



The Emperor Has No Clothes: Naked Objects Meet the Interface

Larry Constantine
Director of Research and Development
Constantine & Lockwood, Ltd.

Abstract. Naked Objects, the latest incarnation of the persistent notion of object-oriented user interfaces, proposes to eliminate the need for visual and interaction design of user interfaces by always presenting users with unadorned domain objects in a standard form and by constraining all interaction to the same few interaction idioms. Such simplistic user interfaces can be generated automatically through a software framework. This article examines the likely impact of the Naked Objects approach in light of its strengths and shortcomings as well as its undeniable appeal to developers and decision makers seeking shortcuts to user interface design. The ultimate significance of Naked Objects may be in the lessons it offers for practicing professionals, lessons that highlight the need for empowering users as problem-solvers by giving them better tools that enable them to achieve diverse ends by diverse means.

Keywords: objects, object-oriented user interfaces, expressive systems, usability, user interface design, usage-centered design

Learn more about usage-centered design at <http://www.forUse.com>.

It may sound like the jocular name of a raunchy hard-rock band, but Naked Objects are a development to be taken very seriously by the usability and user interface design community. If you do not already know about Naked Objects, it is time to learn. The roll-out of Naked Objects (the book, Pawson and Mathews, 2002) and Naked Objects (the programming tool) took place at the recent OOPSLA Conference in Seattle. A quintessential geekfest that began as a narrowly thematic conference focused on object-oriented programming, OOPSLA has matured into a major annual gathering of the seriously hardcore among software engineers and developers.

I first challenged this crowd to recognize the importance of users back in 1996 with my conference keynote, "Objects as if People Mattered." At the latest OOPSLA, a token few sessions and tutorials dealing with users and user interfaces were scattered around the program, testifying that the challenge may have been taken up but that recognition grows ever so slowly.

For usability professionals, however, the most significant event was not in the regular program but in the demo track, where, in a curtained off area of the exhibit hall, Richard Pawson and

Robert Mathews of CSC's Research Services in the UK peddled their Naked Object approach to the problem of user interface design. Their solution for usability? Eliminate user interface design altogether.

Formerly referred to as “expressive systems,” in its bare essentials the Naked Objects premise is simplicity itself. Instead of designing, building, and testing a user interface for the next software system, all the developer needs to do is create software objects that correspond to and fully model the “business objects” that make up the application domain. The Naked Object “framework” then automatically generates an “object-oriented user interface” that simply presents these objects and their related operations (referred to as “methods” by the object-oriented cognoscenti) directly to the user without adornment or embellishment. The software objects are, thus, stark naked to the user.

In practice, this means that the user of the application—of any application based on Naked Objects—sees something like the example shown in Figure 1: a series of windows containing little icons representing classes of objects that the program “understands” or individual objects, called “instantiations” in the jargon of object-orientation. Objects can be “opened” by double-clicking to reveal their contents in a standard format. Users can do something with the system either by dragging-and-dropping one object onto another or by right-clicking on an object to pop up a context menu from which operations supported by the object can be selected. That's it.

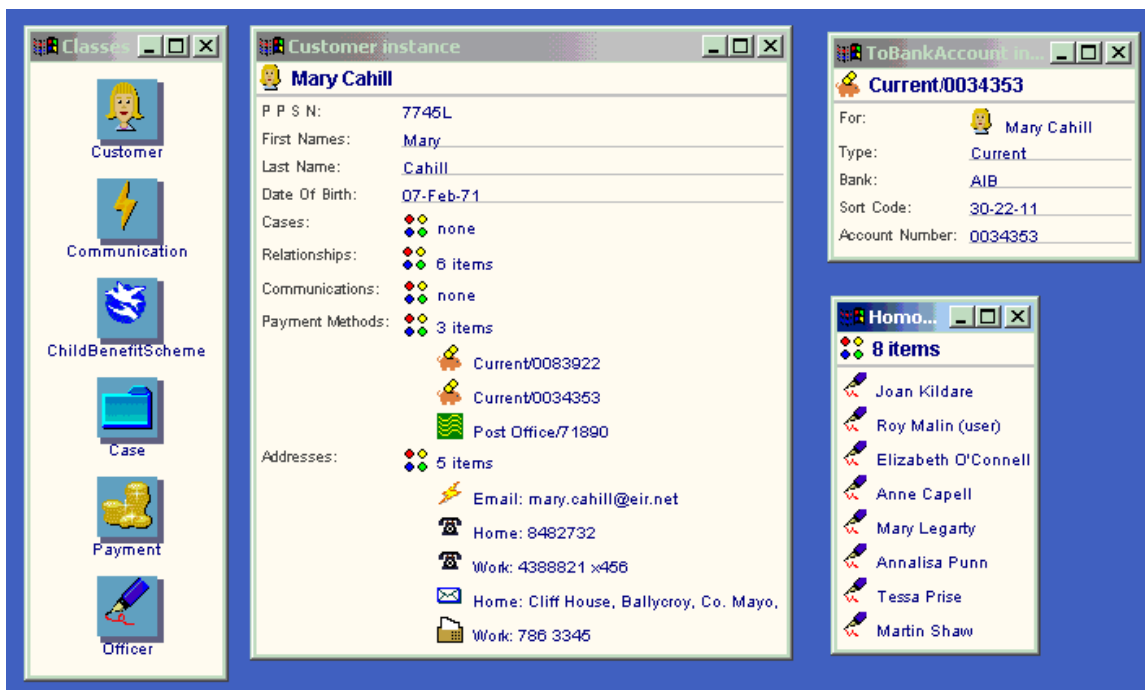


Figure 1 – Example of an expressive interface based on naked objects.
(Used with permission from www.nakedobjects.org.)

All Naked Object interfaces look essentially the same and work the same way. Create a new Customer (or Loan or Account or what have you) by right-clicking on the Customer (Loan, Account, or the like) class icon and selecting “New Instance” from the pop-up menu. To assign a Customer (Loan, Account,...) to a Sales Person (Loan Administrator, Account Manager,...) drag the Customer (Loan, Account,...) icon to the icon for the desired Sales Person (Loan Administrator, Account Manager,...). And so it goes.

The usability problems with such interfaces under most conditions of use are too numerous to go into in detail in this brief report and will be fairly obvious to any competent usability professional. Among these are the fact that context menus are a form of hidden behavior whose contents change and must be perused on every use in order to pick an action. Both drag-and-drop and right-clicking are interaction idioms that a significant fraction of users find difficult to master. With everything in separate windows, window management can become a burdensome overhead. Naked Objects also force users always to pick an object, then an action, even when it might be easier or more natural to pick an action and then the object on which to perform it.

The greatest usability problem with Naked Objects is the one-size-fits-all premise on which the approach rests. Instead of tailoring the presentation of information and the operation of the user interface to fit the unique aspects of the context, the application, and the user needs, one solution is presumed to fit all problems, provided all the relevant domain objects are properly identified with all their important behavior fully modeled.

Nevertheless, Pawson and Mathews muster an impressive panoply of arguments in support of the Naked Objects concept. Most importantly, the approach guarantees that the same business objects comprising the conceptual domain of users appear on the user interface and are coded into the software. These do not each have to be designed and built separately nor will they drift out of synch or be rendered incompatible through future changes or elaborations. The program objects are the interface objects, presented naked and unadorned to the user, and these are the same as the basic objects within which users and business analysts conceptualize the domain of the application. Neat, eh?

This simplistic approach to user interfaces could be dismissed as the laughable were it not for the powerful appeal it has to a couple of constituencies with real clout—far more clout than either users or usability professionals. Naked Objects appeal to managers and they appeal to developers. Looking around at the faces of those attending the demo, I could see the light of a new day dawning in the eyes of numerous instant converts. And their legions will grow.

Managers and business decision makers will embrace Naked Objects in part because the approach is cleverly cast in the right language and invokes the right images. Naked Objects are about business objects, about capturing business objects, constructing software around business objects, and presenting business objects to users. It is about a truly seamless development process that guarantees that features as delivered can be traced directly back to users and the application domain itself.

Of course, the real reason why managers will jump on the bandwagon is that Naked Objects allow them to invest nothing at all in usability or user interface design. The user interface—the “correct” user interface—comes free, generated automatically by a software framework which itself is available for free. With Naked Objects, there is no need to waste time on usability inspections or testing and no need to pay for or deal with those cranky and eccentric interaction designers. Just program the software objects, crank the code through some open-source software, and your user interface is finished.

In all fairness, Pawson and Mathews at times do temper their proposal by suggesting that skeptics may regard it as a prototyping approach that can, if absolutely necessary, be followed by the separate development of a conventional user interface. Nevertheless, their examples and asides make clear that they regard any substantial adornment of the Naked Objects as rarely necessary. Moreover, management wants to believe the claim that Naked Objects by themselves provide a workable user interface. They will, as I learned at OOPSLA, jump at the chance to cut usability from the budget and tune out any qualification to the claims.

Equally important, programmers and software engineers will passionately embrace Naked Objects. Most have never been completely comfortable with usability people, who they see as somewhat fuzzy-headed troublemakers who worry over minutiae and intangibles and delay projects by requiring rework. Programmers love Naked Objects. It proves to them that we are frauds, that all these years we have been flogging issues that are ultimately irrelevant. You do not have to design the user interface to fit the users or their work; users will do perfectly fine with a simple universal interface that just lets them solve their own problems.

In a twist of ironic ingenuity, the advocates for Naked Object argue that they are ultimately on the side of users, that they offer something better than what we have given users in the past. With a broad brush they paint a painfully unpleasant picture of conventional user interfaces as oppressive and disempowering. They construct a straw man in the form of so-called “optimized interfaces.” Although the term is not usual in the literature of user interface design, such interfaces are supposedly the kind most of us usability people devise. Optimized interfaces are rigid and inflexible, forcing users into a set sequence of operations and depriving them of opportunities for autonomous decision making and creative problem solving. Traditional user interface design, so it is claimed, does not respect users for their innate intelligence or problem-solving abilities. It treats them as cogs in the enterprise mechanism, much as they came to be regarded by the long discredited “scientific management” movement of the last century. Pawson and Mathews are at their most impassioned best when invoking the ghost of Frederick Winslow Taylor and painting a picture of the poor disenfranchised user in the modern “electronic sweatshop.”

Of course, we know there is a certain verisimilitude to this portrait. In some special cases, a rigid and uniform enforcement of a specific task structure may in fact be the ideal solution; even Pawson acknowledged this in a long discussion with me. Moreover, naïve designers with simplistic notions of task-centered design might often be guilty of sinning against the all-but-universally recognized principle of user control.

However, the stereotype of authoritarian interfaces hardly reflects modern mainstream notions of good visual and interaction design, and it certainly does not represent the usage-centered school of design (Constantine and Lockwood, 1999). Examples of “optimized” design that leave users firmly and flexibly in control abound. Successful usage-centered designs, such as the award-winning Siemens STEP 7 Lite system (<http://www.foruse.com/pcd/>) and the iMind Integrator (Constantine and Lockwood, 2002), persuasively demonstrate that a user interface can be at once custom-tailored to the task structure and the mental maps of users and simultaneously support flexible problem solving.

To the extent that Pawson and Mathews are criticizing unnecessarily rigid designs that impose on users, without justification, an arbitrary sequence or restricted programmatic logic, they are on the right track, as they are in some other areas. The Naked Objects approach hinges on intensive involvement of users in the collaborative development of an accurate and complete model of the domain objects and their operations. I have no doubt from my conversations with Pawson that this joint modeling is carried out with care and genuine respect for users and their capacities to contribute as collaborators in a problem-solving process. I am quite sure it leaves their users feeling heard and validated, particularly by contrast to all-too-common practices in commercial software development that treat users as irritants or as ignorant incompetents..

Equally important, the Naked Object approach virtually guarantees delivery of the system as modeled. Originating as it does within the object-oriented programming paradigm, the modeling process has sufficient rigor to be used as the definitive framework for constructing the needed software artifacts, thus ensuring that the delivered software matches the user needs, at least with respect to capturing and representing the content of the application domain as viewed by users. The fact that Pawson and Mathews require users to learn their language—the jargon of objects, instances, classes, and associations—to participate in the collaboration

highlights the limits of their willingness to reach out to users but leaves them in good company with the many other methodologists who subject users to obscure notations or arcane terminology.

If it stopped there, with collaborative modeling of the application domain, Naked Objects would deserve to be hailed as a broadly useful perspective deserving of wide practice. Of course, it does not stop there, but rather delivers degenerate user interfaces that force users not only to learn the jargon but to submit to the discipline of a limited and decidedly geekish vocabulary of interaction. Pawson and Mathews, who by their own admission are neither usability experts nor well-versed in user interface design, seem to be blissfully ignorant of the problems that the drag-and-drop and right-click interaction idioms pose for many users, for example. They are also completely enamored of an object-centered worldview and are fully prepared to enforce a rigid object-action grammar for user interaction, failing to realize that selecting an object before an action or an action before an object is a matter of preference that varies among users and across circumstances. Their book cites sources arguing for “object-oriented user interfaces” but ignores the critical literature that long ago discredited the concept (Constantine, 1993; 1996).

Perhaps most perverse about the Naked Objects propaganda is its promise of empowerment and personal control, which will appeal to many of the very users that it will ultimately abuse. Among the susceptible users are those for whom work, even in this post-industrial information age, is haunted by the ghost of Taylor, the spirit that seems to have inspired the naïvely rigid workflow of all too many lock-step performance-support systems. It will appeal to other users who feel justifiably overpowered by Microsoft-style adaptive interfaces that wrest control from them by changing menus at will or interrupting willy-nilly to offer suggestions or help that is almost never helpful. It will attract still others who have grown tired of so-called wizards that march them blindly down unmarked paths with an implicit I-know-how-and-you-don't message.

Pawson and Mathews toss accusations that user interface designers are threatened by their approach because we have a vested interest. It is in our interest, they would argue, for user interface design to remain a black art requiring long apprenticeships and justifying substantial investment of time and money. Perhaps we should be threatened. Of course, we can counter with equal logic that they and their fellow object modelers have a vested interest of their own in trivializing user interface design and eliminating it from development processes and project budgets. However, before the attacks and counterattacks escalate too far, we need to understand that Pawson and Mathews have a message for us, one that we ignore only at our own peril.

The legitimate essence of this message is about empowering users. We need to work closely with and listen to users to understand the true nature of their work and the problems they face. Then we need to reflect that understanding in the systems we design and build. We—and the interfaces we design—need to treat users as intelligent problemsolvers. We need to deliver innately flexible designs that accommodate to varied styles of work and problem solving, that do not usurp the natural prerogatives of users, that do not force them into rigidly ordered processes, and that do not deprive them of the opportunity to devise better ways to achieve their own ends.

My partner Lucy Lockwood and I end our seminars on usage-centered design with a slide about empowerment. We remind designers that empowerment is ultimately not about attitudes or even involvement but about delivering power into the hands of users. To genuinely empower users we need to give them better tools, tools that fit their problems and enable them to accomplish diverse ends by diverse means.

While the Naked Objects approach may not be capable of delivering such tools, its seductive message is certainly capable of appealing to managers, developers, and even some users—as was dramatically evident at OOPSLA. The emperor may have no clothes, but is he not resplendent in his unabashed nakedness?

(For more on usage-centered design, see <http://www.foruse.com>. For more about Naked Objects, see <http://www.nakedobjects.org>.)

References

- Constantine, L. L. (1993) "Object interface or objects in your face," *Object Magazine*, 3 (7), July 1993. Reprinted in L. Constantine, *The Peopleware Papers: Notes on the Human Side of Software*. Upper Saddle River, NJ: Prentice Hall.
- Constantine, L. L. (1996) "Getting the message," *Object Magazine*, 6 (9), September 1996. Reprinted in L. Constantine, *The Peopleware Papers: Notes on the Human Side of Software*. Upper Saddle River, NJ: Prentice Hall.
- Constantine, L. L., and Lockwood, L. A. D. (1999) *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Reading, MA: Addison-Wesley.
- Constantine, L. L., and Lockwood, L. A. D. (2002) "Usage-centered engineering for Web applications," *IEEE Software*, 19 (2), March/April 2002, pp 42-50.
- Pawson, R., and Mathews, R. (2002) *Naked Objects*. Chichester, England: Wiley.

Learn more about usage-centered design at <http://www.forUse.com>.