This article appeared as a guest editorial in *JOOP* in 1991. Citation reference: Bertrand Meyer: *Object-Orientation Outlook 1991: Optimism, pessimism, challenges*, Guest Editorial, *Journal of Object-Oriented Programming* (JOOP), 1991.

Object-Orientation Outlook 1991:

Optimism, pessimism, challenges

An uninformed reader browsing through just about any computer magazine these days – not just JOOP – couldn't be blamed for inferring that the object-oriented revolution is over, having routed the opposition. On paper, object-orientedness is everywhere. No ad, no technology announcement is complete unless it claims that its product or project is object-oriented.

The reality, as everyone in the O-O world knows, is far behind the appearance. Fast as the growth may be, object-oriented technology has only skimmed the surface of the software industry. All the rest is hope and hype, with the "object-oriented" label on software products soon to become as meaningful as "all natural" on food products.

Is this good or bad? Both, of course. It would be hypocritical for a developer or researcher in the field to complain about all the current attention, especially in comparison with the years of preaching in the desert. Just being in the limelight makes it possible to undertake projects that would have been unthinkable before. Yet there is also a considerable danger: the dilution of O-O ideas. If everyone and his brother is object-oriented, how do we separate the wheat from the chaff? How do we present "real" O-O technology so that as to distinguish it, in the eyes of users, customers and government sponsors, from mere lip service?

A similar question presents itself to those members of the community who can't force themselves to take solutions based on "hybrids" such as C++ as the ideal in object-oriented technology (and view them in some ways, as a move back to the pre-structured days of address arithmetic and weak typing). But here also two outlooks are possible. An optimist will see the hybrid approach as attracting people who might otherwise have thought O-O technology too esoteric for them, but will then realize the limitations and move on to the real thing. A pessimist will fear that the approach's failure to produce the advertized benefits – after all, hybrid solutions are likely to yield hybrid quality – will cause a backlash for the whole field, giving us all a bad name. Who can say today whether the optimist or the pessimist is right?

The answer to these questions will depend for a large part on how well the object-oriented community can provide, in the months and years to come, what the industry is clamoring for: complete, extendible and (talk about buzzwords!) open solutions. The groundwork in O-O principles and tools has been done; but there remains to make our ideas applicable to the real goals of companies. A company's IS department is not in the business of programming; it is in the business of solving problems. Solving problems usually involves programming, of course, but the programs must fit within the rest.

To succeed, then, developers of O-O technology must address the big picture. Some of the problems are in more urgent need of a solution than others; here are a few. (Lest this appears like a summary of session titles for OOPSLA or TOOLS, you may also want to check all the fashionable topics that are *not* on this list.)

- Databases. Anyone will tell you these days that data are the lifeblood of an organization. How do we use object-oriented systems to improve access to and manipulation of these data? With all due respect to my colleagues in the O-O database field, it has not yet been proved, at least for "mainstream" applications, that object-oriented databases are the solution.
- User interfaces. It is not enough any more to provide fancy menus and tools for defining screen layouts; object-oriented technology should allow the interactive definition of entire applications from reusable components. Even more importantly, it should integrate the user interface with the data manipulation tools – hence the close connection between this item and the previous one.
- Libraries. We need better quality control methods for libraries; we need more input from formal methods of specification, validation and documentation; we need vertical libraries developed in collaboration between O-O specialists and experts of the problem domain.
- Concurrency. With again no disrespect intended to the already considerable body of valuable work in this field, no one has yet come up with the general model that will reconcile the O-O view of computation with the needs of operating systems, multiprocessing, real-time and distributed applications.

This is enough to keep any organization's plate full for some time. Well, if the solutions were here already, where would the fun be?