The Rise of OOP: Part 1 [The Early Years]

Dina Lamdany

What are Today's Most Popular PLs?

tiobe.com	PyPL Index	RedMonk	StackOverflow
C	Java	Java/JavaScript	Java
Java	PHP	PHP	C#
Objective-C	Python	Python	JavaScript
C++	C#	C#	PHP
C#	C++	C++/Ruby	Python
Basic	C	CSS	C++
PHP	Javascript	C	SQL
Python	Objective-C	Objective-C	Objective-C
JavaScript	Ruby		C
Transact-SQL	Basic		Ruby
[www.tiobe.com,	[PyPL Index, August 2014 Tutorial searches on Google]	[redmonk.com, June 2014 Data from GitHub]	[langpop.corger.nl, August 2014 Data from GitHub]

The Java Tutorials

“If you've never used an object-oriented programming language before, you'll need to learn a few basic concepts before you can begin writing any code.

This lesson will introduce you to **objects, classes, inheritance, interfaces, and packages**.

Each discussion focuses on how these concepts relate **to the real world,** while simultaneously providing an introduction to the syntax of the Java programming language.”

How did we get here?

C.A.R Hoare, “Record Handling,” 1966

“The most valuable feature of a programming language is that it provides the programmer with a **conceptual framework which enables him to think more clearly about his problems and about effective methods for their solution, and a notational technique which enables him to express his thoughts clearly.**”

“A fundamental feature of **our understanding of the world is that we organize our experience as a number of distinct objects** (tables and chairs, bank loans and algebraic expressions, polynomials and persons, transistors and triangles, etc.); and our thought language, and actions are based on the designation, description, and manipulation of these objects, either individually or in relationship with other objects. When we wish to solve a problem on a computer, **we often need to construct within the computer a model of that aspect of the real or conceptual world to which the solution of the problem will be applied.**”

Sketchpad (~1963) Ivan Edward Sutherland, Dissertation Thesis

“Sketchpad has shown the **most usefulness as an aid to the understanding of processes,** such as the notion of linkages, which can be described with pictures.”

“The sketchpad system, by eliminating typed statements (except for legends) in favor of line drawings, opens up a new area of man-machine communication...The **knowledge of the facilities which would prove useful could only be obtained by actually drawing them.**

Examples?

“A **subpicture** capability for including arbitrary symbols on a drawing, a **constraint capability for relating the parts of a drawing in any computable way,** and a **definition copying capability for building complex relationships from combinations of simple atomic constraints.”**

SIMULA I (~1965)

“The language itself must be such that users were invited to make efficient programs”

“The writing of the SIMULA program was almost always useful, since the development of this program (regarded as a system description) resulted in a better understanding of the system.”

“The Development of the Simula Languages” by Kristen Nygaard & Ole-Johan Dahl

SIMULA I → SIMULA 67 (~1967)

"We had observed that **processes often shared a number of common properties, both in data attributes and actions**, but were structurally different in other respects so that they had to be described by separate declarations."

"We needed **subclasses of processes with own actions and local data stacks, not only of pure data records.**..group together common process properties in such a way that they could be applied later, in a variety of different situations not necessarily known in advance...t**he subclass idea of Hoare was a natural starting point"**

"The basic concept should be classes of objects"

Why all of this philosophizing?

“Software Engineering” Conference by NATO Science Committee in Garmisch, Germany, October 1968

In Europe alone had 10,000 computers, growing 25 – 50% per year

OS/360 cost IBM over $50 million dollars a year, 5000 man hours

“In 1958, a European general purpose computer manufacturer often had less than 50 software programmers, now they probably number 1000 – 2000 people; what will be needed in 1978?”

“The basic problem is that certain classes of systems are placing demands on us which are **beyond our capabilities and our theories and methods of design and production** at this time. There are many areas where there is no such thing as a crisis—sort routines, payroll applications, for example. It is large systems that are encountering great difficulties.”

“The research, development, and production phases are often telescoped into one process...In the cold light of day, we know that a **step-by-step approach separating research and development from production is less risky and more likely to be successful**.”

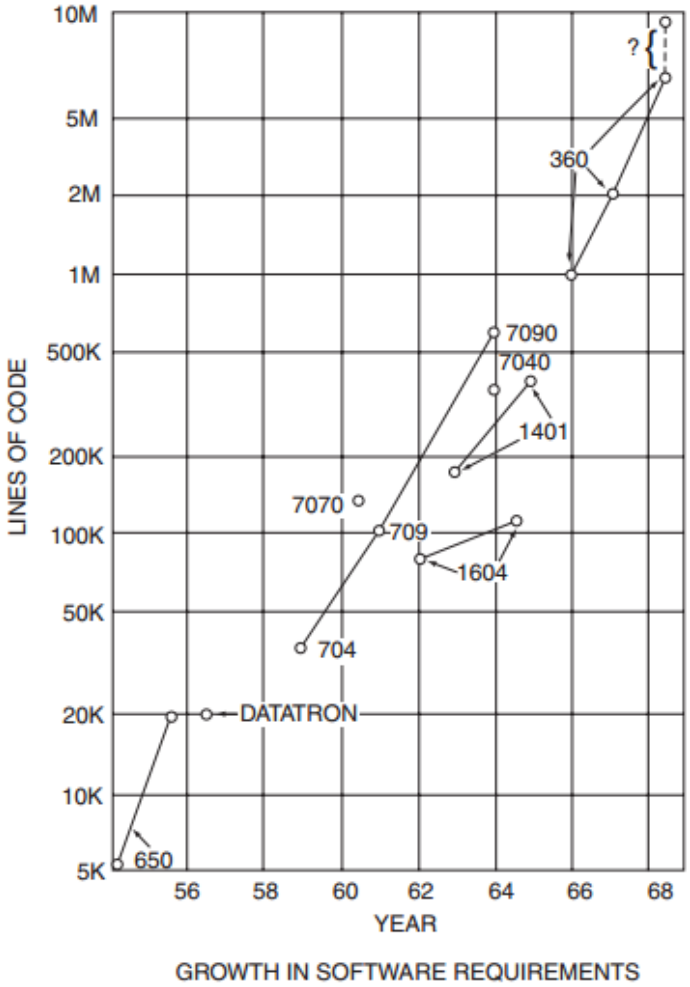

Figure 6. Provided by McClure

Back to philosophy...

Smalltalk, Alan Kay, ~1972

“Though it has noble ancestors indeed, Smalltalk’s contribution is a new design paradigm–which I called object-oriented–for attacking large problems of the professional programmer, and making small ones possible for the novice user. **Object-oriented design is a successful attempt to qualitatively improve the efficiency of modeling the ever more complex dynamic systems and user relationships made possible by the silicon explosion**.”

Alan Kay, “The Early History of Smalltalk”

1. Everything is an object
2. Objects communicate by sending and receiving *messages* (in terms of objects)
3. Objects have their *own memory* (in terms of objects)
4. Every object is an instance of a *class* (which must be an object)
5. The class holds the shared *behavior* for its instances (in the form of objects in a program list

“**What Sketchpad called masters and instances, Simula called activities and processes. Moreover, Simula was a procedural language for controlling Sketchpad-like objects,** thus having considerably more flexibility than constraints (though at some cost in elegance) This was the big hit, and I’ ve not been the same since.

I think the reason the hit had such impact was that I had seen the idea enough times in enough different forms that the final recognition was in such general terms to have the quality of an epiphany. My math major had centered on abstract algebras with their few operations generally applying to many structures. My biology manor had focused on both cell metabolism and larger scale morphogenesis with its notions of simple mechanisms controlling complex processes and one kind of building block able to differentiate into all needed building blocks. **The 220 file system, the B5000, Sketchpad, and finally Simula, all used the same idea for different purposes. Bob Barton, the main designer of the B5000 and a professor at Utah had said in one of his talks a few days earlier: “The basic principle of recursive design is to make the parts have the same power as the whole.”**

...1979, Bjarne Stroustroup starts working on “C with Classes”

A taste of what is to come...

“Object oriented to most people meant ‘good’...you will notice there is no mention of object-oriented programming in the first book”
-Bjarne Stroustroup, Interview

“Its main purpose was to make writing good programs easier and more pleasant for the individual programmer. There never was a C++ paper design; design, documentation, and implementation went on simultaneously”
-The C++ Programming Language, Bjarne Stroustrup, 1986

Some things to look forward to

C/C++ and how they incorporate these new ideas

Java and its rise to dominance

Resources

SOFTWARE ENGINEERING: Report on a conference sponsored by the

NATO SCIENCE COMMITTEE (Garmisch, Germany, 7th to 11th October 1968)

http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

Sketchpad: A man-machine graphical communication system. http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-574.pdf

“The Development of the Simula Languages” by Kristen Nygaard, Ole-Johan Dahl http://phobos.ramapo.edu/~ldant/datascope/simula%20history.pdf

Great book on the history of computing, OOP, etc: “Histories of Computing” by Michael Mahoney

A joke, courtesy of Alan Kay:

“Smalltalk’s design–and existence–is due to the insight that everything we can describe can be represented by the recursive composition of a single kind of behavioral building block that hides its combination of state and process inside itself and can be dealt with only through the exchange of messages. Philosophically, Smalltalk’s objects have much in common with the monads of Leibniz and the notions of 20th century physics and biology. Its way of making objects is quite Platonic in that some of them act as idealisations of concepts–Ideas–from which manifestations can be created. That the Ideas are themselves manifestations (of the Idea-Idea) and that the Idea-Idea is a-kind-of Manifestation-Idea–which is a-kind-of itself, so that the system is completely self-describing– would have been appreciated by Plato as an extremely practical joke [Plato].”