EDITOR **DAVID ALAN GRIER**
Djaghe; LLC; grier@gwu.edu

# Objects in the Mirror Are Closer Than They Appear

**David Alan Grier,** Djaghe, LLC

*Ideas emerge into the field of computer science in different ways. Sometimes they are carried forward by a single prominent publication. However, some important ideas, such as object-oriented programming, are promoted by a stream of articles.*

Despite any claims to the contrary, this column is neither a lesson in history nor a promotion of a "great article" theory of computer science. History, especially technical history, is problematic and is best left to professionals. Too often, it is used to justify a current project or give validity to a plan for future research. Our ideas need to stand for themselves. The fact that we can connect them to a famous result or prominent computer scientist does not make them any more valid.

The idea that there are "great articles" in our field and that we can gain something by reading them is a more intriguing theory. There certainly are great articles. This column has highlighted many of the best contributions that have appeared in the pages of *Computer*. Many remain important pieces of literature. There is much to be learned on the subject of software engineering by reading Barry Boehm's seminal article, the spiral model of software engineering, or Jain, Mao, and Mohiuddin's treatment of neural networks. (See "Body

## ARTICLE FACTS

- Article: "Object-Oriented and Conventional Analysis and Design Methodologies"
- Authors: R.G. Fichman and C.F. Kemerer
- Citation: *Computer*, vol. 25, no. 10, pp. 22–39, October 1992
- *Computer* influence rank: #888 with 972 views and downloads and 91 citations

*Digital Object Identifier 10.1109/MC.2021.3055918*
*Date of current version: 22 October 2021*

of Knowledge" for April 2020 and June 2020, respectively.) However, not all influential articles are great, and many an important field in computer science lacks a single dominant publication.

Let us take the subject of object-oriented programming. It is clearly an important topic. Every student is asked to master the concepts of objects, inheritance, and messages very early in his or her career. We have a whole generation of developers who cannot imagine creating a software system with any other kind of language.

However, there is no prominent and influential article on the subject in the back issues of *Computer*. You have to dig fairly deep to find the work that we are considering in this essay. It is ranked number 888 on our list of influential articles. While that is a commendable showing, it does not suggest that "Object-Oriented and Conventional Analysis and Design Methodologies"[1] was a document that shaped the thinking and practices of IEEE Computer Society members (see "Article Facts"). However, if we consider it in context, we quickly find that it shows us how computer scientists embraced this new programming paradigm and the lessons that they had to learn.

The roots of object-oriented programming go back to the 1960s and 1970s, with the language Simula (1967) and the language Smalltalk (1972). However, the approach did not become widely used until the 1980s. At universities, the language Common Lisp (1984) encouraged academics to cast their ideas in an object-oriented framework. In industry, C++ (1985) gave the same impetus to commercial software.

However, the existence of object-oriented languages was not enough to guarantee that object-oriented programming would be used. The history of programming is littered with examples that promised to change the nature of coding but, ultimately, fretted and strutted their brief moment on the stage before they vanished and were forgotten.

The success of object-oriented programming never depended upon a single language any more than it relied on a single great article. If we look at *Computer* between 1990 and 2000, we see a steady stream of articles on object-oriented programming. After publishing 16 such articles in the prior decade, the magazine published 10–12 articles a year on the subject. For two years, 1996 and 1997, it published 24 articles a year, roughly a quarter of its output.

If you look at these articles from the 1990s, you find that many of them attempt to fit a specific application within an object-oriented framework: expert systems and graphical user interfaces (1990); databases and distributed systems (1991); 3D graphics, audio processing, and real-time programming (1992); and distributed processing, parallel processing, computer-oriented engineering (1993). The list of topics continues to grow for the remainder of the decade.

In this body of literature, our current article ("Object-Oriented and Conventional Analysis and Design Methodologies") asks the obvious but highly important question: what has changed with this new way of programming? "Object orientation certainly encompasses many novel concepts, and some have called it a new paradigm for software development," noted the authors. "Yet, the question of whether object-oriented methodologies represent a radical change over such conventional methodologies as structured analysis remains a subject of much debate."[3]

After a great deal of analysis, the authors conclude that the new form of programming represented a substantial change. "Object orientation is founded on a collection of powerful ideas," they noted, "that have firm theoretical foundations." At same time, they acknowledged that there was no commonly accepted method for creating object-oriented software. "None of the methodologies reviewed here," they concluded, has "achieved the status of a widely recognized standard."[3] Because of this, they accepted the idea that "a move to an object-oriented environment in general may be seen predominantly as a radical change."

This conclusion helps us understand the early body of knowledge on object-oriented programming and suggests why we do not find a single dominant article on the subject. In all, *Computer* published 134 articles on object-oriented programming between 1990 and 2000. Most of these articles treated specific applications and the methods for developing them. They continued and expanded the work of authors Fichman and Kemerer. These articles considered specific development techniques and looked at the ways that object-oriented methods changed them.

There was no single great article because there was no easy way to combine all of those specific techniques into a single narrative. The "great article" was not one but many. It was the collection of the 134 *Computer* articles that marked the transition from conventional programming to object-oriented languages.

This returns us to the fundamental approach of this column. It is not interested in history nor does it promote the idea of a literature of great articles. It is concerned with how new ideas emerge in our field. Currently, there are three or four major topics coming to the forefront of our discipline. Among this list, I would include quantum computing, machine learning, and blockchain. It would be useful to understand how these ideas might become commonplace and affect ordinary practitioners of our field.

Of the three, machine learning is the most mature. It has also produced a body of literature that closely resembles that for object-oriented programming. Over the past decade and a half, researchers have published more than 100 articles in the field, including the one that was featured in this column. Step by step, these works show how this technology builds on the various subfields of computer science, such as cybersecurity, computer vision, data mining, and modeling. This body of literature has produced a cohort of practitioners who use it as their sole means of engaging with computing technology. They may do a small amount of conventional coding, but they are primarily engaged in the work

of designing a neural system, collecting data, training their system, and evaluating their work.

Quantum computing has not advanced as far into our community as machine learning. It has produced literature in *Computer*. This magazine has published more than 60 articles on the subject, though many are reports on the progress of the technology. Of all of the topics that are current in *Computer*, it could benefit from a single prominent article, one that focused on how to use quantum to solve well-understood problems. We hope for such an article, although one many not appear until quantum environments become much more common.

Blockchain is the most interesting case of the three. It has produced a major prominent article, though not one published in *Computer*. It has also led to a stream of publications that illustrates how it is connected to different fields in computer science and might solve certain problems. However, for the moment, it remains a minor technology, one that has yet to have a major impact on the field of computer science.

This is part of the process of how ideas emerge in our field. Sometimes, they are pushed forward by a major prominent article that is read by many members of the IEEE Computer Society. However, others, as was the case for object-oriented programming, are carried forward by a multiplicity of articles, which are written by many members.


## ACKNOWLEDGMENT

For these 2021 columns, "Body of Knowledge" takes its information from a report prepared by the IEEE Publications office on 20 November 2020, and the statistics were current as of that date. Other citation services can and do provide different numbers.


## REFERENCE


1. R. G. Fichman and C. F. Kemerer, "Object-oriented and conventional analysis and design methodologies," *Computer*, vol. 25, no. 10, pp. 22–39, 1992. doi: 10.1109/2.161278.



**DAVID ALAN GRIER** is a principal with Djaghe, LLC, Washington, D.C., 20003, USA. He is a Fellow of IEEE. Contact him at grier@gwu.edu.





*Digital Object Identifier 10.1109/MC.2021.3115378*