Brown, Peter J. (1996) *Building Novel Software: the Researcher and the Market-place.* In: Milner, Robin and Wand, I., eds. Computing Tomorrow: Future Directions in Computer Science. Cambridge University Press, pp. 21-32.

...

Future Research Directions in Computer Science

Edited by

Contents

1

Building novel software: the researcher and the market-place

P.J. Brown, Computing Laboratory, University of Kent at Canterbury

Abstract

Much of the software we use today arose from new ideas that emanated from researchers at universities or at industrial research laboratories such as Xerox PARC. Researchers who are concerned with building novel software are usually keen to get their creations out into the field as soon as possible. This is partly because they may get extra brownie points from their research sponsors, but, perhaps more importantly, because usage in the field often leads to new insights, which in turn result in the next step forward for the research.

Nevertheless, building something that people will use means making big sacrifices in the research aims. In particular the prime quality of any research, novelty, must be kept in check.

My work for the past decade has been in on-line documents, a topic that is fairly close to the market-place. I will relate some experiences of the trade-offs between research aims and the market-place, and put this in the context of the future lines of development of the subject.

1.1 Introduction

The aim of this Chapter is different from most of the others. I wish to consider researchers at the applied end of the spectrum: people engaged in creating novel software tools. I will analyse my experience over the past ten years in trying, reasonably successfully, to create novel software tools in the electronic publishing area — specifically in hypertext — and will try to draw some general lessons which may help future research.

A typical aim of the software tools researcher is to pioneer a novel interface style, such as a successor to the desk-top metaphor, or to pioneer a new approach to an application area — the Walter Mitty dream that most of us have is a breakthrough equivalent to the discovery of the spreadsheet.

There is a lot of this kind of research going on, and the eventual outcome should be a flow of successful new software products in the market-place. Unfortunately, in the UK especially, this has failed to happen. Although many novel software products on the market have indeed originated at Universities or industrial research laboratories, few of the inspirations have in recent years come from the UK and even fewer are the products of UK companies. For example in an article in *Computing* in August 1993 entitled 'UK firms founder in software league', the results of the Ovum *1992 Software Product Markets Europe* report are analysed: there are only 15 Europe-based firms in the top 50 of software suppliers in Europe, and of these only three are UK-based. (Indeed this total of three only arises from taking a generous view of what UK-based means.) The problem, a much-quoted one, is that the researchers are not getting products to the market-place, and that the UK is worse than most in this aspect.

Part of the problem is doubtless the attitudes of those of us involved in this research; many of us are far too introspective. We develop tools aimed only at computer scientists exactly like ourselves. All too often — certainly in over half the cases — the words 'exactly like ourselves' are just right: the tool is only usable by its creator. The introspection, sadly, carries over to student projects. Frequently, when one sees a list of projects, it is dominated by tasks related to the compiling of programs rather to using the computer for real-world tasks.

It is, nevertheless, far too simplistic to say that the problem can be solved by changing people's attitudes. Indeed even if the researcher does manage to make the mental leap and see the world as a typical computer user would, rather than a highly atypical one like the researcher herself, there remains the phenomenon that brings a mixture of joy and exasperation to all creators of software: users never behave as you expect them to. As, for example, Needham and Hopper(1993) point out, the only thing you can expect is that novel software will be used in an unexpected way.

Thus it is vital, if the creations of researchers are to reach the market-place, to have a feed-back loop involving real usage. A main theme of this paper is to highlight some of the sacrifices that researchers must make in order to make this feed-back possible, and to show some of the benefits.

1.2 The feed-back loop

The ideal sequence proceeds as follows:

(1) the researcher has a brilliant idea — the killer application that will make spreadsheets obsolete, if you want to pursue the Walter Mitty example.

(2) the researcher's team builds a prototype.

(3) real users use the prototype. Some difficulties are thrown up, but, on the positive side, a few users use the prototype in an entirely unexpected yet extremely effective way. The original brilliant idea can be generalised — it is even more brilliant than it first appeared to be. It may be that the unexpected usage is so different that it lies right outside the field of usage imagined by the researcher — indeed the researcher's imagined use may be shunned. Then the original idea is still a brilliant one, but not for the reasons the researcher thought.

(4) a new prototype is built, and step (3) is repeated.

It is vital that the user feed-back is not based on toy examples. As another paper in this volume (Peyton-Jones) argues in depth, unrealistically small examples lead to wrong conclusions.

1.3 An illustration

In order to illuminate this feed-back loop and the general goal of getting research to the market-place, I will now introduce a piece of my past research, whose success is probably due to good feed-back from users. I will try to draw general lessons about the costs and benefits.

The research began in 1982. At that time graphics workstations, following the pioneering work at Xerox PARC, were just beginning to appear. My research was concerned with presenting documents to be read from a computer screen rather than from paper. In 1982 — and it is still partly true today — documents displayed on screens were just reproductions of paper documents. Not surprisingly, given all the advantages of paper, users preferred paper. It would be a huge coincidence if the best way of presenting material on screen turned out to be the same as on paper, especially as displaying a document on the computer opens up new facilities such as searching, adapting material to the user's needs and multimedia. The advent of the workstation, and the new interface style that came with it, opened the way to novel approaches for presenting on-line material.

Note that the aim of the research was based on an application: reading documents from a computer screen. This is perhaps a reason for success. All too often my research has been driven by the technology, and the fascination with the technology has become so great that real-world applications of the technology have faded into the background.

As an aside, a great aid at the beginning of the research was the SERC's bold initiative of making graphics workstations available to researchers in the UK at an early stage. The initiative got a bad name because of problems of management and problems with the chosen PERQ system, but such problems are almost inevitable in any initiative that aims to move before the market does. Now, perhaps because memories fade, my many unhappy experiences with the PERQ can be recalled with a smile rather than a tear. I now feel that my research benefitted greatly from the initiative.

1.4 Industrial involvement

By 1984 the research had produced a prototype, which had seen a limited amount of real usage. The system was called *Guide*. Guide can be classed as a hypertext system, though — perhaps because of its application-oriented aims — it differed from other hypertext systems in that it was not based on screen-sized 'cards' and not based on links. (Indeed the difference remains today, though inevitably, as products have copied the good features from each other, differences have become blurred.)

Probably the most significant event in the whole research programme happened in 1984 when a start-up company, OWL (Office Workstations Ltd) became involved. By 1986 OWL had produced an implementation of Guide for the Macintosh, billed as the world's first personal computer hypertext system. Soon after this a PC product was created, and this has sold tens of thousands of copies — not a blockbuster but a steady earner. Most importantly, the retail product led to much profitable corporate business. For example the Ford Motor Company in the USA has all its workshop manuals in Guide form, and General Motors is treading a similar path.

A huge number of government programmes in the UK have been involved with technology transfer between academia and industry, and it is interesting to recall how the collaboration with OWL came about: *it happened by chance*. OWL, who were — and still are — based in Edinburgh, were visiting a colleague at the University of Kent Computing Laboratory, and were shown Guide as they passed through a room. Everything else developed from the chance meeting. It is significant, however, that, unlike the majority of UK companies, OWL were going round universities in order to find out what new things were happening. On the university side, we were receptive to such visits.

My own experience of working with industry is that collaboration is much easier *outside* the government schemes that supposedly foster industry/university collaboration. The reason is simple: these schemes inevitably have artificial rules and, given any set of artificial rules, academics and industrialists are past masters at cooking up artificial programmes that fit these rules. If you really want to get something done, however, you do better to keep out of the schemes: as a managing director of a small company (who had better be nameless) said to me about a scheme that gave 25% extra government funding: 'I would need more than 25% to put up with all the hassles and delays caused by working within the scheme'.

1.5 Feed-back from the market-place

The OWL products obviously involved considerably change to the University's Guide prototype. The interface was made to fit accepted styles, some esoteric features were dropped, and the general design was adapted to fit a small computer — personal computers really were small in 1986. Nevertheless the principles of Guide remained intact. What OWL did, therefore, was to surround a radical product with a friendly and familiar casing.

The University's prototype itself evolved into a product which has sold in pleasing numbers on Unix workstations. This product involves a lot more research-oriented and speculative features than OWL's Guide does. I will use the term *UNIX Guide* when talking about the University's product.

The OWL involvement led to huge benefits to the research programme. OWL's staff, being closer to the market-place and to technical developments that were about to affect the market-place, gave insights into new application areas, particularly multimedia and CD-ROM. Indeed as a result of OWL, I was a participant in the first CD-ROM conference (Lambert 1986), held in Seattle in 1986. Without OWL I would not have had the opportunity to anticipate the impact of CD-ROM in the electronic publishing field.

Perhaps an even bigger impact than OWL's staff arose from OWL's customers. Inevitably most of the feed-back from customers is, in research terms, mundane: they want features like more flexibility in the use of fonts, or compatibility with the XYZ graphics format. Nevertheless once in a while the real gem comes along: the customer who has exploited a feature in an unexpected but highly effective way. I describe an instance of that later, when I am discussing some technical matters. It is, however, worth quoting at this stage an extreme example of a user gem, since it illustrates so well the difference between the customer's view of the world and the designer's. One customer, Paul Frew, has become a hypertext poet — perhaps the first such. His creation, 'Hyperwalt', has achieved some critical acclaim. He says that he was really excited when he came across Guide's button mechanisms because to him they were a *new literary form.* This was not a concept that figured in the original requirements specification for Guide!

1.6 Success?

OWL's involvement led to a number of successes. Guide won the BCS technical award in 1988, and, in North America, was nominated for, but did not win, the *Byte* magazine award for the year. OWL was taken over at a price per share that was an immense multiple of the original cost.

Ironically, in terms of research funding, these successes were setbacks. It became much harder to obtain research funding, presumably because Guide was too close to the market. I felt this was unfair because Guide had helped create this very market — but then I would. Nevertheless, I sometimes feel that the best approach for continuing research funding is to get close to the goals of the research but never quite to achieve them.

The OWL take-over, though a financial triumph for the original investors, was arguably a set-back for the Guide product. It did not seem to sit very comfortably in a large diverse corporation. Interestingly, the world has now turned full circle, and, as a result of a management buy-out, Guide is now marketed by a small company again: InfoAccess in Seattle.

1.7 Conservative or radical

I would now like to move on to discuss specific issues in running a research programme with the aim of producing prototype products that can be used to get real user feed-back. (For more details of these issues, explored from a hypertext viewpoint, see Brown (1992).) Obviously sacrifices need to be made, and the nearer a research programme is to the theoretical end of the spectrum the less acceptable these sacrifices are. Thus I would like to re-emphasise that I am talking about research at the practical end of the spectrum; at the theoretical end it is often much better to take an idea to its limit and *not* to dilute it with pragmatic compromises.

The first issue for the practically-oriented is this: if you are designing a new approach to software, how radical can you afford to be? As I have said, we all dream of being able to completely change the world, but few, of course, achieve it. Some isolated *dramatic* successes are the early work in the Cambridge University Mathematical Laboratory, the work at Xerox PARC already mentioned, the work at Bell Laboratories, and the first spread-sheet. (It is interesting to note that Xerox PARC and Bell Laboratories were research institutions with a largely free rein — a freer run than UK universities now have.)

To be realistic, particularly if you are a small team, it is safer to be radical in only a few ways. Let me give some examples relating to the Guide work. Users who view a document expect to find a scroll-bar and a searching mechanism. As it happens, the conventional scroll-bar and searching mechanisms, based as they are on static linear documents, are completely wrong for Guide, which has a model where documents appear to grow and shrink, and where documents may be distributed or even created on the fly by running a program and piping its output into Guide. Various radical attempts have been made to design new scrolling and searching mechanisms for this new environment, but none has found any user acceptance. It would have been better in retrospect if the research effort had been confined to Guide's basic document model, which is both novel and acceptable to users, and the accoutrements had been left in their boring conventional forms. Indeed the conventional packaging was surely a factor in the success of OWL's Guide product.

UNIX Guide has maintained a radical approach in its treatment of menus. Any UNIX Guide document can act as the Guide menu: this has the merit of great flexibility and internal consistency, and has indeed been greatly exploited in real applications. It does, however, mean that Guide is not compatible with, say, Motif's menu style, and thus a large number of conservative users are driven away. OWL forsook this approach to menus — neat and elegant as it may be — and stuck to what the conservatives wanted.

The issue of radicalism/conservatism carries down to lower-level issues such as file formats. We researchers always see our own creations as the centre of the world, with the rest of the world's software as a minor adjunct. To combat this, I try to remember the wise words of an OWL salesman "Users do not want hypertext: they want solutions. Hypertext is only ever part of the solution". The statement is equally true if you substitute any other technology for hypertext. Thus if you design a hypertext system it is going to be a loser unless it fits with other software tools such as spelling checkers, databases, version control systems, ... After some early disastrous design decisions had been corrected, UNIX Guide adopted a manner of working that allowed it to be integrated with all the other available UNIX tools. This has been a continuing strength.

It is interesting to contrast this approach with that of another hypertext system, *Intermedia* (Yankelovich *et al*, 1985). In research terms Intermedia has probably contributed more than any other system. However because it worked in its own special world it is now, sadly, largely dead.

1.8 Straitjacket

In spite of the accepted truth that novel software is always used in ways that the designers never imagined, designers often inadvertently impose a straitjacket on the way it is used. Many examples of this occur in the field of CSCW, a field where real usage is now only beginning. A lot of CSCW systems impose roles on the participants, roles that are based on existing practices that are not computer-supported. Inevitably the new environment leads to a change of practices, and the roles that the software imposes make it unusable.

It is, of course, easy to see the mistakes of others. However in my own current research field, hypertext, or more generally, electronic publishing, the mistakes have been as great. Ideally every electronic publishing system should be *policy-free*: the author should not be constrained to a particular style. Certainly an author or organisation should be able to design their own policy, or house style, and impose this, but the system should not impose one all-embracing style. It is a sad monument to our failure, therefore, that if you look at any electronically-produced document — whether a hypermedia presentation or a modest piece of paper produced by a word-processor — you can guess what system has produced it. Thus the system has either imposed a policy or made it hard for the user to escape from some default policy.

This imposition of a policy is particularly tragic in hypertext systems:

we do not yet know how best to present material on computer screens — even textual material, let alone multimedia material — and thus it is particularly unfortunate that all hypertext systems impose an authorship policy.

There are instances in UNIX Guide where, instead of supplying a set of primitive features that can be freely combined, a large number of policies have been dumped into a bucket and the bucket has then been presented to the author. (For those who know Guide the mechanism for glossaries is a particularly bad example of this.) Worse still, a price of having a community of users is that there is a demand for compatibility between one version and the next, thus perpetuating mistakes.

In spite of a lot of bad, author-constraining, features, however, Guide has one shining success, and this relates strongly to the user-feed-back cycle. From the very start, Guide has had a mechanism for grouping buttons (hot-spots) together; when a button within a group is selected, the entire group, which may include ordinary text and pictures as well as buttons, is replaced by the selected button's replacement. The original motivation was targetted at the case where the reader replies to a question posed by the author, e.g. the author might ask the reader to pick a geographical area by selecting one of a group of mutually exclusive buttons labelled *North*, *Central* and *South*. The grouping mechanism was therefore called an *enquiry*.

When OWL's Guide product went out into the field, some imaginative users showed that the enquiry mechanism generalised what was previously thought to be two different approaches to hypertext:

- the approach where a document is a continuous scroll, with buttons within it that can fold/unfold text.
- the approach where a document is split up into separate cards, often corresponding to the size of a screen.

Thus Guide authors had the ability to take either approach and, most interestingly, to take a hybrid approach involving a split-screen. Such an approach, which in the initial design of Guide was not even considered — indeed it still seems ridiculous at first sight — was the basis of one of the most successful UNIX Guide applications. This is ICL's *Locator* system (Rouse, 1991), used to diagnose fault reports.

1.9 Customising

The next, and penultimate, issue I would like to discuss is a siren that is always trying to lure software researchers to their doom. The siren is called customising. The illusion that the siren uses to entice us on to the rocks is the generalised system, which, by setting a few parameters, can be customised to serve a wide variety of roles.

When I was a research student in the late sixties the siren was extolling the merits of extensible languages. A basic core language could be customised, by adding a few tables and procedures, to look like COBOL; a few twiddles of knobs gave it the capabilities of APL instead, whereas for LISP ... . The siren organised extensible languages conferences, to which we flocked. She also showed us UNCOL, the universal intermediate language that could be tailored to meet the needs of every programming language and every machine architecture. Indeed the siren has managed to make use of UNCOL many times over the years.

More recently, in HCI, the siren has introduced the general interface that can be customised to cover everyone from beginner to expert. The interface may even customize itself automatically by adapting its behaviour as the user progresses.

The siren's success in HCI has caused her to use a similar strategy in hypertext. Thus S.S. *Guide* set out towards the rocks which are labelled "the single hypertext document that can be tailored to any type of user and to any type of usage from tutorial to reference". UNIX Guide contains a host of customization facilities, but I have never seen them successfully used except in a most simplistic way.

In any area there is, of course, a degree of customization that is entirely feasible. Hypertext has, for example, its guided tours, whereby a user may, for example, visit a limited number of pages within a more general document. The moral is, however, that such customization has severe practical limits — often because stepping beyond certain limits leads to a combinational explosion — and researchers grossly underestimate what the limits are.

To reinforce the theme of this paper, these gross misconceptions by the researchers are only possible if there is no user feed-back or if the researchers fail to notice that there is a slew of customization facilities that no real user ever seems to master.

1.10 Wrong emphasis

Researchers may, on the one hand, adopt the maxim that the user is always right. On the other hand the researchers may take the somewhat arrogant view that their role is to educate the users. So far this paper has committed itself to the former approach. To add some balance I will end with an issue where the best approach is, I believe, the arrogant one of ignoring the clamour of users saying what they want, and instead providing what the researcher thinks they really need. The issue relates to maintenance.

The whole focus of electronic publishing, including hypertext, has been on one-off or short lived documents. In spite of all the lessons of software engineering, there is no attention to the problems of maintaining documents. Instead the emphasis, which is driven by user demands, is on ever more complex facilities for *creating* documents, and these are likely to make maintenance problems even greater. If I had to guess which research direction in electronic publishing will be most important in the future it will be attention to maintenance, especially, of course, when it relates to really large documents. Some, but not all, of the work in software engineering will have parallels in document maintenance.

Thus I hope that researchers will educate users to switch their emphasis from fancy features for creating documents to such issues as discipline, higher-level abstractions, checking and verification.

Perhaps we researchers also have to educate ourselves. It is human nature that researchers will be more drawn to producing gee-whizz effects than by 'boring' areas such as checking and discipline. Nevertheless as an application area matures the gee-whizz effects become ever more of an irrelevance.

Another way that we computing researchers need to educate ourselves, incidentally, is to spend more time looking outside the computing field. Although I have claimed that maintenance of large documents might be the most important research area in electronic publishing, the issues that will really dominate are ones of intellectual property and copyright, where technical work of computer scientists can only play a small part.

1.11 The future

I will end by first summarising my hopes for the future of electronic publishing, and then the particular message of the paper.

Like everyone else I would like to see a world of electronic publishing where:

- authors can give free reign to their creative talents, helped rather than constrained by their software tools.
- creation of documents, especially multimedia ones, is economically viable even in cases where the potential readership does not run into four figures. This is not an issue I have discussed explicitly here, but it relates to discipline and high-level abstractions.
- maintenance of documents is feasible over long periods, thus allowing documents with a high initial cost to recoup this over time.
- the world that the user sees is an integrated one, with tools slotting together, either visibly or invisibly, to perform the task in hand, and to give the illusion of a single uniform document covering all available information.
- following on from the above, authors and readers do not waste huge amounts of time on the minutiae of finding/obtaining documents, getting them to print out, overcoming incompatibilities, etc.
- finally — and this is the biggest and probably vainest hope of all — a world where new approaches can find their place, rather than a world where twenty-year-old practices lumber on, continually proliferating the twenty-year-old mistakes. My belief is, however, that the best way of achieving this is an incremental one of introducing an advance on one front at a time, rather than aiming to destroy the user's previous world in one go.

Finally, a good way of encapsulating the main theme of this paper is to take a recent quote from Brian Kernighan, referring to one of the many successful and (reasonably) novel products that have emanated from Bell Labs.: "... the reason for its success is that the design is pragmatic, a set of engineering compromises instead of a set of religious beliefs". There is room for researchers to produce elegant failures or pragmatic successes, but perhaps we have too much admired the former rather than the latter.

References

Brown, P. J. (1992). UNIX Guide: lessons from ten years' development, in Lucarella *et al* (Eds.) *Proceedings of the ACM conference on hypertext*, Milan, (ACM Press, New York) 63–70.

Lambert, S. and Ropiequet, S. (Eds.) (1986). *CD ROM: the new papyrus*, (Microsoft Press, Redmond, Wa.)

Needham, R.M. and Hopper, A. (1993). The view from Pandora's box, *SERC Bulletin 5, 2*. 26–27.

Ovum (1993). UK firms founder in software league, *Computing, 19 August 1993*, 7, (news article about the 1992 Ovum *Software Markets Europe Report.*)

Rouse, G.W. (1991). Locator — an application of knowledge engineering to ICL's customer service, *ICL Technical Journal*, **7**, *3*, 546–553.

Yankelovich, N., Meyrowitz, N. and van Dam, A. (1985). Reading and writing the electronic book. *IEEE Computer*, **20**, *9*, 15–30.