https://www.nadin.ws/archives/687

UI's in the Post-Romantic Age of Computation (1993, unpublished)

The romantic age of computation is coming to an end. Finally, those hopes and promises related to computers, and which made for headlines, are starting to materialize. The critical mass in computation is being reached. I write these statements realizing that in the almost 30 years of my involvement with the "analytic engine" and how we interact with it, I took the position, frequently stated in public, that this is still a technology in *statu nascendi*, rather primitive and expensive, and only marginally satisfactory – with the exception of data processing, in particular database management, and more recently scientific visualization. Many factors prompt this maturation: the new scale of computation facilitated by powerful chips; extended virtual address (64 bit and up); higher speed (over 100 MHz) and, even more important, broader bandwidth, facilitating multisensory I/O implementations and thus new interfacing possibilities; multithreading, making possible multimedia; and effective networking, making possible distributed and cooperative computation. Certainly, other aspects related to new forms of computation – neural networks, pen, and VR, optical, and biomolecular computing – deserve to be mentioned. As a matter of fact, the emergence of new molecular materials (biological and nonbiological) that can be used for data processing certainly changes, in dramatic ways, both expectations and possibilities connected to computation. They embody heterogenous alternatives to what used to be a homogenous activity. Indeed, whether data or word processing, computer graphics or even AI applications – all could be reduced to programs and data designed and structured to be accepted by a given hardware configuration, moreover, to eventually take full advantage of it.

The alternatives mentioned make the distinction *general purpose-special purpose* computing more and more meaningful. With the advent of such new modalities, we can no longer apply those reductionist strategies that were used to represent non-computable or non-tractable functions through approximations basically intended to allow for some digital processing (no matter how approximate). And we can no longer afford to waste digital or human resources while we remain blindly immersed in figuring out what the next generation of user interfaces, graphical or not, should be without realizing that the new quality of computation they are supposed to embody is what determines their condition.

It all started with the file

Qualitative changes define this new stage of computation. It is no longer a matter of incremental progress – from 128K to some 1, 10, or even 100 Mbyte in RAM; or from 5 Mbytes to 80 or to 600 Mbytes, or even 1 Gbyte in storage; or from 400 to 1200 to 9600 or even 96000 bps for network communication; or from single digit compression to at least 3-digit loss-less performance; or of performance from some tens of millions of floating point operations (Mflops) to hundreds, but rather to billions, i.e., to G-flops, all with a spectacularly improved price/performance ratio. To put it more clearly: It is a matter of fundamentally new expectations translated into our ability to define fields of human activity impossible without computation. This translates into distributed and collaborative computing, real-time systems, integration of heterogenous machines and of analog and digital processing, and most important, into computing

ubiquity. The new concept of computation is relevant not as an alternative (faster *or* more precise) to mechanical, pneumatic, or electric means, but as a new way of looking at the world in which we live and address our practical and theoretical questions.

The romantic age of computation prompted interest in what used to be called man-machine communication. The expression testifies to a male-dominated culture, anthropomorphism, and very simplistic expectations originating in incipient communication theory. In search of an inclusive description of what was accomplished during this phase, one can say that the attempt was geared towards *pointing to, choosing from, naming*, and later – much later – *illustrating* what machines could accomplish. Pointing to, choosing from, and naming are characteristic of cognitive phases of human infancy. In the mid-1960s, punchcards, teletypes (TTY), and mechanical printers were all there was to the interaction between a central uniprocessor, limited, directly addressed memories, some auxiliary storage (on tape, drum, or small disk), and the people who worked on the computer. The Computer Museum is a very good place to see what kinds of interfaces (to use a word that was made up later) were used back then. Dials and knobs and slider bars are the most visible part of the gauge-based interface. It was very little, but not much more was necessary. The use of gauges was part of the technical culture of the day. The rather immediate level of input/output control made the computing enterprise look like very skilled manual labor, more reminiscent of mechanics than of electronic engineering.

A breakthrough, from the perspective of man-machine interaction, came in the prompt and command linear language made available to developers. For all except the designers and builders of machines, the computer appeared to be in total control. Questions originated within the application (most of the time the computer is one application). There was no interactivity, only an accepted sequence and a lot of confusion when the rigid framework was challenged by attempts to overcome its inflexibility, or by mere typing errors. The batch system was monolithic and definitely closed.

Nevertheless, the very basic element around which user interfaces would eventually emerge was already in place. It was the *file*, a metaphor that proved so powerful that no user interface could escape its influence. In some ways, the entire development up to our days is defined by the adoption of the file as the nucleus around which everything related to digital computing turns. In itself, the file illustrates a limited pragmatic experience with data, an attempt to organize, store, and retrieve it well before automatic processing became possible. It is by no semantic or pragmatic accident that the file was selected for this purpose, but if some other metaphor had been chosen, chances are that user interfaces would have evolved differently. The underlying semiotic basis for this statement is not new.

Concepts are active components of human thinking. They are “designs” of actions and powerful cognitive filters. In retrospect, almost the entire evolution of user interface during the romantic age of computation boils down to the “theater” of the file, or the “magic” of files: file names, operations on files (open, append, close, remove), keeping track of files (by name, time, or type), storage of files (in directories, folders, file cabinets). Regardless of the operating system and application, the file seems to be the very locus of human-machine interaction (notice *man* becoming *human*). Sometimes the metaphor unfolded profitably, and with a certain appropriateness, as in the desktop metaphor, to which I shall return shortly. Other times it

became bizarre, such as at the meeting point between the domain specific to the experience of working with files and that of other metaphors. Recently, a "golden retriever" metaphor was used: the good old dog and the entire vocabulary of throwing a ball to be fetched, a dog tag for labeling, and even a bone in the trash can to suggest the unavoidable operation of deleting. Some might wonder whether I assign too much meaning to the early choice of the file and what I see as the consequences of this choice. Suffice it to say that the file is an implicit *metrics* of computers, affecting the design of computer languages, process interfaces, user interfaces, and applications. As an implicit metrics, it imposed a level of granularity that does not always do justice to the more dynamic aspects of data relations. It might as well be that the 11-character names plus time and date allowed for labeling files in DOS is inadequate. Longer names (in Windows, for instance, or under Finder) are not the answer either. Directories (even in UNIX, very powerful in this respect) and the difficulties associated with operating on them are the immediate consequence of the level of granularity implicit in the use of files. Retrieval in this universe remains rigid, regardless of how powerful search engines are. The technology at this point is, strictly speaking, one of literal problem-solving. Interactivity is not yet in the vocabulary of goals or possibilities of this new human practical experience.

But back to more of what happened after the cathode ray tube (CRT) became the dominant window into and out of computers. First, human-machine interaction became user interface. Actually, the CRT became the interface between the hardware, the program, and those computing or making computing possible. The experience of television viewing helped in the acceptance of CRT displays. But it also projected expectations for which the technology was not yet prepared. From very concrete prompts to selections from a menu, we traveled a cognitive path of increasing complexity, but of a rather constant level of generality. Second, once commands became possible, we accessed a new cognitive realm: that of abstractions. All in all, we moved away from direct interactions (turn a dial, push a tab) to confirmations (related to prompts), choices (from a menu), to mediations between the machine and the person working in some capacity with it. The visual, enticing for many of its qualities (more intuitive, easier to comprehend, modular), soon constituted its own domain as a mapping from abstraction to concreteness. In effect – and this was shown many times – we had a trade-off: the more expressive the means of user interface became, the less precise and less effective they proved. Even in our days, the stricter means of command-based interface are preferred by programmers to the richer environments of visual representation (iconic or otherwise). Screen text editors still come closer to the long-time experience of language that humankind accumulated than to the equivocal culture of images invoked more and more after the emergence of the desktop metaphor.

The desktop

Many things are present in this user interface metaphor. Some were repeatedly discussed (I myself dealt with the semiotic aspects, Nadin, 1988); others not sufficiently understood, though generously copied. The desktop metaphor extends the user interface. Pointing, clicking, and the pull-down action add to the language of human-machine interaction. They are part of the alphabet, actually the vocabulary, of this language, and subject to a grammar that relatively gently expanded from the correctness of a sentence to that of an operation. But regardless of how much the desktop metaphor impacted upon computing and its dissemination in areas of human

activity where few would have predicted – the graphic arts, obviously – it still belongs to a beginning stage of our understanding of a new form of human experience. The desktop *illustrates*, effectively buffering the user from the applications for which it stands through a familiar visual vocabulary. It takes on a life of its own and imposes its conventions upon those working with it. It also imposes rules that often prove so difficult that the more complex an application, the more one wonders why it is subjected to the limitation of a metaphor to which it is not reducible. Word processing under the desktop metaphor is easy to handle, definitely easier than under vt100/text style interface. But Adobe Photoshop™, for instance, is definitely tortured into "clothing" that does not do justice to the pragmatic dimensions of the application. And so was the relational database program Double Helix™ (by Odesta), the first I am aware of that attempted to make available tools for visual programming (the *icon well concept*). The development of Windows™, of the X Window System™ as an industry standard, of OSF/Motif™, of Open Look™, etc. made this point even more obvious. Illustrations, as expressive as they can get, in the end remain illustrations, even if they are rendered digitally. The fact that this direction was in the meanwhile embodied in development tools (APIs, IDTs, UIMs) helped a lot in the dissemination of some families of interfaces, and even in the improvement of their presentation component, but not in coping with the complexity of new applications and new forms of use.

The success of the desktop metaphor is the result of many factors. Foremost, it was conducive to technological progress. It also reflected upon the ways people conceived their own work; that is, it not only presented the computer in a more familiar guise, but it also allowed users to rethink their own patterns of work. Desktop publishing is the result of the bi-directional influence of the desktop metaphor. But as with any metaphor, it soon started being taken literally and being confused with the medium for which it stood. Metaphor theory states that metaphors reach their climax once their referents increase (as we experience through the fast generation of new and quasi-new applications) and, moreover, when dissimilarities among such referents grow above a certain cognitive threshold. Indeed, progress in computation resulted in many new dissimilar applications that the desktop metaphor can no longer account for with an acceptable degree of credibility and efficiency. Moreover, the qualitative change from homogenous to heterogenous computation today renders unacceptable almost all what emerged as effective UI in the romantic age of computation.

This brings up the main point I want to make. What is essential for UIs is the notion of computation they embody and actually make possible – not on the technological side, but on the human side. As all data indicate, the desktop metaphor, along with the many variations it spawned, accommodates design work in 2D. It should by now be clear that, in order to address intrinsically 3-dimensional human activities, we cannot adopt evolutionary strategies. The childish 3D file cabinets, offices, or information rooms (Robertson *et al*, 1993) meant to open access to large information spaces simply cannot do. Practical experiences in 3D (and in 3+D) are of a different cognitive nature. UIs address conceptual problems, not only problems of appearance or illustrative appropriateness, as some people still think. A slanted lid on a trash can, pseudo-3D icons – the by-product of an otherwise remarkable development (NeXTStep™) – and endlessly embedded pull-down menus are only some examples of what can go wrong when we refuse to acknowledge the difference in nature between functional and formal aspects of UI and our obligation to connect them to active use of technology. The pseudo-3D icon, for instance, is

a formal accent, misleading insofar as it promises something (related to the third dimension) that it does not deliver. Aside from the computational overhead, it is strange to have a pseudo-3D icon that offers no additional functionality while target pointers are usually difficult to identify in selecting panes or for resizing or rescaling open documents (this applies not only to Open Look™, SUN, or NeXT, but also to other various attempts to visualize UNIX). Our new obsession with usability tests and measurements, no doubt justified as a means for evaluating how well a UI solution is implemented, is, after all, relevant only for that solution. It is not the answer to whether better solutions are necessary and how to arrive at them. To optimize an illustration is desirable, but not a substitute for eventually understanding why the illustration approach in itself is slowly exhausting its potential.

Human-machine integration

Technologically driven progress in computation required the help of UI in order to serve a growing community of people rethinking their questions in computational terms. Indeed, knowledge became computational to a great extent, and so did communication, management, marketing, and the arts. More chips are present in our daily life than we are aware of. What is new in the new age of computation is that human concerns, from very complex to trivial, are becoming the driving force. This necessitates the rethinking of the computer, including the language of our interactions with it, i.e., the UI, from scratch. Virtual reality is no longer reducible to file management, as it was never a desktop application. And for that matter, one can see why multimedia simply cannot evolve within the limitations of a file-based metaphor or its desktop video editing embodiments. QuickTime™ shows that the desktop cannot support UIs of dynamic nature. Multisensory, broadband, high speed input – many times significant not through the parallel sequences of data, but through the intrinsic relation of data – is another example of aspects not reducible to files (without a definite loss of meaningful qualifiers), or to static metaphors. In some cases, the new computational mode, i.e., pen-based, is added to traditional computation, and the old interface (iconic conventions) is used. That such a solution does not do justice to the new quality of processing is evident. Handwriting as input is qualitatively different from typing. Why then force the new input into the order of old files, when the intended outcome is flexible communication, not portability of data? Moreover, the notion of PDA – a special purpose computing – requires a different pragmatics of negotiations, related to distribution and networking, and implicitly a different perspective of UI. A revolution was announced (this applies to Apple's Newton™, as it applies to EO™, Sharp, etc.); the media came, the consumers are already in the frenzy of collecting another gadget. But after the lights were turned off and the cameras moved to another sensation, we notice that the revolution simply did not take place. Indeed, some new icons (for Names, Dates, Undo, Assist), even where the word would suffice, and a pen interface, which looks good, emerged. But in the final analysis, instead of introducing a different type of human-machine interaction, the designers simply expanded what they already had in stock. Let's face it: The good news is that technology has made available a portable communication station that can be used to control home entertainment equipment, make cellular phone calls, and connect to sources of information important to the user, transmit and receive faxes, and serve as a radio pager station. The bad news is that the slick device does not do justice to its intended functionality because UI considerations remain practically unrelated to the new product due to the lack of an integrated conceptual model. Interactivity, insofar as it is becoming possible through the PDAs, is probably the least supported feature.

Recently (if three years ago can be qualified as recently in an industry where the cycle of obsolescence is close to three years), a report on software architectures and metaphors for Non-WIMP user interfaces raised the issue of alternatives to the desktop metaphor and its many relatives (Green and Jacob, 1991). It is clear that more and more computation professionals realize the need to transcend what has been appropriate until now, but which is becoming more and more of a liability today. But there is no solution in the land of *non*, even the land of non-WIMP, i.e., what is not windows, not icons, not mice, and not pointing. The affirmative, as we know it in and through the new quality of computation, is the fertile ground for the new UI. On heterogenous systems, such as those used in biochemistry research or in new materials engineering, the work of parallel programs can be segmented across different machines.

Incoming data, filtered according to defined goals on a powerful mainframe, can be pipelined to vector multi-processors, then passed to high performance workstations for graphical rendering and displayed as animation sequences. For all this to happen according to design, one has to provide interface tools, graphical or otherwise. Moreover, those new types of UIs will also have to support the design and development of parallel programs. In such cases, the UI itself becomes an abstract model of the complex computation, and a window to the multiple processes that take place. A UI in such an environment is supposed to automatically map procedures to the machines in the heterogenous network. It is obvious that instead of any illustration- or metaphor-based UI, we need interfaces that "know" enough about the operating systems of the various machines networked, about the network, and about the application. The experience gained in this direction by the PVM (Parallel Virtual Machine) project, especially with the graphics interface (HENCE, for Heterogenous Network Computing Environment) is a beginning (Beguelin *et al*, 1993).
UI considerations have to be integrated in the design of computers and applications themselves. Moreover, they should become part of the considerations involved in designing new operating systems. This would finally allow us to move from the illustrative to a constitutive approach. Pragmatic dimensions of computation are defined at the highest level of computer design. They are prefigured in chip design and computer architecture. Very high performance, special purpose processors already integrate in their designs considerations that general purpose machines ignore. A good cache-hit ratio and vectorization facilities support activities that in terms of user interface are very different from those approached on a general-purpose desktop. The trend that we experienced in the romantic age of computation, i.e., 100-fold improved computer performance with a 10,000-fold cost reduction, simply ignored the crucial role of UIs in making what is technically possible a reality of human achievement. If somebody would have added all the CPU cycles available and related them to the *meaningfully* used cycles, we would end up with an efficiency count of which nobody could be really proud. One reason – let us entertain a hypothesis here – is exactly the lack of integration of UI in the hardware and application. Yet another reason for the need to approach UI as a constitutive part of computation (understood as process) is the shift from what some call command-based computing to interactive computing, or better yet, from human-machine interaction to human-machine integration.

What takes increasing precedence is the task we actually try to accomplish, from very sophisticated applications to networking home appliances, not the operation of a machine. The trend is convincingly supported by object-oriented operating systems where the file is replaced by objects and messages. Instead of running a program, an operation relying heavily on user

interfaces (from launching it to establishing pipelines to other programs, and finally, saving, storing and quitting), we have a concert of functions invoked by the data. Editors are replaced by a global resource manager, since the object-oriented operating system acts like a composite editor. We can accomplish real-time systems in object-oriented computation. The former have the specification of correct system state and behavior connected to constraints related to time in the real world. As a matter of fact, the application is the system as it interacts, on a non-command basis, with entities in the world in which we live (air traffic control and vehicular traffic control are good examples of what all this means). An appropriate object-based UI concept for such systems, with emphasis on messages, needs to account for the dynamic nature of the world. It should also provide all it takes for human operators to effectively "insert" themselves in the system, if for no other reason than at least to be able to override behaviors when these might become unpredictable or counter-productive. Representation of concurrency, which is common not only in the control and management of nuclear power plants but also in cooking, and which such systems support, is far from being trivial. But UIs will have to tackle this problem. In order to do so, we have to work on languages of interaction and integration instead of wasting time on interface as theater or magic.

Human-computer integration changes the emphasis from illustration and syntax to dialogue (semantically based) and pragmatics: "This is what has to be done!" One does not need much imagination to see how the ugly boxes that took over the world of offices and some of our home desks finally give way to a physical presence that no longer signifies the transformation of all humankind into typists and terminal (sic!) onlookers. The very low bandwidth of keyboard input (ten characters per second), or of mouse operations (point, select, click), or the modest bandwidth of raster display cannot be compared to the multisensory devices tracking eye or even body motion in 3D, or to the multimedia output of stereoscopic animations against a stereophonic musical background. But neither can the keyboard and the mouse be compared with the unobtrusive tracking devices already available, or with the output capabilities of wide high-resolution flat panels and dynamic musical stereo. The telecomputer, which will make wide choice available (think of 540 channels as only a beginning) requires representation and selection procedures for which no WIMP is good enough, and where formal considerations of graphic communication are no longer of help. These are all new UI aspects, not technological problems; design challenges not marketing schemes.

What we do not have, but desperately need, is the language of interactivity as an integral part of the system's language. Real-time operating systems affording multithreaded communication (multithreading covering the entire I/O area) of large bandwidth need to integrate the basic elements of language interactivity (as this refers to entities in the world or to humans). They need to support variable granularity levels pertaining to multichannel communication. Variable granularity is also essential in processing messages in object-oriented operating systems. A minimal gesture, tracked in the appropriate context, can definitely represent much more than the dialogue boxes of today's UIs. Such gestures can be interpreted within an object-based UI model. Instead of user interfaces tied to a machine, we should look at distributed interfaces, obviously integrated in network operating systems. We should be able to download an interface as the function chosen by the user requires. Yes, this will put variable machines in the hands of ordinary people, to be used for remote working, learning, and creativity. Neither the operating system nor the user interface should buffer the application. None of the commercially available

operating systems currently in use – UNIX (in its many variations), Apple's Finder or its new pen computing system, Windows (NT or not), OS/2, Mach, etc. – even comes close to the real-time operating systems of video games. If computing will ever succeed in our homes, it will have to afford the same kind of operating systems as the games do. The issue is not speed, but directness and predictability of interaction. As things stand today, no multimedia system, regardless of how elaborate its UIs are, does justice to its multi-dimensionality. None offers an appropriate embedded language of human-machine integration into a real-time operating system. In the process of developing an understanding of what languages of human-machine interaction and of human-machine integration should be, we would need to better understand time management, parallelism, distributed and collaborative computing. Technology made possible the generation of computation corresponding to its next level of maturity. Human interface activity can make it real. Computation as a resource, a utility comparable to electricity and telephony, is quite different from the dominant discrete atomistic computation of today. But it will not turn into a resource without the appropriate means of user interface.

References

A. Beguelin, J. Dongarra, A. Geist, V. Sunderam. "Visualization and Debugging in a Heterogenous Envirnoment," in *Computer*, 26:6, June 1993. pp. 88-95

Mark Green, Robert Jacob. "SIGGRAPH '90 Workshop Report: Software Architectures and Metaphors for Non-WIMP User Interfaces," in *Computer Graphics*, 25:3, July 1991. pp. 229-235

Mihai Nadin. "Interface Design and Evaluation-Semiotic Implications," in *Advances in Human-Computer Interaction*, vol. 2 (H.R. Hartson and D. Hix, eds.). Norwood, NJ: Ablex Publishing, 1988. pp. 45-100

George G. Robertson, Stuart K. Card, and Jock D. MacKinlay. "Information visualization using 3D interactive animation", in *GUIs-The Next Generation, Communications of the ACM*, 36:4, April 1993. p. 62