSLA '00 Minneapolis, Minnesota USA
Copyright 2000 ACM 0-89791-88-6/97/05 ..$5.00

The complementary roles that meaning and formalism play in conceptual integrity can be illustrated with what we call the 'transference deduction':

The Transference Deduction: What is said (A) of one thing (B) may also be said of another (C) by virtue of the relation between them (BC).

$$\frac{AB \quad BC}{AC}$$

Four examples of the transference deduction that are relevant to object-oriented programming are:

	Every animal is mortal.
1.	Man is an animal.
	∴ Man is mortal.
	Man is mortal.
2.	Socrates is a man.
	∴ Socrates is mortal.
	Homo erectus stands upright.
3.	*Homo sapiens* evolved from *homo erectus.*
	∴ *Homo sapiens* stands upright.
	Key A opens this door.
4.	Key B is a copy of Key A.
	∴ Key B opens this door.

At this point it would appear that all four examples are formally valid according to our model of the transference deduction. Where they differ is in the meanings of the words employed, the relations between the things, and therefore the certainty of the conclusions (AC in the model). With respect to the certainty of the conclusion, we can see that this is determined by the relation of the *middle term* (B) to each of the other terms. For this reason, we may say that the middle term is the 'cause' of the conclusion: it is the reason why we bring the other two terms together (A and C). The middle terms of the example deductions are 'animal', 'man', '*homo erectus*', and 'Key A', respectively. In the last example, the duplicate key will open the door only if it is well made, and so the conclusion seems less certain than in the first example.

According to Brachman, 'understanding what is on either end of the link is also the key to understanding the import of the link' [9]. This is where Aristotelian logic has something to offer to conceptual integrity over and above rigorous formalism.

1.1 Overview of Aristotelian Logic

Aristotle's logical works are commonly referred to as the *Organon* (Greek, for *tool* — and reason is the tool of tools). The *Organon* is divided by Thomas Aquinas according to the three operations[1] of our reason in its activity of coming to know about *things* [2]. The first two of these rational operations are concerned with reason as the *faculty of understanding.* It is the third operation which treats reason as a the *faculty of discoursing* whereby we go from what is understood *already* to the formation of a new truth derived from established premises.

1. Definition The first operation is the subject of Aristotle's treatise on the *Categories* [3], and is referred to as *simplex apprehensio* by Thomas Aquinas [2]. 'Apprehension' refers first to 'reaching out and grasping onto', for example the branch of a tree or a criminal. Here it refers to how reason reaches out and grasps the rational structure in things. This notion of 'apprehension' is also found in the phrase *real world apprehension* as used in the BETA conceptual framework [23, p.2]. So, the first operation begins with grasping the unity in things, which we express in words, and culminates in the art of definition. Once we have grasped the *definitum* (the thing to be defined) in a unified way, we must see what *differentiates* it from other similar things. These distinctions can be either *essential* or *accidental.* If they are essential then they tell us *what kind of thing* the definitum is. If they are accidental then they tell us *something about the definitum other than what it is.* In differentiating the definitum from other similar things we will have to *divide.*

2. Predication The second operation of reason, the subject of Aristotle's treatise *On Interpretation* [3], is referred to as *compositio et divisio* by Thomas Aquinas [2]. Still ordered to understanding, the second operation creates statements in which a predicate is either affirmed or denied of a subject (i.e. *composed* with or *divided* from). The predicate may be said of the whole nature of the subject, or of only of some individuals in which that nature is found. A statement must be either true or false — it is not the subject and the predicate which are true or false on their own, it is in the composition or division of them that truth and falsity arises.

3. Inference All of Aristotle's other logical works [28] address the third operation of reason, which reason is most perfectly named. Notice that the first operation of reason (apprehension of natures) is ordered to the second operation (telling us what belongs or does not belong to those natures), and that the second operation is ordered to the third operation (where reason attains its goal of knowing about things).

This final operation of reason is concerned with the valid ordering of our statements for the sake of a conclusion. *Validity* and *invalidity* are properties of an argument as a whole, whereas *truth* and *falsity* are properties of individual statements. Of course, since the conclusion of an argument is also a statement, it too is either true or false. In the third operation Aristotelian logic focuses on deductive argument in the form of the *syllogism.*

[1]We use the word 'operation' because it is etymologically linked to 'opus', and so implies that the activity is carried out for the sake of a specific work (i.e. an argument).

2. THE FIRST ACT: DEFINITION

In order to define a thing we must grasp it first in a unified way, as a *one* kind of thing. To do this we will need a *principle of unity.* There are two such principles of unity that we need to distinguish here: the *unity of matter* which makes the individual thing to be *this one*, and the *unity of form* which makes it to be the *kind of thing* that it is. We will refer to things that are one according the first principle as *singulars* or *individuals*, such as 'Socrates'. We will refer to things that are one according the second principle as *universals*, such as 'man'. We can immediately see that there is some similarity to the notions of 'object' and 'class' in object-oriented programming.

Aristotle refers the singular as *primary substance*, and the universal as *secondary substance* [*Categories* §5]. This is the historical root of the distinction between 'first order' and 'second order' logic; Aristotelian logic may be considered (loosely) as a 'second order' logic because it is considers only the universal as the proper subject of logic. The reason for this approach is that the singular is the *proper object of the senses* — individuals are differentiated by their matter which is grasped by the senses rather than by reason. This raises an important point: in recent centuries much of the work in logic has been done by mathematicians who work primarily with *immaterial* singulars (such as numbers). Aristotle starts his enquiries with *material* singulars — things in the real world, as does object-oriented programming (at least according to some). Because this is an investigation of logic we must focus on the universal, even though conclusions must still be verified in the real world from which logic originates.

Grammatically Singular and Logically Universal One of our central examples will be the statement *man is an animal.* This example illustrates some important issues that cannot be illustrated with more common computer science examples such as *coloured point is a point.* It is also important to note that we render our example with *grammatically singular* and *logically universal* terms: this is intentional and central to the paper. If we used grammatically plural terms we would obscure the unity of the universal — it is important to realize that the universal represents a oneness of nature, or unity of form which is grasped by reason, rather than simply a collection.

2.1 Signs, Words and Concepts

What we are trying to define must not only be a universal, said of many, but it must designate the same nature in each of the things of which it is said. The logic of the first operation of reason is concerned with *things* as represented in reason by *concepts* that are then expressed in *words* [*On Interpretation* §1]. It is through the ordering of our words (and the concepts represented by the words), that we come to know the world.

As everyone acquainted with a second language realizes, the choice of a word to signify a given nature is *arbitrary.* For the logician, however, whatever the language, is important to maintain the same meaning when we define. But language is an instrument for communicating, so if we wish to communicate, the arbitrary sign must signify also in a *conventional* way — that is, the sign selected has to be 'agreed upon by all'. Even if one of the singular pleasures of using words is that we can play with their different conventional meanings (why we enjoy the *double entendre*), it is important for the programmer and the logician that a word not change its signification *within a given context.* For this reason, Aristotle begins his treatise on the *Categories* by pointing out different ways in which words can be used to signify concepts: *univocally*, *equivocally*, and *analogically.*[2]

We begin by paying attention to the fact that it is not *words* that are univocal, equivocal or analogical, it is *our use* of words that is univocal, equivocal or analogical in meaning. Some people find symbolic logic attractive because variable names such as x have no inherent signification, thus making it easier to use them in a univocal fashion. However, once we give meaningful names to our variables we must respect those names and what they 'stand for'.

How words can be used to signify concepts is relevant to the programmer in three ways: first, in terms of the language he uses to speak about programming; secondly, in terms of the words he uses to name things in his programs; and thirdly, in terms of the impact his naming of things has upon compilers (and other program analysis tools).

Univocal We use a word *univocally* when, within a given context, we *maintain the same signification.* For example, we can use the word 'bat' to signify 'an instrument for striking a ball'. Here the word can be said of a wooden bat or an aluminum bat, of a baseball bat or a cricket bat. We must use words in a univocal fashion if we wish to define well.

Equivocal We use a word *equivocally* when, within a given context, we use it to *signify different and unrelated natures*, or even deliberately play with more than one meaning. For example, we can use the word 'bat' to signify both 'an instrument for striking a ball' and 'a small flying mammal' (a characteristic transferred to the umpire by the fan who yells that the umpire is 'as blind as a bat'). *The metaphorical use of a word is always equivocal* because the nature signified does not belong essentially to the subject, and consequently two different natures are signified simultaneously (this is why it is an equivocation).

For example, if we call a data structure a 'tree' we do so metaphorically: a data structure is a mathematical abstraction; a tree is a living plant. While the data structure and the plant share an accidental resemblance of form, they are not of the same substance (and, for Aristotle, substance is very closely related to essence). We can express this metaphorical usage of 'tree', and any other metaphor, in the form of the transference deduction:

[2]In [*Categories* §1] Aristotle speaks of things being named *equivocally* and *univocally.* The notion of *analogical* signification (which has elements of both) is developed by the mediæval commentators.

A bifurcating plant is called 'tree'.
A bifurcating data structure is like a bifurcating plant.
∴ A bifurcating data structure is called 'tree'.

and, by extension:

A tree has leaves.
A bifurcating data structure is called 'tree'.
∴ A bifurcating data structure has 'leaves'.

The English term 'metaphor' is from the Greek *metapherein*, which means 'to transfer'. It is from this meaning that we have named the transference deduction: both metaphor and the transference deduction involve transference, and both need only an accidental similarity as the basis of the transference.

In the context of software, 'writing', 'engineering', and 'growing' have all been examined by the computer science community as metaphors for the development of software (e.g. [10, 12]). The metaphorical use of the term 'inheritance' has also received a fair amount of discussion in the context of object-oriented programming, and we will treat this particular metaphor later.

The use of metaphor is also quite common in programming itself: we name our classes, variables, etc. most often in a metaphorical fashion. Some programming methodologies, such as XP [8], explicitly encourage this, since metaphor helps us to understand the less familiar in terms of what is more familiar.[3] There are times when the metaphorical usage of words does not seem to cause too much confusion: for example, nobody expects to find sap in a data structure. There is a large body of literature on metaphor itself (it is the subject of several journals), but we are not aware of any work that investigates the use of metaphors within programs and the impact of this on program comprehension — what constitutes good use of metaphor in the development of software?

Finally, a common example of plain equivocation in programming is found in the discussions of structural subtyping: a `Cowboy` and a `Shape` class that both define `draw` methods. The activity signified by `draw` is different in each case, and the danger of this particular equivocation is well understood.

Analogical Our usage is *analogical* when, within a given context, *we use one word to signify concepts that, although they are different in some respects, are* 'essentially' *and not just* 'accidentally' *alike.* Within the context of baseball, one's turn 'at bat' refers to 'the activity of striking the ball', and it is from this activity that 'bat' as 'an instrument for striking the ball' is named (i.e. the verb and the noun are related essentially). Another example of analogical usage is the Greek word *logos*, from which the word English word *logic* is derived. The first meaning of *logos* is *word*; secondly, it signifies the *concept* (or idea) which the word rep-

[3]This is why Plato refers to poets as the 'fathers and authors of wisdom'. However, in comparison to a proper scientific understanding, he refers to them as 'liars twice removed from the truth'. (Which has to be understood in terms of Plato's metaphysical doctrine of *Ideal Forms*: art imitates nature, which in turn imitates *Ideal Form*).

resents; thirdly, it signifies *the intelligible nature of things* represented in the concept; and finally, *Logos* refers to that *Reason* whose works are expressed in *the intelligible natures of things* (this is the *Logos* of John 1:1). The purpose of *logic* is to order our *words* and our *concepts* so that we can come to know the *natures of things.*

Within object-oriented programming, we may consider that 'method overloading' and 'method overriding' are intended as analogical usage: it is generally considered poor programming practice to give the same name to methods that are not related. This intention is part of what is captured in the notion of 'subclass responsibility'. Programming languages such as BETA encourage this more explicitly by use of the `inner` construct instead of arbitrary method overriding.

2.2 Predicable Relations

Good definition is the foundation for clear thinking since it tells us not only the *kind of thing* something is, but also how it is so *in a distinct way.* In defining, whatever we say of the definitum is said either *essentially* or *non-essentially* (accidentally). If it is said essentially, it tells us *what the subject is.* If it is said non-essentially (accidentally), it tells us *something about the subject other than what the subject is.* To state that 'man is an animal' says something essential. To state that 'Socrates has a snub-nose' says something accidental. These ways of characterizing a subject, either essentially or accidentally, are known as *predicable relations.*[4]

2.2.1 Essential Relations: Genus and Species

In modern times the terms *genus* and *species* are most commonly used as biological terms, in part due to the eighteenth century Swedish biologist Carolus Linnaeus. However, these terms have a much older history in logic. Those essential predicable relations in the line of telling us what something is are 'genus' and 'species'.

The predicable relation of *species* is that whereby *something one (universal) is said of many, telling us the nature of the many, which differ only as individuals* — i.e., individuals which differ according to their matter since matter is the principle of individuation. When 'man' is said of 'Peter', 'Paul', and 'Mary', it tells us what kind of thing each of them is. In summary, then, 'species' tells us what kind of thing an individual is.

The predicable relation of *genus* is that whereby *something one is said of many which differ according to kind or species.* In terms of abstraction, *genus* is *more universal* than is *species* (which is also a universal), and it tells us *what kind of thing* a species is. For example, 'animal' (genus) tells us what kind of thing 'man' (species) is — and we should notice that this relation always admits of being expressed as 'man is a kind of (species of) animal'. Similarly, if we say that 'animal is a kind of living substance', we see that once more there is a relation of genus (living substance) to species (animal). So 'animal', for example, can be both a genus (when predicated of 'man') and a species (when 'living substance' is predicated of it).

[4]The teaching on predicable relations originates in the *Isagoge*, written in the third century AD by Porphery the Phoenician. Porphery gives a third essential relation which he calls *specific difference.* For Porphery's criteria of 'specific difference', philosophers have found only two instances: 'rational' said of 'man', and 'sentient' said of 'animal'. A more common and more useful notion of specific difference is 'that which is added to a *genus* to distinguish sufficiently the definable *species* (the *definitum*) within that genus'.

Every definitum must be seen as a species, and the first element in every good logical definition will be its *proximate genus*, telling us in an essential way what kind of thing the definitum is. However, the genus needed to define well is the proximate (closest) genus. 'Animal' tells us immediately what kind of thing 'man' is. 'Living thing' is *too* generic in trying to tell us what 'man' is. As we approach definition, then, we take note of the following:

1. Individuals can never be logically defined, nor are they the subject of logic because their differences are only material rather than a difference of kind.
2. Any kind of thing for which there is no higher genus can never be defined logically, nor can it be a subject of logic, because every definitum requires a genus.
3. The definitum must be a species.
4. Finally, because the genus is more universal than the species, then every definition will require a *specifying difference* to distinguish in an adequate way the definitum from everything else contained within the genus. In a way, the specifying difference reduces the genus to the same universality as the species.

2.2.2 On the Notion of Species

Since idea of the predicable relation of genus to species is so critical both to defining and to object-oriented programming, we will examine it in further detail. Brachman says of its importance:

> This type of IS-A relation that carries structure between structured descriptions is one of the most radical departures from representation schemes based on standard predicate logic. Almost all of the other IS-A relations are easily expressed in standard quantificational languages. [9]

This type of IS-A relation is at the core of Aristotelian logic, and here we wish to investigate how the object-oriented programming notion of 'class' is similar to the notion of 'species'. Now, to say 'the notion of species', without qualification of context, could easily lead us to an equivocation, for there are some very different notions of species (just as there are different notions of 'class'). We will examine the notion of species held by Plato, by Aristotle, and by Darwin. All of these are all superficially similar to the object-oriented programming notion of class (and to each other), to the extent that each has a *designation*, an *intension* and an *extension* (terminology fairly common in computer science and used in [23]). Plato's notion of species is mostly metaphysical (*Ideal Form*); Aristotle's is logical, biological, and metaphysical (*universal*); and Darwin's is primarily biological. We will examine the difference in the notions of species according to three criteria: the *primary relations* associated with the notion of species; the *metaphysical importance* of the notion of species to that thinker; and the *criteria* of what constitutes a species. Programming, of course, is a practical activity and so classes do not have any metaphysical importance. However, this criteria is important to understanding these different notions of species, and may provide some insight into object-oriented programming.

For some time the object-oriented programming research community has made a distinction between *sub-classing* (code sharing), *subtyping* (interface sharing), and *is-a* (a conceptual relation) (e.g. [20, 32]). This is an important distinction, but it does not transfer well to these three notions of species. Furthermore, these three ideas tend to overlap in most common class-based programming languages. Consequently, for lack of better terms, we will use the terms 'subclass' and 'super-class' in this discussion, but this should not be read as 'sub-classing = implementation inheritance'.

At the outset, it is worth noting that the notion of 'class' in computer science is acknowledged to be different than the notion of 'set'. The latter is strictly extensional, whereas the former is both extensional and intensional (e.g. [24]). A set's identity is determined solely by its membership (extension), whereas a class's identity is determined (at least in part) by its intension. In the writings of Bertrand Russell and, consequently, much of mathematics and logic in the twentieth century, the word 'class' is taken to mean 'set', in contrast to the interpretation used in object-oriented programming. Rose notes that Russell's notion of 'class' as 'set' is not be read into either Plato or Aristotle [27, p.6].

Plato Plato's notion of species is translated in English as *Ideal Form*. This notion of species is similar to Aristotle's: the difference is primarily metaphysical. In coming to their respective notion of species, both Plato and Aristotle were motivated to solve one of the central problems in Greek philosophy: how to reconcile the intelligibility of the real world with the fact that material beings are constantly changing.[5] Plato attempts to solve the problem by saying that changing individuals are material imitations of the *Ideal Forms*, which are eternal and unchanging (i.e. have neither a beginning nor an end in time). The primary relation associated with Plato's notion of *Ideal Form* is that of material singulars which are said to 'participate in' or 'imitate' the *Ideal Form*. The notion of 'class' in object-oriented programming is Platonic to the extent that classes pre-exist objects in terms of program execution (as the *Forms* pre-exist material singulars), and that classes are used 'as a template for generating objects'. We may not say that prototype-based languages are Platonic, however, they deal only with material singulars (objects): to say that one singular is an imitation of another is entirely different than a material being participating in an *Ideal Form*.

Aristotle Aristotle, like his teacher Plato, insists that truth is eternal and unchanging, and this is also captured in his notion of the *universal*. However, Aristotle views material beings as what is 'really real', and the universal as an abstraction which exists only in the mind. This is why Aristotle refers to singulars as *primary substance* and the universal as *secondary substance*; it is also the reason why the Aristotelian position is sometimes referred to as 'metaphysical realism'. Aristotle offers a more reasoned solution to the problem of reconciling the intelligibility of the real world with the constant flux of material beings by showing that the material world is logically intelligible in terms of the *natures of things* (which are expressed in the universal).

Since both species and genera are eternal and unchanging, there is no temporal ordering between them, nor is one dependent on the other. The species, however, is always first in the order of coming-to-know (after singulars, of course).

We may say that classes in object-oriented programming are Aristotelian insofar as the programmer first comes to know individuals (objects) in the problem domain and then develops abstractions (classes) that contain them. We may say also that the notion of class in object-oriented programming is Aristotelian to the extent that classes seem to have a secondary ontological status to objects.

It is important to keep in mind that a class developed by a programmer for the sake of solving a practical problem is not intended to be some representation of eternal unchanging truth, even if the programmer usually hopes that the class will transcend time and changes in the problem domain. Moreover, the act of programming does not force the programmer to take a metaphysical position on the true nature of reality.

There are two primary relations associated with Aristotle's notion of *species*: the relation of species to genus, and the relation of individual to species. Both of these relations are relations of logical abstraction: the species represents the intelligible form of the singulars contained under it, and the genus represents the more abstract intelligible form of the species contained under it. The term *instantiation* may be used to describe both the relation of individual to species and the relation of species to genus, although it refers more properly to the first rather than the second. In this sense the Aristotelian notion of genus corresponds to some notions of *meta-class*, such as that present in Telos [25].

The contemporary use of the prefix *meta* originates only indirectly with Aristotle. The original Greek meaning of meta is 'after'. We now use meta, in the sense of metaphysics or metadata, to mean 'of a higher order'. This usage of the term is derived from Aristotle's book on the *Metaphysics*. Aristotle never used the term metaphysics, but instead referred to the subject of that work as 'first philosophy' or 'theology'. The term metaphysics originates from Andronicus of Rhodes (first century BC), Aristotle's first editor, who placed the book on the *Metaphysics* after the book on the *Physics* in his compilation. If we look at the distinction between Aristotle's *Physics* and *Metaphysics* we find that both treat of the same subject matter, but in different respects: the former is concerned with *proper causes* and the latter with *ultimate causes* (or first principles). For our current considerations, an analogous relation would be to say that

[5] Paramenides and Heraclitus take radically opposing positions on this problem. Paramenides says that all change is an illusion, and Heraclitus says that change alone is real. Aquinas refers to such extremes as *beautiful errors* because their juxtaposition permits us to understand the true nature of the problem.

'man is a species', where 'man' indicates a class and 'species' indicates a 'meta-class'. Aristotle's notion of genus should not be looked at in this contemporary meaning of 'meta': an Aristotelian genus is much closer to the idea of an *abstract super-class*, since it cannot have direct singular instances.

Darwin Darwin, primarily a biologist, restricted his notion of species to living things, whereas Plato and Aristotle use the term in a much broader way. Darwin was not a metaphysician, but his notion of biological species has of course caused much controversy over what may be called metaphysical issues. As is commonly known, the primary relation associated with Darwin's notion of biological species is *evolution*.[6]

The 'inheritance' mechanism in object-oriented programming has been likened to this Darwinian notion of evolution (e.g. [34]). As we will see below, considering super-class according to Darwin's biological relation leads to a much different notion than if it is considered according to Aristotle's logical relation.

The primary tenet of Darwin's hypothesis is that it is possible for one species to evolve into another species or, more properly speaking, that the individuals of one species evolve into the individuals of another species. For example, one may say that *homo sapiens* evolved from *homo erectus*, which in turn evolved from *homo habilis*, and so on. Such a relation has some interesting properties:

- Members of both the old and new species can exist simultaneously.
- Usually the old species becomes extinct.
- The features of the new species are not a strict superset of the features of the old species: some features may no longer be required due to changes in the environment (e.g. ice-ages).
- Sometimes the new species is more complex than the old species. Sometimes the new species is merely an adaptation to environmental conditions, and not necessarily more complex.
- The relation is always between two species rather than between a species and a genus. To say that '*homo sapiens* is in the genus *homo*' would use the Aristotelian relation of logical abstraction. Consequently, the old species is not more abstract than the new species: both are equally abstract/concrete.
- It is possible for a species to evolve into a different species *in a different genus*. That is, a species in a genus other than *homo* could evolve into a species within the genus *homo*. Note again that 'genus' is a logical abstraction (and an Aristotelian notion).

One obvious and important metaphysical difference between the Platonic/Aristotelian notion of species and that of Darwin is that the former represents eternal unchanging truth, whereas the latter does not. The implication of this is that Darwinian species are only accidentally different from one another (in the Aristotelian sense of accidental and essential).

[6]The word evolution is much older than Darwin, and its first meaning is of the growth and development within an individual.

The object-oriented programming relation between a sub-class and a super-class has been compared to the Darwinian relation of evolution, with the conclusion that the two are similar [34]. This comparison took the position that a sub-class is an *incremental modification* of a super-class. If this is indeed the intended relation between a sub-class and a super-class, then there is a strong similarity between the two relations. However, one must be careful to distinguish between *what a class represents* and *how a class is represented*: that is, between the concept signified by the name of the class and the text fragment that represents the class to the compiler. Certainly the text fragment that represents a sub-class may be considered as an incremental modification of the text fragment that represents the super-class. However, it is another thing entirely to think that the concept the sub-class represents is an incremental modification of the concept that the super-class represents.

Darwin's relation of evolution is most similar to the relation between 'parent' and 'child' objects in prototype based programming languages: here the relation occurs between individuals, which is where Darwin's relation properly occurs as well, and there is a stronger sense of transfer between them.

Abstraction and Evolution There are many terms used to describe the relation of sub-class to super-class in object-oriented programming. In Table 1 we evaluate some of these terms against the Aristotelian relation of logical abstraction between species and genus, and the Darwinian relation of biological evolution between two species. A check ($\surd$) indicates that the description is appropriate for the relation, while an x ($\times$) indicates that it is not. The table contains more terms that we have space to discuss in the text: only the most important and interesting terms are given further discussion.

Description	**Abstraction** logical species / genus Aristotelian	**Evolution** biological two species Darwinian
is-a	$\surd$	$\times$
kind-of	$\surd$	$\times$
specific / general	$\surd$	$\times$
Liskov substitutability	$\surd$	$\times$
variability / commonality	$\surd$	$\times$
abstraction	$\surd$	$\times$
specialization	$\surd$	$\surd$
incremental modification	$\times$	$\surd$
adaptation	$\times$	$\surd$
mutation	$\times$	$\surd$
extends	$\times$	$\surd$
derived / base	$\times$	$\surd$
complex / simple	$\times$	$\surd$
new / old	$\times$	$\surd$
evolution	$\times$	$\surd$

Table 1: Comparison of abstraction and evolution.

The phrase 'is-a' is applicable to the Aristotelian relation, but not to the Darwinian one: we would not say that '*homo sapiens* is a *homo erectus*'. However, the word 'is' (the verb 'to be') is known to be problematic in both philosophy and computer science (e.g. [9]). The phrase 'incremental modification' is only applicable to the Darwinian relation: we may say '*homo sapiens* is an incremental modification of *homo erectus*', but we may not say that 'man is an incremental modification of animal' (and here it is extremely important to understand that animal is an abstract idea, i.e. a genus).

The Liskov principle of substitutability, that an instance of a sub-class may be used wherever an instance of the super-class is expected, requires that we be able to conclude the transference deduction with certainty. This requirement only holds for the Aristotelian logical relation of abstraction — it does not hold for the Darwinian relation of evolution. For example, there are things that can be said of *homo erectus* that cannot be said of *homo sapiens*, whereas there is nothing that can be said of animal essentially that cannot also be said of man or of any other kind of animal.

The term 'specialization' is unique in Table 1 because it is the only term that appears to be suitable for both the Aristotelian logical relation of abstraction and the Darwinian biological relation of evolution. This appearance, however, is only skin deep — the term is applicable to both ideas only if it is used equivocally. 'Specialization' has at least two distinct meanings, one that refers to the Darwinian relation, and one that refers to the Aristotelian relation. (Although 'specification' is a more accurate term for the Aristotelian relation, this is a different sense of 'specification' than is commonly used in computer science). 'Generalization', on the other hand, is applicable only to the Aristotelian relation: we could not say '*homo erectus* is a generalization of *homo sapiens*'. Finally, it is worth noting that UML uses the terms 'specialization' and 'generalization' rather than 'inheritance'.

On Interface and Implementation 'Inheritance' Given the previous analysis, it may be tempting to conclude that the Aristotelian relation of logical abstraction corresponds to 'interface inheritance' and that the Darwinian relation of evolution corresponds to 'implementation inheritance'. This conclusion, however, is not warranted. Although the Darwinian relation seems quite suited to the idea of 'implementation inheritance' because of the metaphor of genetic transfer as 'implementation', it does not follow that a relation between programming classes which is modeled on the Aristotelian relation could not involve code-sharing, nor does it follow that a relation between programming classes which is modeled on the Darwinian relation could not involve interface sharing.

When 'implementation inheritance' and 'interface inheritance' are held as *contrary terms* there is a twofold distinction: the former implies code-sharing whereas the latter does not; and that the latter one alone allows the transference deduction to be concluded with certainty. The Aristotelian logical relation allows us to conclude the transference deduction with certainty in all cases, but some deductions still hold under the Darwinian biological relation, e.g.:

> *Homo erectus* stands upright.
> *Homo sapiens* evolved from *homo erectus.*
> ∴ *Homo sapiens* stands upright.

The reason that we cannot always conclude with certainty when using a Darwinian-like relation is that, for Darwin, species are only accidentally different — whereas for Aristotle, genus and species are related essentially. When we employ an Aristotelian relation in a transference deduction, we immediately see how the premises are the 'cause' of the conclusion. Such 'causation' is not always present in the Darwinian-like relation, for example:

> Uni-cellular organisms are asexual.
> Man evolved from uni-cellular organisms.
> ∴ Man is asexual.

Beta and Smalltalk In programming languages like BETA the intended relation between sub-class and super-class seems closer to the Aristotelian relation of logical abstraction, whereas in languages such as Smalltalk it seems closer to the Darwinian relation of evolution. This seems evident both in language used to speak of them and in their design. For example, the BETA `inner` construct and static typing make it easier to conclude the transference deduction with certainty — whereas arbitrary method overriding and dynamic typing in Smalltalk make the conclusion of the transference deduction less certain. Of course, it is possible for a programmer to use the relation between sub-class and super-class in either programming language in a fashion which is either more Aristotelian or more Darwinian. Interestingly enough, this choice has been referred to as an 'essential' use of inheritance vs an 'accidental' use of inheritance [30].

2.2.3 On the Notion of Inheritance

The term 'inheritance' is often used in a metaphorical fashion in object-oriented programming to describe the relation between a sub-class and a super-class. Here we compare the meaning of the word inheritance with five kinds of transference relations: that of a prototype and its imitation; incremental modification; the Darwinian relation of evolution between species; the Aristotelian relation of logical abstraction between a species and a genus; and the Aristotelian relation of logical abstraction between an individual and a species. We include the incremental modification relation here because it has been said to be the 'essence' of inheritance (or, more precisely, incremental modification in the presence of a late-bound self-reference [35, 32]). We find that 'inheritance' seems to be an acceptable metaphor for the first three relations, but that it is an exceedingly poor metaphor for the Aristotelian relations of logical abstraction.

The Meanings of the Word The word 'inheritance' can be used to signify a number of different things in English. Its first meaning is from economics, and signifies external material possessions passed down from parent to child (usually on the death of the parent). In this first meaning, there are five important elements to note: something *external* is *transferred* from one *individual* to another; the two individuals are in a *temporally ordered* relationship; and there is a real *dependence* of the recipient upon the donor. Subsequent English usages speak to 'inheritance' in a biological way (heredity), or to social or cultural traits (heritage) passed down from one to another, and such uses are at least as old as Shakespeare. When we move from external goods to passing down either physical or social traits, we seem to be using 'inheritance' in a metaphorical way. This is also the case when we speak of 'inheritance' between groups of people, although in the English usage of the word, this is said usually because of some real transference between the individuals in those groups (and this does not deviate as far from the original meaning as does speaking of a relation strictly between universals).

Evaluating the 'Inheritance' Metaphors In Table 2 we summarize our analysis of 'inheritance' as a metaphor for the five transference relations given above against the five criteria for the meaning of the word, also given above. The five criteria are indicated as follows: **Tr** indicates *transference*; **Ext** indicates *external* goods; **Ind** indicates *individuals*; **Time** indicates *temporal ordering*; and **Dep** indicates *dependence.* A check ($\surd$) indicates that the usage conforms to the original notion; an x ($\times$) indicates that it does not meet the criteria; and a bullet ($\bullet$) indicates that elements of the original notion are present in some fashion.

As a metaphor for:	**Tr**	**Ext**	**Ind**	**Time**	**Dep**
Prototype/imitation	$\surd$	$\times$	$\surd$	$\surd$	$\surd$
Incremental modification	$\surd$	$\times$	$\surd$	$\bullet$	$\surd$
Darwinian species/species	$\surd$	$\times$	$\bullet$	$\surd$	$\surd$
Aristotelian species/genus	$\surd$	$\times$	$\times$	$\times$	$\times$
Aristotelian ind./species	$\surd$	$\times$	$\times$	$\times$	$\times$

Table 2: Evaluation of 'Inheritance' Metaphors

As can be seen from Table 2, all of these metaphorical usages of 'inheritance' retain the notion of *transference*, and none of them speak of *external* goods.

As a Metaphor for the Transference Relation between Prototype and Imitation This is the case in which the inheritance metaphor seems most suitable: there is a transference between individuals that exist in a temporally ordered relationship, with real dependence of the recipient upon the donor. The only discrepancy with the original meaning is that the transfer is not of external goods. However, this metaphorical use of the word 'inheritance' is in accordance with some of the commonly accepted metaphorical uses of the word in English.

As a Metaphor for Incremental Modification The 'inheritance' metaphor seems to be suited to the idea of incremental modification, notwithstanding the use of the word 'class' to describe a fragment of program text (and we will now substitute the word 'module' for 'class' in this paragraph). Clearly the derived module is dependent on the base module. The phrase *incremental modification* implies that the derived module is written after the base module but, due to refactoring, this is not always the case (hence the $\bullet$ in Table 2). There is, however, a potential problem if we use the term 'class' for a fragment of program text and use the term 'incremental modification' to describe the relation between a sub-class and a super-class. We have to be clear on whether 'incremental modification' refers to the concept that the name of the class signifies, or whether it refers to the fragment of text that represents the class. If we model the relation between sub-class and super-class on the Darwinian biological relation of evolution, then 'incremental modification' is an apt term for both. If, on the other hand, we model the relation between sub-class and super-class on the Aristotelian logical relation of abstraction, then we have a problem: the text that represents the sub-class may be an incremental modification of the text that represents the super-class, but the idea represented by a species *is not* an incremental modification of the idea represented by its genus. We cannot say that 'man is an incremental modification of animal'. This kind of error involves switching modes of supposition within an argument (a topic we will address later).

As a Metaphor for the Darwinian-like Relation between two Species The 'inheritance' metaphor in programming has been likened to the Darwinian-like relation of two species, with the conclusion that the metaphor is acceptable (e.g. [34]). As Table 2 shows, this metaphor is in accordance with commonly accepted metaphorical uses of the term in English: it clearly meets the temporal ordering and dependence criteria; and it speaks of a transference between species because of a real transference between individuals (an allowance that is made in some common English usages of the word).

As a Metaphor for the Aristotelian Logical Relation of Genus and Species As Table 2 shows, the only common criteria between the meaning of the word 'inheritance' and the Aristotelian logical relation of genus and species is that there is some kind of transference involved. As explained previously, there is no temporal ordering between genus and species, neither is there dependence of one on the other; and obviously they are not individuals. Moreover, the individuals of the species belong also to the genus, so we cannot appeal to the idea of a transference between universals based on some real transference between the individuals (as we can for the Darwinian relation of evolution).

As a Metaphor for the Aristotelian Logical Relation of Species and Individual 'Inheritance' is also a poor metaphor for the Aristotelian logical relation of abstraction between a species and an individual. The only thing that the two ideas have in common is some notion of transference.

One of the important points of this paper is to show that the notion of transference is extraordinarily common, and that this basic notion is involved in more sophisticated ideas that differ from each other essentially. It is important that we see the differences between these things that are similar, as well as seeing the similarities between these things that are different.

2.2.4 *Accidental Relations: Proper and Common*

The non-essential tells us not what something is but, instead, *what belongs to* or *inheres in the subject* in some way. From the viewpoint of definition, even these non-essential connections can help us to distinguish adequately one kind of thing (i.e. species) from another. These non-essential connections, also called *accidents* (that which *occurs in* a subject), are of two kinds: *proper accidents* (or *property*) which coincide with the species (definitum), and *common accidents* (simply called *accident* when contradistinguished from a *property*).

A *property* is something that is not the essence of the definitum and yet *belongs to that species alone, and is convertible with the logical subject* (i.e. can be used wherever the logical subject is used). Some examples of properties are 'man forges tools' and 'gold is the most malleable metal'. There is a close relation between essence and proper accident, since a proper accident exists in a subject because of its essence. It is because the essence of man is to be 'rational animal' that activities such as 'grammatical', 'risible', 'self-determining' are properties that flow from his nature.

The common accident can be contingent either *because it is found in only some of the individuals in a species* or *because it is found in individuals of different species*. If it is found in only some individuals in a species it is 'too small' to be used for defining. If it is found in more than one species it is 'too great' to sufficiently differentiate the species within a genus. However, because what is contingent cannot serve as a basis for differentiating a species within a genus, we should not be too quick to dismiss the value of such accidents found within different species. Combinations of these accidents may occur only within the definitum. Plato defines man as 'a featherless, furless, biped with fingernails and toenails'. Such a definition can teach us something very important in terms of defining by accidents: what is all important is the specifying difference. If the specifying difference does not adequately distinguish the definitum within its genus, then either it includes things which should not be included or excludes things which should not be excluded. For example, if we try to specify 'man' with just 'furless biped' then we will include things which should not be included in the species man (e.g. birds). To specify 'man' as only 'featherless and furless' would include snakes and naked mole rats, amongst other creatures.

Most programming languages do not allow the programmer to distinguish between *proper* and *common* accidents. From the standpoint of logic we see this as a limitation, as do Grogono and Sakkinen from more practical grounds. They argue that identifying 'essential' and 'accidental' attributes to the compiler facilitates the automatic generation of semantically appropriate copy and clone operations [17]. They note a twofold distinction of 'essence' and 'accident'. They do not consider this sufficient for their purposes. They refine this to a fourfold distinction with three normal cases and one abnormal one (for the programmer to override the default behaviour). Perhaps their three normal cases correspond to 'essence', 'proper accident' and 'common accident'.

2.3 Determination of the Categories

To define well *what something is* (using our terms univocally) we need to talk about its real being. Now, 'whatever is' must exist in one of two ways. Aristotle designates these two modes of real being as *substance* and *accident*. These mutually exclusive notions exhaust the whole of 'whatever is' and must be seen as co-relative. The logical notion of *substance* is 'that to which it belongs to exist in itself and not in another'; the logical notion of *accident* is 'that to which it belongs to exist in another and not in itself'. 'Substance', in English Aristotelian thought, is a sometimes criticized translation of the Greek *ousia* — a word that is linked etymologically to the word for 'being'. Consequently, 'substance' has a broader meaning in English Aristotelian thought than does 'matter'. The meaning of 'substance' in the BETA conceptual framework [23] is closer to the Aristotelian notion of 'matter' (although this is an issue worthy of further study).

Now since classifying, or defining, is to say what *kind of something* the subject is, we must bear in mind the distinction between the *universal* and the *singular*: only the *universal* is predicable of a subject; the *singular*, never a predicable, is the ultimate subject of which everything is said. These distinctions are illustrated in Table 3.

that to which it belongs to ...	**Substance** *exist in itself and not in another*	**Accident** *exist in another and not in itself*
Universal *be predicable of a subject*	man	colour
Singular *be the subject of what is predicable*	Socrates	the paleness of Socrates

Table 3: Universal and Singular contrasted with Substance and Accident

Aristotle notes [*Categories* §2] that the universal is *predicable of* a subject whereas the singular is not. The universal 'man', for example, is predicable of, or can be used in defining 'Socrates'. The singular 'Socrates' is not predicable of anything, and so cannot be used to define. Łukasiewicz, who takes a different view, claims that the singular is predicated of something and supports this with four examples from Aristotle: 1) 'that white object is Socrates', 2) 'that which approaches is Callias', 3) 'Socrates is Socrates', 4) 'Sophroniscus was the father of Socrates' [21, p.6]. In each of these examples, however, the relation is one of identity and not predication. Operational statements in mathematics, e.g. '$1 + 1 = 2$', are also statements of identity or sameness. '2' is not being seen as predicated of '$1 + 1$' but is seen as another way of saying '$1 + 1$'. On the contrary, when we predicate one thing of another, the predicate must be considered, necessarily, as greater than the subject because we 'situate' the subject in the context of what *kind of thing* it is.

2.3.1 Etymology of 'Category'

Category comes from the Greek word *kategoria*, literally, 'to speak in the agora'; *agora* is the Mediterranean notion of the 'plaza', the daily gathering place for the ancient Greek neighbourhood or community to discuss political, religious, and commercial affairs. *Kategoria*, involved 'speaking about something in a public forum' (*forum*, in fact, is the Latin equivalent of *agora*).

Aristotle's treatise on the *Categories* is primarily ordered to speaking properly about any subject, that is with classifying or situating any subject in its proper place. Our English word, *predicament*, is derived from the Latin *prædi-*

care which also emphasizes 'asserting publicly', or 'speaking about things with others'. This is why Aristotle's treatise on the *Categories* is entitled *Prædicamentorum* in Latin.[7] Both the Greek *kategoria* and the Latin *prædicamentorum* originally refer to dialogue (in its turn, ordered to *logos*). To escape multiple monologues it is necessary to keep the discussion focused on the subject. Developing this focus is Aristotle's purpose in the *Categories*. If the contemporary meaning of *category* is closer to 'class' or 'type', then we can say that Aristotle's concern in the *Categories* is with classification.

2.3.2 Aristotle's Ten Categories

Aristotle's logical categories are what we would call *supreme genera*, sometimes referred to in computer science as *top level ontological categories* (e.g. [31]). For example, if we classify humans as animals, and animals as living things, and living things as natural substances, we can see that the supreme category is *substance*. SUBSTANCE is the ultimate category for whatever exists in itself and not in another (note that substance and accident are *modes of being*, not species of being).

For that which exists in another and not in itself, that is, for what belongs to or inheres in a substance, Aristotle posits nine supreme genera of *accidents*. These are not some arbitrary enumeration, but categories at which he arrives by careful divisions.

Accidents may be divided into *intrinsic* and *extrinsic*. Those which are intrinsic may be subdivided into those *absolutely* intrinsic and those *relatively* intrinsic. Those accidents absolutely intrinsic to a subject are its *matter* and its accidental *form*, and these give rise to the categories of QUANTITY and QUALITY, respectively. What is relatively intrinsic is something that inheres in the subject but which refers us to another, and this is what belongs to the category of RELATION.

The extrinsic accidents he divides *according to causality*, *according to measurement*, or *according to neither causality nor measurement*. What is divided according to causality looks at the subject either as a cause (ACTIVITY), or is in the subject as an effect (PASSIVITY). The division of accident according to measurement can be either *according to time* (WHEN), or *according to place*: the latter can be subdivided either *absolutely* (WHERE) or *relatively* (POSITION — that is according to the disposition of its internal parts). The final category, that which belongs to the subject neither according to causality nor according to measurement is what Aristotle designates as HABIT (in the sense of wearing clothes or decoration).

Commentators in the Middle Ages made further sub-divisions but the categories themselves were never challenged until Kant did so in his *Critique of Pure Reason*, by adopting an *a priori* viewpoint. Those who wish to pursue different structural interpretations of reality might want to consult John Sowa's new book, which gives an overview of Aristotle, Kant, Peirce, Husserl, and Whitehead within the context of computer science [31].

2.4 Logical Division

We have continued to stress that all of the considerations of the first operation of reason are for the sake of defining well. To do this we must know *what kind of thing* each thing is, hence we must see what distinguishes, separates, or divides, one species from all others within the same genus. The necessary tool in order to achieve this is division. Division is first developed by Plato, and is evident in his definitions of the *sophist* and the *angler*. Rose explores this relation in depth [27], and suitably so, for as Sir David Ross insists:

> Aristotle's translation of Plato's metaphysical doctrine into a doctrine from which the whole of formal logic was to be developed is a most remarkable example of the fertilization of one brilliant mind by another. [29]

Aristotle speaks of two fundamental kinds of division: the division *of a universal whole into its specific parts*, and the division *of an integral whole into its composing parts*. The first kind of division is that of genus into species. The second, of a whole into its composing parts, is referred to in the computer science literature as aggregation, composition, has-a, consists-of, contains, and is-a-part-of. Logical division, as is all human activity, can be done well (or perhaps with greater ease, done poorly). Traditional logic arrives at some common sense rules for making good divisions:

1. The parts must be inferior to the whole. By definition, a part cannot be greater than that of which it is a part. This emphasizes what is meant by 'part' and what is meant by 'whole'. This does not mean that we can divide into singulars, however, because the difference between singulars are only material — whereas form makes something to be the kind of thing that it is.

2. The division must be exhaustive. This rule states that the members of a division must exhaust the whole which is being divided. In the practice of programming we often simply *enumerate* some species within a genus rather than making a complete division of the genus. For example, a program that is only concerned with rectilinear shapes may not define a circle class (and hence does not fully divide the genus of shape). We take similar shortcuts when dividing an integral whole into composite parts, because the program is only a *model* whose purpose is to solve a practical problem, rather than to initiate a philosophical investigation of the subject matter.

3. The parts must be formally opposed. 'Mutually exclusive' is the intent of 'formally opposed' in getting to the purpose of this rule. Any division in which the dividing members overlap (such as dividing person into student and female) is a bad division.

[7] We prefer to keep the term 'predicament' rather than 'predicate' because the later belongs more properly to the vocabulary of the second act of reason, where something is predicated of a subject with truth or falsity. The use of *predicate* in modern mathematical logic (i.e. first order predicate calculus) refers to *function* (in the mathematical sense). This meaning originates with the nineteenth century German mathematician Gottlob Frege.

Opposition may be between different subjects in a genus or differences that exist in a subject. Every opposition, however, must be either negative or affirmative, between a mode of *being* and *not-being* or between a mode of *being* and *being.* The two modes of opposition between subjects are *contradictory* and *contrary.* The two modes of opposition in a subject are *privative* and *relative.* These distinctions are illustrated in Figure 1, where plus (+) indicates 'being' and minus (−) indicates 'not-being', so plus/minus (+/−) indicates opposition between 'being' and 'not-being'.

$$\textbf{\textit{Opposition}} \begin{cases} \text{between subjects} \begin{cases} \textit{contradictory} \quad (+/-) \\ \textit{contrary} \quad (+/+) \end{cases} \\ \\ \text{in a subject} \begin{cases} \textit{privative} \quad (+/-) \\ \textit{relative} \quad (+/+) \end{cases} \end{cases}$$

Figure 1: Modes of Opposition

Contradictory opposition is a pure opposition which divides perfectly because it involves affirmation and negation of being with respect to the same subject; for example something is either *rational* or *non-rational.* Contrary opposition is between two positive terms which are extremes *within* the same whole which is being divided, for example, between *rational* (according to reason) and *absurd* (contrary to reason), or dividing sex into *male* and *female.*

Privative opposition is between the presence and absence of a characteristic within a subject which is intended to have that perfection which is lacking in it. For example, *rational* and *irrational* activity: only that which is intended by nature to act rationally can act irrationally. Similarly, blindness is a lack of seeing, but it can be said only of that which by nature is intended to see.

The fourth type of opposition is not truly opposition for one member does not exclude the other, but it still *refers* to or *relates* to the other. The simplest example of relative opposition is in family relations; for example, *parent* and *child.* One is not a child unless there is referral to a parent, or a parent unless there is a child (notice also that one can be both parent and child at the same time, but in different respects).

The kind of division used determines the view of the definitum and a division is only as strong as the mode of opposition employed in making it. For example, to divide animals by contrary opposition into those that *live on land* and those that *live in water* is not a good division since amphibious creatures do both, and flying creatures do neither.

Currently, there is no one way to express different kinds of opposition in programming languages or knowledge representation formalisms. Exceptions may be used to express privation, but may also be used for other purposes. We know of one work that considers explicit expression of contradiction in object-oriented programming [1].

4. There must be a consistent basis in each division.

The fourth rule, that there must be a consistent basis kept throughout the division, is the most subtle, and so we will first give an example: student may be divided *by sex* into male and female; *by degree* into graduate and undergraduate; *by faculty* according to the organization of the university; and so on. The different terms result from a different basis of division of the subject. An example of a division made without a consistent basis of division would be of 'human' into *respiratory system* and *appendages*: the first division is based on function, the second is based on mode of attachment.

The idea of the basis for the division is represented in object-oriented programming most closely by the UML notion of *discriminator.* The idea is also captured in some knowledge representation formalisms, such as Loom [22], which names it *partition.* In Loom, if a genus is divided according to different bases, the tool ensures that these species may not be subsequently merged. We think that programming languages should allow for explicit representation of the basis for division, and should enforce some rules with respect to it (as is done in Loom).

2.4.1 Squares, Rectangles and Abstract Super-classes

Winkler [36] initiated a discussion in *Communications of the ACM* a few years ago as to whether Square should be a sub-class of Rectangle or Rectangle should be a sub-class of Square. The former approach was thought to follow a 'concept-oriented view', whereas the latter represented a 'program-oriented view', because the dictionary defines a square as a type of rectangle, but only one data member is necessary for a Square class and two are required for Rectangle. Grosberg [18] resolved this problem by saying that a class that has sub-classes may not have direct instances and, conversely, that a class that may be directly instantiated may not have sub-classes. Hürsch [19] later formulated this as the *abstract super-class rule*, which his analysis determined was a good programming guideline.

The *abstract super-class rule* is in direct accordance with the Aristotelian ideas that we have just expounded. First of all, the Aristotelian notion of genus is very much like an abstract super-class; for example, there is nothing of which we can say that it is an animal without being able also to specify *what kind* of animal it is. Secondly, having a single sub-class violates the rules of logical division: a thing cannot be divided into one; it must be divided into at least two — otherwise the thing being divided is not an abstraction. The *abstract super-class rule* forces one to divide the problem in a manner similar to Figure 2, where right-angled, four-sided geometric figures are divided into equal-sided (Square) and non-equal-sided (Rectangle).

$$\text{Right-Angled} \begin{cases} \text{four sided} \begin{cases} \text{equal} - \text{sided} \\ \text{non} - \text{equal} - \text{sided} \end{cases} \\ \\ \text{non-four sided} \end{cases}$$

Figure 2: Division of right-angled geometric figures.

2.5 Definition

The culmination of the first operation of understanding, and the reason for all of our prior considerations, lies in the art of *definition.* Definitions may be either *nominal* or *logical.*

Nominal definitions are concerned with the material aspects of the words themselves, rather than with the concepts which the words represent. Nominal definitions may be divided into *etymological* (explaining the origin of the word) and *common usage* (describing the various ways in which people have used the word).

Our concern is with logical definition, which attends to the concept that is signified by the word. We now know that a logical definition must contain both a proximate (immediate) genus and a difference that adequately distinguishes the definitum in a specifying way. There can be many different kinds of logical definitions, because of the different ways in which they specify the definitum.

There are two ways to define something such that the specifying difference is in terms of *intrinsic* principles. The first is the *essential* definition, which specifies or tells us in the definition the kind of thing the definitum (species) is. For example, in the statement 'man is a rational animal', the species to be defined (the definitum) is 'man', the proximate genus is 'animal', and the specifying difference is 'rational'.

The second kind of definition according to what inheres in the subject, is a *non-essential* definition which can be either by a *proper accident* or a *common accident* (property or accident). Both of these tell us something about the subject other than what it is. Since a property belongs in an exclusive way to the species, it too provides a distinctive difference; for example, 'man is a tool-forging animal'. The definition by common accident can adequately distinguish a definitum (species) only when multiple trans-specific accidents coincide in *this* species; for example, Plato's definition of man 'featherless, furless biped with fingernails and toenails'. These are the three kinds of definition which use *intrinsic* principles as specifying differences.

The definitum (species) can also be adequately differentiated within its genus by what is *extrinsic*; for example, by *operational* or *causal* definitions. The operational definition designates (or at least implies) a standard of measure and an indication of the specifying result; for example, 'man is an animal with forty-six chromosomes'. In the experimental sciences, which use operational definitions, subjects are defined in terms of the way in which they are measured and the results of that measurement. For example, water may be defined as a liquid that boils at a certain temperature and freezes at a certain temperature (note that these temperatures must be given with respect to a pressure). The causal definition must employ at least one of the extrinsic causes (the agent or the end); for example, 'man is an animal who chooses what he sees as good in order to be happy'.

In the Aristotelian tradition, the specifying differences for artificial things can be given only in terms of what is extrinsic to the subject since artifacts are dependent for their being on the agent and his intention. Some computer scientists have commented on the difficulty of attempting to define artificial things intrinsically; for example, Wegner on the notion of table [34, p.551], or Sowa on the notion of chair [31] (which must be distinguished from a toilet by its final cause or purpose).

Definition is critical to both a science and a program, because neither the one nor the other is distinguished by its subject matter, but instead, by the way in which it defines or treats its subject. Human nature is the subject of both anthropology and psychology: these sciences differ by the way in which they define the subject, rather than by the subject itself. Likewise, many programs may be concerned with the same logical subject but in different respects, and because of this will want to define the subject in a different light. We think it would be of benefit to programmers to be able to explicitly express the logical definition(s) of the subject(s) within programs.

3. THE SECOND ACT: PREDICATION

If we consider a class as a logical subject (i.e. as a species or as a genus), then we may also consider attributes and operations (fields and methods) as things that we say, or predicate, of a subject. This is the concern of the second operation of reason.

Once we have grasped adequately the notion of the subject we are speaking about, we can now begin to make a statement about it — an activity called *predicating* in the Aristotelian tradition. Statements are often referred to as propositions but, strictly speaking, a statement becomes a proposition only when it is *proposed for the sake of* some conclusion (a notion belonging to inference in the third act of reason). At the risk of oversimplification, we will affirm that the logical statement is composed of three parts:

- *subject* (that of which something is said)
- *predicate* (that which is said of a subject)
- *verb copula* (the verb 'to be')

The verb 'to be' is important in the proper formation of a statement because we address the *being of the subject* when a predicate is affirmed or denied of the subject by means of the verb copula. A statement in its proper form looks like this: 'man / is / an animal'; or 'man / is not / a machine'. A statement such as 'our linesmen play tough defence' in its proper statement format would be 'our linesmen / are / players who play a tough defence'.

Because a statement affirms or denies something about the very being of the subject, every statement we make has the property of being either *true* or *false* (something determined by what is predicated). The elements of the statement (either the subject or the predicate) by themselves, are neither true nor false. Only when the composite expression (the statement) affirms or denies the predicate of the subject does truth enter in.

3.1 Divisions of Statements

Traditional logic divides statements (propositions) according to their kind of *unity* — how they are one, either absolutely or relatively. Those which are absolutely one are called *categorical statements* (named from *category* as discussed earlier) and these are divided *according to quality*, *according to quantity*, and *according to matter.* We will now examine each of these divisions.

the division of the categorical statement according to

unity { *absolute* categorical
relative compound

A statement is *absolutely one* when it predicates one thing (thus called the predicate) of another (called the subject) by means of the verb copula. Such categorical statements are necessarily true or false depending on the kind of matter in the content of what is predicated.

Statements which are *relatively one* are called compound statements because their elements are categorical statements linked (copula) by some *operational connective*, rather than the verb copula; for example, **if** humans are rational, **then** they are able to create works of art. The principal kinds of compound propositions are the *conditional* (**if** <categorical> **then** <categorical>), the *conjunctive* (**both** <categorical> **and** <categorical>), and the *disjunctive* (**either** <categorical> **or** <categorical>). Each species of compound proposition is treated as an affirmative and each has distinctive properties of truth or falsity which depend on the copula.

Current object-oriented programming languages usually permit only the expression of categorical statements. Nearly all programming languages contain conditional statements — such conditional statements, however, are used to express control-flow rather than relations between classes. Compound statements (including conditionals) in traditional logic express relationships between classes. We will return to compound statements after examining the categorical statements of which the compound statements are composed.

the division of the categorical statement according to

quality { *affirmative*
negative

In traditional logic, the quality of a statement is what is most essential. A categorical statement is either *affirmative* or *negative* on the basis of whether the predicate is affirmed of the subject or denied of the subject. This primary division applies to the statement or proposition as a whole and is not to be confused with the statement having a negative subject or a negative predicate. The statement 'every non-animal is a non-horse' is an affirmative statement.

Most current object-oriented programming languages do not contain an explicit verb copula; statements are implicitly affirmative. The 'ShouldNotImplement' mechanism of Smalltalk is a notable exception to this. Exceptions (in the programming language sense) are used also to substitute for a negative verb copula, although it is not always clear when they are being used in this way because of their relation to program execution. Vlissides gives an example of exceptions (again in the programming language sense) being used to express the categorical negative proposition *file nodes cannot contain other nodes* [33]. His example models a hierarchical file system, which can be expressed by the following categorical statements: files are nodes; directories are nodes; a node may contain other nodes; file nodes are not nodes that can contain other nodes. He discusses the trade-offs between replacing the last two statements with *directory nodes may contain other nodes*, or keeping them as stated.

the division of the categorical statement according to

quantity { *universal* every or none
particular some

The quantity of the categorical statement is determined by the quantity of its subject. The subject may be either of *universal* or *particular* quantity, depending on whether we are predicating something of the whole of the subject or of only part of the subject. A universal subject best uses a *grammatically singular* form of the verb copula. The universal, expressed in terms such as **every**, **any**, etc., is treated as a grammatical singular because one nature is being signified. In Aristotelian logic **every** is the preferred manner of expressing a universal affirmative statement because **all** can be used either as a universal or as a collection.

These concepts of *universal* and *particular* quantity need to be distinguished from the *universal* ($\forall$) and *existential* ($\exists$) quantifiers used in first order predicate calculus. Aristotelian logic employs universal terms, and first order predicate calculus employs singular terms, and this is the root of the distinction. It is very easy to confuse these two notions of quantity, however understanding the difference has led to improvements in static call graph construction algorithms for class-based languages.

One of the problems is that $\exists$ is now read as 'there exists'. This association was not made in the early days of first order predicate calculus, but begins to be found in the later writings of Bertrand Russell, and finds its fullest expression in the writings of Quine. The particular quantity in Aristotelian logic is expressed by 'some' and should not be read as 'there exists'. 'Some' may be read as 'there exists' only when speaking of immaterial beings (e.g. mathematical ones, such as numbers). For example, 'some numbers are greater than ten' is equivalent to 'there exists a number greater than ten'. However, 'some people have red hair' is not equivalent to 'there exists a person with red hair' — now we are speaking with universal terms and of material beings. When we say 'some people have red hair' we are speaking of part of the nature rather than making an 'ontological commitment' to the existence of such a person. Orenstein gives an excellent discussion of these issues in his book *Existence and the Particular Quantifier* [26], and Rose [27] also defends Aristotle against accusations of inconsistency leveled by Boole with respect to the notion of *existential import*.[8]

Understanding that this notion of quantity speaks of a nature (expressed in a universal) rather than of the existence of singulars is root of the distinction between *Rapid Type Analysis* [7, 6] and *Class Hierarchy Analysis* [14, 15]: *Rapid Type Analysis* is an improvement over *Class Hierarchy Analysis* because it removes uninstantiated classes from the call graph. Speaking of a nature does not entail the existence of material singulars; defining a class does not entail that it will be instantiated.

the division of the categorical statement according to

matter { *what is necessary*
what is impossible
what is neither necessary nor impossible

[8]Incidentally, Boole's criticism is the reason that first order predicate calculus considers only contradictory opposition.

The final division of the categorical proposition, the one the lies at the very core of the determination of truth or falsity for any statement, is the one that completely separates traditional logic from symbolic logic. This is the division of the statement *according to its matter*, the content rather than the form (although the form must be appropriate to the matter).

A statement has *necessary matter* when the predicate inheres in the very nature of the subject *per se*. This means according to what the subject is or what necessarily follows upon the kind of being which it is: for example, 'every human is rational' or 'every human is capable of artistic expression'. A universal affirmative statement is *true* only when the matter is necessary.

A statement has *impossible matter* when the predicate can never be said of the subject. Traditional logicians describe impossible matter as what is *per se repugnans*, or inimical, to the very nature of the subject. An example of impossible matter would be *no animal is a stone*. A universal negative statement is true only when the matter is impossible.

When the matter is *neither necessary nor impossible* it is said of a subject only contingently (or accidentally) — it may or may not be found in a given subject: for example, 'some people have red hair'. When the matter is contingent both the particular affirmative and the particular negative can be true simultaneously. Without a grasp of the predicable relations from the first operation of reason we would not know what should be the appropriate quantity of a statement.

3.2 Supposition

Not only is the second operation concerned with the properties of statements, but with how the subject stands in relation to each predicate. This kind of distinction is found in programming languages such as Smalltalk, which has both class and instance level attributes. Mediæval logicians referred to this facet of statements as *modes of supposition*, or what is 'supposed' by the subject in relation to each predicate. Supposition becomes a serious concern when statements are used in an argument, because terms in an argument must not only keep the same *signification*, but also the same *supposition*.

Real Supposition In this mode of supposing, the subject in relation to what is predicated of it, stands for the nature of the species and for all of the members of the species (i.e. for the intension and the extension). If we say that 'man is rational', the term 'man' in this context, stands both for human nature and for all of those the individuals in which that nature is found.

The term *personal* supposition was used in mediæval times to signify that what was supposed was not simply the species, but all of the *personae* (or individuals) of that species.

Logical Supposition In this mode of supposing, the subject, in relation to what is predicated of it stands *for the nature of the subject alone*. It *does not* stand for any of the individuals of the species but only for the concept; it does not refer to real beings (i.e. for the intension only). Consequently, what is predicated is simply a logical relation. 'Man is a species of animal', for example, cannot be said of individual men because what is supposed by 'man' in this case, is only the logically universal nature. We cannot change supposition within an argument, even if the premises are true, as in the following illustration:

> Man is a species of animal.
> Socrates is a man.
> Therefore, Socrates is a species of animal.

Fallacious arguments using both logical and real supposition might say that since 1 out 3 people will die of heart disease, then heart disease will the be the cause of death for 2 people in a family of six.

Obviously logical supposition can only be expressed using a grammatical singular, since here we are speaking *only* of the nature and not of the individuals of that nature.

The notion of class level attributes in languages such as Smalltalk is very similar to the logical mode of supposition.

Material Supposition In this mode of supposing, the subject, in relation to what is being predicated of it, stands only for the composition of the word itself rather than for either the logical or the real nature which the word can signify. Again, changing what is being supposed in the use of the terms within an argument may produce humorous fallacies. The following goes from *material* to *real* supposition:

> 'Man' is a three-lettered word.
> Socrates is a man.
> Therefore, Socrates is a three-lettered word.

Using the phrase *incremental modification* to describe the relation between two classes can lead to fallacy caused by switching modes of supposition. Consider the mathematical formalism for the relation between subclass and superclass ('inheritance') given by Wegner and Zdonik [35] as $R = M + O$ (result = modification + original). If we take 'man' as O and 'wo' as M we get the result R = 'woman'. This becomes fallacious only if we switch from the material mode of supposition to the real mode of supposition.

4. THE THIRD ACT: INFERENCE

It is sometimes said that object-oriented programming is 'more natural' to our reason. If this is the case, we posit that it is because of its similarity to Aristotelian logic. Aristotle initiated the formal study of logic by examining inferences similar to our 'transference deduction'. These inferences, inspired by Plato's method of division, were called 'perfect' by Aristotle. Their perfection has been model ever since of how our reason should proceed in deduction.

4.1 Formal Inference — Categorical Syllogism

The first two millennia in formal logic were concerned with the *syllogism*, a deductive formalism for the act of inference or reasoning, which proceeds from what is more universal to what is less universal — a composite expression in which *one thing being given* another thing necessarily follows. Just as we saw that there were two types of statements (categorical and compound), so too there are two types of syllogism, the categorical and the compound. Our focus in what is to follow will be on the formalism of the categorical syllogism.

The inference in deductive reasoning is *valid* if the *consequent* (conclusion) follows with real dependence upon the *antecedent* (premises), or *invalid* if the antecedent is not the cause of the consequent. The premises are designated as the *major premise* (which is 'major' because it contains the predicate of the conclusion), and the *minor premise* (so called because it contains the subject of the conclusion). The conclusion brings together its subject and its predicate from the antecedent premises because of a *middle term* which appears in both premises. The middle term is the key to every argument because it is the basis, or cause, of the inference. We can portray how this inference works in various notations; at the beginning of this paper we phrased the 'transference deduction' like so:

A is said of B	major premise
B is said of C	minor premise
$\therefore$ A is said of C	conclusion

A is the major extreme; B is the middle term; and C is the minor extreme. This may also be expressed as following:

$$\frac{AB \quad BC}{AC}$$

For the remainder of this discussion we will use the following phrasing (where S, M, and P stand for the subject of the conclusion, the middle term, and the predicate of the conclusion, respectively):

Every M is P	major premise
Every S is M	minor premise
$\therefore$ Every S is P	conclusion

Which may also be expressed (without indicating the quality or quantity of the statements) like so:

$$\begin{array}{ccc} & M & P \\ S & M & \\ \hline S & & P \end{array}$$

4.2 The Matter and Form of the Syllogism

The *proximate matter* of the syllogism consists of the three statements: the major premise, the minor premise, and the conclusion. The *remote matter* of the syllogism is the terms of which the statements are composed: the subject of the conclusion, the predicate of the conclusion, and the middle term. Each of these terms appears twice and only twice in the syllogism.

The *figure* of the categorical syllogism is determined by the position of the middle term. This is what determines the three possible figures for syllogistic inference, illustrated below (without indicating the quality or quantity of the statements):

First

$$\begin{array}{ccc} & M & P \\ S & M & \\ \hline S & & P \end{array}$$

Second

$$\begin{array}{ccc} & P & M \\ S & & M \\ \hline S & P & \end{array}$$

Third

$$\begin{array}{ccc} M & & P \\ M & S & \\ \hline & S & P \end{array}$$

4.3 The Perfection of the First Figure

Aristotle notes that a perfect syllogism needs nothing other than what has been stated in order to make evident the necessary consequence [*Prior Analytics* 24b22–23]. Rose provides a detailed account of the perfection of the first figure in [27].

It is the first figure which makes a conclusion most evident as a necessary consequence. We noted above that the three figures of the syllogism are determined by the three possible positions of the middle term that can provide valid syllogistic inference. If we look again at figure one:

$$\begin{array}{ccc} & M & P \\ S & M & \\ \hline S & & P \end{array}$$

we see that only in figure one does the middle term satisfy both notions of 'middle'. First of all, just as in figures two and three, so in figure one, the middle term is 'middle' by being related in the premises to both the subject and the predicate of the conclusion. But the unique perfection of figure one is that it is 'middle in universality' for, as the diagram shows, it is greater in universality than the subject of the conclusion (in the minor premise), and it is lesser in universality than the predicate of the conclusion (in the major premise). As a result it is perfectly evident that the predicate must be said of the subject in the conclusion. And because of the perfection of the middle term, it is the only figure that can produce a conclusion that is universal and affirmative. It is also the only figure that can validly infer all four species of statements: universal affirmative, universal negative, particular affirmative, and particular negative.

Second

$$\begin{array}{ccc} & P & M \\ S & & M \\ \hline S & P & \end{array} \qquad \begin{array}{c} \text{major premise} \\ \text{minor premise} \\ \hline \text{conclusion} \end{array} \qquad \begin{array}{ccc} M & & P \\ M & S & \\ \hline & S & P \end{array}$$

Third

The identifying characteristic of figure two is that the middle term occupies the predicate's place in both the major and minor premise. Because of this, it participates only partly in the perfection of the first figure, for while the middle term is greater in universality than is the subject of the conclusion (like figure one), it is also greater in universality than is the predicate of the conclusion and so it is not middle in universality. Nevertheless, because it satisfies partially the notion of 'middle' with respect to universality, it can produce a valid conclusion — but only negative conclusions.

The third figure, on the other hand, has as its characteristic that the middle term occupies the subject's place in both the major and the minor premises. It too participates only partially in the perfection of the first figure, for here, like figure one, the middle term is less universal than is the predicate of the conclusion — but the middle term is less universal also than the subject of the conclusion. Because it is less universal than both the subject and the predicate of the conclusion, it produces the least evident conclusion, even when it is valid. And since the subject does not 'drop down' (so to speak) as in figures one and two, but must 'cross over' from the predicate's place in the minor premise, the only possible valid inferences produced are statements that are particular in quantity.

5. CONCLUSION

At the outset of our enquiry we posed the problem of the *complementary* roles of meaning and formalism in conceptual integrity. We illustrated this with four examples of the 'transference deduction'. Let us now address again the illustrations with which we began:

Every animal is mortal.
1. Man is an animal.
∴ Man is mortal.

Man is mortal.
2. Socrates is a man.
∴ Socrates is mortal.

In these two illustrations, the predicate is said essentially of the subject in the minor premise, and the relation of predicate to the subject in the major premise is the predicable relation of property. Hence we have statements that are true. Further, the middle term in each inference is middle not only by being attached to the major and minor premises but, more importantly, by being a middle in universality. For this reason the 'inevitable' inference is very evidently both valid and true. By contrast, in the second pair of illustrations,

Homo erectus stands upright.
3. *Homo sapiens* evolved from *homo erectus.*
∴ *Homo sapiens* stands upright.

Key A opens this door.
4. Key B is a copy of Key A.
∴ Key B opens this door.

the resemblance to the first pair of inferences is only accidental. And this resemblance is accidental in two distinct ways. First of all, the middle term is only accidentally related to both the predicate and the subject in the major and the minor premises of both arguments. By this we mean that there is a relation only of predicable accident and, in the minor premises, these are not sufficient differences to adequately define the subject. Secondly, and even more importantly, while the 'middle term' is attached to both the subject and the predicate of the conclusion in the major and minor premises, in both cases there is no dimension of 'middle in universality', which is the real basis for inference and for validity in reasoning.

Conceptual integrity depends on rigorous formalism, but such formalisms are advantagous only when real logical relation is carefully discerned. It has been our contention that Aristotelian logic will be of benefit to programmers in performing their art *with order, with ease, and without error* [2].

5.1 Future Work

We see potential future work in both theoretical and practical dimensions. On the theoretical side, both the Beta conceptual framework and the word 'is' are worthy of further philosophical investigation (and these directions are not unrelated). We would also like to see the impact of the use of metaphor in programming investigated. On the practical side, we think that there are many good ideas discussed in this paper that may be of benefit for programming language design. One area that we have not had space to explore in this paper is the formalism developed for statements and syllogisms in the Middle Ages: specifically, determinations of the truth of statements and the reductions of the other figures to the first figure. We also have some ideas for designing a programming environment inspired by the form of the syllogism.

Acknowledgements

First we would like to thank those who have patiently reviewed earlier drafts of this paper: Paul Asman, Scott Kerr, Thomas Kuehne, Kim Mens, Tom Mens, Marcellus Mindel, Amedeo Napoli, Gary Rathwell, Tarver Swejkowski, Kresten Krab Thorup, and the anonymous referees. Paul Asman and Marcellus Mindel have been particularly generous with their time. We have also received some helpful comments from Samantha Gross and Ole Lehrmann Madsen.

We would also like to acknowledge those who have encouraged us, as this work might not have been written otherwise: Karel Driesen, Mohammed Kamel, Kostas Kontogiannis, Peter Sweeney, Dave Thomson, and Frank Tip.

Finally, this work was supported in part by the IBM Centre for Advanced Studies and the National Research Council of Canada.

6. REFERENCES

[1] Anquetil, N., and Vaucher, J. Expressing opposition in the object model: Simple as NOT. *ACM SIGPLAN Notices 33*, 1 (January 1998).

[2] Aquinas. *In Libros Posteriorum Analyticorum Expositio (Commentary on the Posterior Analytics of Aristotle).* Marietti, 1964. Original text composed circa 1265 AD. Translation ours (where appropriate).

[3] Aristotle. Categories. In Ross [28]. Translated by E.M. Edghill. Also available from The Internet Classics Archive (http://classics.mit.edu).

[4] Aristotle. On Interpretation. In Ross [28]. Translated by E.M. Edghill. Also available from The Internet Classics Archive (http://classics.mit.edu).

[5] Aristotle. Prior Analytics. In Ross [28].

[6] Bacon, D. F. *Fast and Effective Optimization of Statically Typed Object-Oriented Languages.* PhD thesis, University of California at Berkeley, December 1997. UCB/CSD-98-1017.

[7] Bacon, D. F., and Sweeney, P. F. Fast static analysis of C++ virtual function calls. In Coplien [13], pp. 324 – 341.

[8] Beck, K. *Extreme Programming Explained.* Addison Wesley, 1999.

[9] Brachman, R. J. What is-a Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *IEEE Computer 16*, 10 (October 1983), pp. 30 – 36.

[10] Brooks Jr, F. P. No silver bullet: Essence and accidents of software engineering. *Computer 20*, 4 (April 1987), pp. 10–19. Reprinted from *Proc. IFIP Congress*, Dublin, Ireland, 1986.

[11] Brooks Jr, F. P. *The Mythical Man-Month.* Addison Wesley, 1995. Twentieth anniversary edition.

[12] Bryant, A. 'It's Engineering Jim ... but not as we know it': Software Engineering — solution to the software crisis, or part of the problem? In *ICSE'00* (Limerick, Ireland, June 2000), M. Jazayeri and A. Wolf, Eds., pp. pp. 78 – 87.

[13] Coplien, J., Ed. *Proceedings of Object-Oriented Systems, Languages and Applications (OOPSLA)* (San Jose, California, October 1996).

[14] Dean, J., Grove, D., and Chambers, C. Optimization of object-oriented programs using static class hierarchy analysis. In *ECOOP'95* (Århus, Denmark, August 1995), W. Olthoff, Ed., Springer-Verlag. LNCS 952.

[15] Diwan, A., Moss, J. E. B., and McKinley, K. S. Simple and effective analysis of statically-typed object-oriented programs. In Coplien [13], pp. 292–305.

[16] Gjessing, S., and Nygaard, K., Eds. *Proceedings of European Conference on Object-Oriented Programming* (Oslo, Norway, August 1988), Springer-Verlag. LNCS 322.

[17] Grogono, P., and Sakkinen, M. Copying and comparing: Problems and solutions. In *ECOOP'00*, E. Bertino, Ed. Springer-Verlag, Cannes, France, June 2000.

[18] Grosberg, J. Comment on considering 'class' harmful. *CACM 36*, 1 (January 1993), pp. 113 – 114. Technical correspondence.

[19] Hürsch, W. L. Should superclasses be abstract? In *ECOOP'94* (Bologna, Italy, July 1994), M. Tokoro and R. Pareschi, Eds., Springer-Verlag, pp. 12 – 31. LNCS 821.

[20] LaLonde, W. R., and Pugh, J. Subclassing $\neq$ subtyping $\neq$ is-a. *Journal of Object-Oriented Programming 3*, 5 (January 1991).

[21] Łukasiewicz, J. *Aristotle's Syllogistic From the Standpoint of Modern Formal Logic*, 2^{nd} ed. Oxford Clarendon Press, 1957. A study of Aristotle's *Prior Analytics* from the Greek text. This study presents the first modern mathematical formalism developed for the syllogism.

[22] MacGregor, R. The Loom Web-site. Contains a list of the numerous papers written on Loom. http://www.isi.edu/isd/LOOM.

[23] Madsen, O. L., Møller-Pedersen, B., and Nygaard, K. *Object-oriented Programming in the* Beta *Programming Language.* Addison Wesley, 1993.

[24] Mylopoulos, J. Classes and instances. *International Journal of Intelligent and Cooperative Systems 1*, 1 (April 1992).

[25] Mylopoulus, J., Borgida, A., Jarke, M., and Koubarakis, M. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems* (1990). There are many other papers on Telos in the literature.

[26] Orenstein, A. *Existence and the Particular Quantifier.* Temple University Press, Philadelphia, 1978.

[27] Rose, L. E. *Aristotle's Syllogistic.* Charles C. Thomas Publisher, 1968. A study of Aristotle's *Prior Analytics* from the Greek text.

[28] Ross, W. D., Ed. *The Works of Aristotle, Volume 1: Logic.* Oxford University Press, 1928.

[29] Ross, W. D. *Aristotle's Prior and Posterior Analytics.* Oxford University Press, 1957.

[30] Sakkinen, M. Disciplined inheritance. In *ECOOP'89* (Nottingham, England, July 1989), S. Cook, Ed., Cambridge University Press, pp. 39 – 56.

[31] Sowa, J. F. *Knowledge Representation: Logical, Philosophical, and Computational Foundations.* Brooks/Cole Thomson Learning, 2000.

[32] Taivalsaari, A. On the notion of inheritance. *ACM Computing Surveys 28*, 3 (September 1996), pp. 438 – 479.

[33] Vlissides, J. *Pattern Hatching: Design Patterns Applied.* Addison Wesley, 1998.

[34] Wegner, P. The object-oriented classification paradigm. In *Research Directions In Object-Oriented Programming*, B. Shriver and P. Wegner, Eds. MIT Press, Cambridge, Massachusetts, 1987, pp. 479 – 560.

[35] Wegner, P., and Zdonik, S. B. Inheritance as an Incremental Modification Mechanism — or — What Like Is and Isn't Like. In Gjessing and Nygaard [16], pp. 55 – 77.

[36] Winkler, J. F. H. Objectivism: 'class' considered harmful. *CACM 35*, 8 (August 1992), pp. 128 – 130. Technical correspondence.