On the Function, Design, Execution, and Care of Demos

A. Newell

16 July 1975

Let me proceed by making a series of assertions, I will call all interactions between a visitor (V), a scientist-explainer (E), and an experimental system (X), a *Demo*, leaving open as the matter for exploration what a demo might consist of behaviorally.

1 There are five communicative functions of a demo.

1. *To Claim:* To demonstrate that X has a performance or capability claimed of it.
2. *To Explain:* To exhibit the structure of X (or its behavior), i.e., to use as an audio-visual aid.
3. *To Strut:* To convey the underlying quality and/or style of E's research (or more inclusively that of E's group or institution).
4. *To Entertain:* To entertain, and otherwise enliven or relax, a long and intensive communication effort (e.g., of the departments whole research program).
5. *To Educate:* To educate V in the basics of the field (either computer science generally or of the subarea pertinent to X).

All of these are important. A single demo does not always have all functions (or perform them equally well), though often there is a mixture of several functions simultaneously in a single demo. E.g., rarely does (3) form the focus (though occasionally it does in trying to point out that we operate with a particular interactive style), but it is almost always floating in the background, and a prime part of what V takes away from the demo session. The same is true of (5) especially for V's who are not computer scientists themselves, some, e.g., executive-types of all kind, may get almost all their basic education in a scientific field on the fly by means of such things as demos and official presentations.

2 Demos should work.

This is the obvious point, I say it only because, if I didn't, the story would be incomplete. Equally obvious, though often honored in the breach, are the conditions that seem to be required to make a demo work reliably:

1. *Safe:* The demo system should exist in frozen or saved form; one should not have added a last little wrinkle to the demo system, without retesting (the whole) demo.

2. *Tested:* The demo should have been thoroughly tested. This especially means firing it up from scratch (demos that collapse before they get started are the most embarrassing of all). It also means that a fair amount of playing around the main line of the demo should have been done on the final saved version, to test out the robustness of the demo.

3. *Fresh:* The demo should be fired up from scratch just prior to the demo. Old demos erode.

3 Demos are composed from the following ingredients.

1. *Points:* A particular scientific or technical point, that clearly shows something V can understand and remember–that answers the question "So what?"

2. *Stories (or Story-Lines):* The sequence of events that builds up to making the point. It introduces V to what he has to know to make the point understandable.

3. *Expansions:* Additional optional subparts that can be used to explain aspects that V might not understand, either because of lack of V's preparation, or just because repetition with variation etc., are often needed.

4. *Background:* Additional information on the system that is providing the demo. It is primary expository in nature (though not in form), and is often optional if V wants more general information.

5. *Side Shows:* A demo often offers the opportunity to make a number of points that are not directly contributory to the demo itself. Often these are points about system development, system structure, or research style. They are optional.

6. *Themes:* A set of points may be used to illustrate a theme which no one point does separately. The theme itself may be the real point of the demo, rather than any of the ostensible points.

The reason for such a zoo is so you can ask yourself what the ingredients are of the particular demo you are designing—what it has, and what it is missing. However, not all demos use the full complement of ingredients. They are just that—ingredients from which to compose demos.

4 Demos must have at least one point.

So simple, but so important! Almost as important as that the demo should work.

5 A story-line with point must take only a few minutes.

This is about the maximum length of time V will be willing to attend and follow the demo. This is a few times as long as a commercial. The HEARSAY-1 movie is 20 minutes long and has an immense amount of detail in it compared to what a demo should have.

Demos tend to be longer than a few minutes. But if so, then the demo should have several points and with each point its story-line. (Observer that the story-line is not just preparation, i.e., bringing the system up, it is a sequence of events that engages V's interest and leads to anticipate, hence to be prepared for and to recognize, the point when it finally happens.

One of the reasons for structuring demos with multiple points, expansions and background is to permit a demo to evolve into a longer communicative event. But this should happen because V's interest has been captured.

6 Something should be happening at all times.

Dead spaces in the action are as deadly for demos as they are for movies. Continual action holds V's attention.

V's attention will be captured successfully if almost anything dynamic is going on, no matter how critical. The key aspect is that is be a little unpredictable, so that V does not know exactly what is coming. Repetitive patterns are no good. But the unfolding story need not be deep either. Think of the Westinghouse downtown sign, and why it is successful, though simple.

Demos tend to be long because X has a pace of its own, in terms of initialization and the speed with which it behaves. Thus, most demos have an immense amount of standing around in them. (And professional demo watchers do learn to be relaxed about this.) The use of appropriate background (see below) can make an immense difference here, providing something to do and learn while being paced by a slow system.

7 Backgrounds are essential to good demos.

They accomplish a collection of important functions: being trivial in their computational demands, they are almost sure to work whether or not the points do. Thus they can salvage a demo. Also, they provide a filler that permits rapport and understanding to build up. They are almost always optional and interactive, so they are easy to modulate to the interest and understanding level of V. They often accomplish functions such as (1.3) & (1.4) much more effectively than do the making of points. By being less strident (they simply explain, rather than try to make a point) they raise less resistance on the part of V. (That is, by the Communicative D'Alembert's Principle, points always give rise in V to the attempt (often only covert) to counterpoint.) They also permit education of basic things about computer science, which points rarely do.

8 Initiation, recovery and re-initiation should be easy, safe, fast, and self-declaring.

The start of the demo is the critical moment. Delays here are where you lose V, where his impressions about the competence and style of E, his group, and his institution are set (first impressions and all that).

The self-declaring aspect is particularly critical. The system should continually emit signals that show its progress, so that E can monitor it. Similarly, when it is ready it should say so in no uncertain terms, so that no moments of hesitation and checking are required. These are easy matters to arrange with most of the systems we are working with, but they are no less important for that.

9 Demos should not be (nor appear to be) demo-like.

To be demo-like is essentially to be a canned set-piece, unresponsive to any specific needs of V and the momentary communication situation. It is to be patently a piece of PR communication, which probably conceals as much as it reveals about the ongoing research. It is to be likely out of touch with the actual current research, since it was no doubt put together at some earlier time and frozen there. Such demos make V feel that the demo is not for him, but that he is simply a generalized viewer.

It is of the essence of demos to be demo-like, and much that one does to produce an effective reliable demo contributes to a demo-like flavor. But there are great advantages to rising above it where ever possible.

Some techniques that are useful to make demos non demo-like:

1. *Spontaneous:* Make the demo spontaneous, i.e., run by E in a way that appears that he could do many different things at any point.

2. *Participative:* Make it so that V himself can operate or perform part of the demo where he has some options. This implies strongly that the system itself is not just running down a predetermined path.

3. *Thick:* Use background and expansions, which almost always provide a strong optional character in displaying aspects of X on demand. They make the demo seem much larger than the little path traced out before V—thick, as opposed to thin.

4. *Current:* Have the demo contain really last minute results. This makes it clear that the demo is not an ossified affair that was done once a year ago and is simply drug out for any visitor. It makes the demo vital.

5. *Casual:* Appear to just go to work on X, rather than create a stage-production atmosphere. For example, consider the following scenario: having a detailed time schedule (Demo is assigned machine time from 1110–1135), telephoning in advance to see that all is ready, arriving at the machine and having the dialog... "Ready for the Demo?" "Ready!" "Stand-by...", etc. All this contributes to an effective demo by making it run and run correctly. But all this also contributes to the message that getting the demo to run is a big deal, and not something that happens every day or every time. To be casual and relaxed (while still have the demo work!) is a much preferred style and communicates strongly about point (1.3).

10 Demos should (almost) never be fake.

Actually, the issues of fakery, simulation, editing, speeding-up, and recapitulation are pretty complicated. One dimension runs from fraud through puffery to convenience. Another runs along the attention focusing characteristics of various media—movies, of whatever sort, have some nice properties. Often scientific demands produce forms of legitimate fakery that then have interesting side effects. An example is so-called incremental simulation (a mixed human-computer system to produce at all stages of development a total performance, but with the human role gradually fading out). When can it be said of a system being brought up under incremental simulation that it is operational?

Of the many kinds of fakes, we can distinguish three:

1. *Pure Fakes:* The demo appears to live, but is in fact a stored image of the output or edited output of something run earlier.

2. *Recapitulation Fakes:* the demo is live alright, but nothing Occurs during the demo that could not as well have been cast into hardcopy and simply shown on output. This is a form of instantly generated movie.

3. *Support Fakes:* The total system is in part a faked environment so that the object of interest can run.

Our concern here is with the legitimate pressures to fake. There are plenty! Most movies of computer performances are faked in that the various views of what purports to be a single computer performance never existed as an actual run—a certain amount of editorial freedom is almost always exercised. (It is a (hidden) feature of the HEARSAY-1 movie that all perceived integrated performances are genuine single runs on the PDP10.) The problem is that the movies world must live by fakery when entertaining and has trouble distinguishing what fakery is permitted in documentaries.

The best rule, though an obvious one, is that all fakery must be announced clearly by the demo itself. It is best done as part of the sorts of background information (announcements) that appear in many demos and computer runs of all kinds. Then the fakery has the best chance of being accepted for what it is (or certainly should be), namely an appropriate part of the total scene to maximize communication.

11 Hard copy should be available for V to take away.

The positive function of hard copy is clear. V has only a few minutes (and even less attention) to devote to a demo. Some sort of hard copy adds to the communication by letting V review it later.

Not all hard copy is alike. All types serve the function of giving V something to remember the demo by; but they otherwise can serve quite different purposes.

1. *Fact sheets:* These can give quantitative detail and propositions about results that V can then rely upon (not having to rely on memory). Actually, many details will not be mentioned in the demo itself, e.g., structural details of the system, or performance

numbers. It is especially good to get in the qualifications to performance numbers and claims that tend to get glossed over in verbal presentation. This is a pretty important function: with a fact sheet it is the case that the claims made are those, and only those, that have been written down. However, the fact sheet tends to make the demo non-personal and demo-like.

2. *Output:* Generated output from the system is more memento-like. It does provide a picture of the actual results, and in so far as V saw the very same display, it is a much more direct reminder than a fact sheet.

3. *Protocols:* An especially effective device is a protocol that shows the exact interaction between the human and the system during the demo. It has some aspects of the generated displays, acting as a memento. But it contains a lot of trivial detail or running the demo. Its virtue lies in the fact that it is what happened and thus constitutes about the best record for V to recall what actually occurred.

   Protocols play a special role in instructing people to use a system quickly and effectively, especially when their interactions with the system are circumscribed. This is a strong function in some demos (e.g., those that specifically instruct for later use, and also those that precede "letting V try it").

4. *Reports:* It is important to know what scientific and technical papers have data relevant to the demo and to be able to get copies for V on a moment's notice. Best, of course, is to have copies around that can just be handed to V (providing you want to give away the paper—which is usually the case).

12 Demos should be designed so that they can grow with the system under research.

Demos take time to create and more time to update. Thus arises the tendency to have only an old canned demo, which was created once when the spirit move, but has not been touched since. Up-to-date demos that hold the latest thing are quite effective, but unless planned for they will never get created.

Remember that many us will have visited here before and seen the demos before. The way out of this is to have demos that roll with progress, so there is always something new in them.

13 V should not be left in doubt by the demo about the legitimacy or the quality of what he has seen.

Many (most) V's do not know the field of the demo in detail. Thus, they do not really know whether to be impressed or not. Calibration against the scientific and technical state of the art is an essential aspect of a good demo.

Calibration is often just a question of E getting to know the state of the art of the particular features shown. A demo that can produce as an expansion of the state of the art

elsewhere, as well as the advance that it itself represents, is particularly impressive, though usually hard to do and expensive in demo-development time.

With long term projects (such as C.mmp, SUS, Hydra) the relevant issue is often the prior state of the project itself. Given that V can only assimilate an abstracted view of what he has seen, successive snapshots of large projects (even a year apart) may appear to reveal no forward motion whatsoever. This is a common and important difficulty. In these cases, one needs to be especially concerned that the demo be able to establish the prior art of the project itself so that progress can be understood (in so far as it exists).

For long-term projects the old demo is particular insidious, since it precisely obscures forward motion. The old demo declares one to be at time T-months ago, leaving it to epiphenomenal talk to establish what is really the current state. For V's who are unacquainted with the entire project this is not of much account; for V's who are repeated visitors (e.g., agency sponsors), this may be really critical. As partial compensation, the type of comparison (X_{now} vs. $X_{\mathrm{now}-t}$)) is the one most amenable to the notion of an expansion that shows the prior state, as mentioned under point (12).

14 Several people should be able to demonstrate X.

This is important because it means that a demo can be used even when the creator is not around.

It also provides for important error checking and parallel recovery during a demo when more than one such person is around.

Summary

I have no doubt left out some aspects—I know there is nothing here on adaptation to the audience or on the use of cover stories (which are associated with story-lines, but are not the same thing at all). Also, I have not provided any worked-out examples, so that these precepts can be seen in action (or at least in analysis). All of these things are needed.

I would welcome additions and refinements, as well as evidence of various sorts on particular points and examples that illustrate them (or provide counter examples). If sufficient additional material becomes available, then I will get out an updated version.