Champagne Prototyping: A Research Technique for Early Evaluation of Complex End-User Programming Systems

Alan F. Blackwell
University of Cambridge
Alan.Blackwell@cl.cam.ac.uk

Margaret M. Burnett
Oregon State University
burnett@cs.orst.edu

Simon Peyton Jones
Microsoft Research
simonpj@microsoft.com

Abstract

Although a variety of evaluation techniques are available to researchers of visual and end-user programming systems, they are primarily suited to evaluation of research systems. It is important to have evaluation techniques suitable for real-world programming environments, in order to satisfy real-world product managers of the usefulness of proposed new features. To help fill this gap, we present a new evaluation technique, based in part on Cognitive Dimensions and Attention Investment, called "Champagne Prototyping". The technique is an early-evaluation technique that is inexpensive to do, yet features the credibility that comes from being based on the real commercial environment of interest, and from working with real users of the environment.

1. Introduction

This paper introduces "Champagne Prototyping"[1], a low-budget research technique for use during the early prototyping phase of designing changes to end-user programming systems. It includes a method for soliciting user feedback on a new programmable feature and a method for analyzing this feedback in a structured manner. It combines the important advantages of not requiring availability of a complete prototype, while still allowing limited interactive exploration of a predefined scenario.

We developed our approach in response to the following research situation. We were designing changes to an *existing, large and complex commercial* system (Microsoft Excel), rather than designing or enhancing a research system. Furthermore, we needed to obtain usability information from human users even though there were no student assistants or research staff dedicated to this project. Thus, we needed a relatively cheap approach.

[1] The origin of this name becomes clear in Section 4.

Our approach contains two innovations in research methodology. First, it introduces a new way to use Cognitive Dimensions and the Attention Investment model (described in Section 2), using them to classify actual activities done and observations made by *actual users*. Further, our case study helps to *independently validate* CDs and the Attention Investment model. This validation comes in the form of explanations and observations made by our end users, in which they themselves pointed out instances of CD and Attention Investment concepts.

Second, the approach introduces a new variant of the "Wizard of Oz" technique that is especially relevant to end-user programming research. With the Wizard of Oz, a researcher simulates the system behind the scenes while the user interacts with an interface mock-up. The researcher "executes" the way the non-implemented system is supposed to respond to a user's interactions, causing appropriate output to appear. Evaluating programmable systems is a challenge for Wizard of Oz, because simulating the effect of executing can be almost impossible for a human researcher. For example, if the user's program manipulates large amounts of data, or if the environment contains many subtle interactive features, the likelihood of a human researcher correctly "executing" all the subtleties correctly is small. (Both of these situations are true of most Excel spreadsheets.)

Like classic Wizard of Oz, with Champagne Prototyping users get the impression the entire system is executable. Unlike classic Wizard of Oz, Champagne Prototyping gives the user access to a rich interactive environment that genuinely can be executed on the computer—but the core feature of interest is *not* actually executable by the computer or even by the researcher. The rich executable context allows the user to interact with the system in order to understand the circumstances in which the new feature would be applied and the effects it will have, but finesses the fact that the feature itself has not been implemented.

Champagne Prototyping is still in its infancy, but our experience suggests that it is a promising approach

for evaluating design changes to real-world end-user programming systems. In this paper, we describe our experience with the approach through a detailed case study. We hope that other researchers interested in the complexities of improving existing, real-world end-user programming systems will benefit from this technique, and perhaps, through subsequent use of it, will contribute refinements of their own.

2. Background to the project

In previous work, we had developed an approach to supporting the creation of user-defined functions in Excel, in which user-defined functions are implemented as separate spreadsheets [8]. These "functions as worksheets" are parameterized by example, and can be used in formulae via syntax identical to that of existing functions. (In the current commercial version of Excel, user-defined functions must instead be defined in a procedural programming language.) One implication of this approach is that it must be possible to pass spreadsheet regions to and from these functions as a single parameter. This means in turn that Excel must be extended with a matrix data type, where a whole matrix can be stored in a single cell and manipulated in matrix formulae.

What methods had we applied to address usability issues earlier in the project? Our design work on user-defined functions made use of the Cognitive Dimensions of Notations (CDs) framework, in the manner originally proposed by Green & Petre [5]. Green and Petre hoped that CDs would raise the level of discussion around programming language design, by providing a "discussion vocabulary", for use by designers to discuss and track important properties of a design as it evolves over time. Two of us were experienced in the use of CDs, and we used them extensively during design discussions in order to offer critiques of possible solutions, and to identify likely work-arounds or trade-offs.

The Attention Investment model of abstraction use [2] has also been proposed as a discussion tool, where a simple model of cognition allows designers to anticipate likely end-user programming behaviour. This further design vocabulary was still under development, but we found that, as with CDs, it provided a useful basis on which to discuss and document design rationale for the implementation of user-defined functions.

The results of these discussions were preserved in a record of design rationale, as seen in Figure 1. This figure reproduces a page of our design documentation, in which for each user task (represented by a row), the left column showed our design sketches, notes, and trade-off decisions, while the right column contained

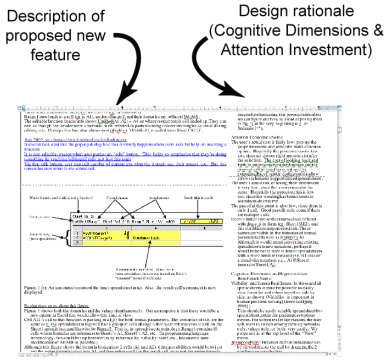

Figure 1. This (deliberately illegible) screenshot shows one of our working design documents, recording design rationale in terms of Cognitive Dimensions and Attention Investment alongside proposed features.

justifications and issues raised by Cognitive Dimensions, Attention Investment, and Representation Benchmarks [9].

3. The problem

At the start of the study we report here, user-defined functions had been accepted by the Excel team as likely to be useful in future versions of Excel. However the precise manner in which the new user-defined functions might be implemented had not yet been finalized. In particular, as noted above, we believed that support for matrix values in a single cell was a highly desirable feature, to support parameters and return values from function sheets. We also believed that matrix values would bring a number of other valuable benefits to advanced Excel users.

However, we encountered a certain healthy skepticism about whether those benefits would really be achieved. In particular, *would our target audience understand the idea of a matrix value inhabiting a cell of an Excel worksheet?* One of the few aspects of the Excel design that has remained constant over many years is that each cell in a spreadsheet contains a single scalar value. If we put multiple values in a single cell, wouldn't this completely disrupt users' mental model of the spreadsheet? The Excel team were quite reasonably concerned about the consequences of making such a radical change to the product.

How can researchers and designers deal with this kind of deep question at requirements specification time? It is always possible to argue to and fro about what will or will not be intuitive to some notional user. But these arguments represent the opinions of designers rather than those of end users.

In short, the challenge we address in this paper is: *how can a designer gather at least some objective data about a proposed design change to an existing, com-*

plex environment, at a very early stage in the design cycle, and at acceptable cost?

3.1 Prototyping

The most obvious step for us to take was to advocate a full-scale user study, based on a working prototype. However, even in a large company, there is often insufficient resource to prototype every potential feature, in order to conduct usability studies on it. Instead, usability testing is usually focused on laboratory studies of already-implemented features, with a view to tuning their design in detail.

Nevertheless—although we know better now—we did indeed construct three different prototypes to explore proposed designs for a user-defined function interface. The first was implemented within a current release of Excel, using Visual Basic. It simulated the creation of a user-defined function by programmatically creating a new sheet within the open workbook, and populating it with parameter values (always the same values). The disadvantage of this technique, beyond the fact that it could only be used to simulate the creation of a single function with a single set of parameters, was that it did not allow us to evaluate user interface features that were different in appearance to standard Excel, since the entire interface was implemented in standard Excel.

We therefore implemented a second prototype in Macromedia Director, using the Lingo scripting language to simulate spreadsheet recalculation for any values entered by the user. Although this allowed us to simulate the appearance of any new interface elements we wanted, the cost of simulating a sufficiently large subset of spreadsheet behaviour in Lingo was far too high to support unconstrained user operation.

We also made a third, "click-through," prototype that was less ambitious than either of the first two. This was implemented using PowerPoint, with each slide in the presentation containing a complete rendering of the user interface, and each successive slide changing in response to a predetermined user action. This allowed us to simulate any appearance of the interface, but with no flexibility at all in the user's interaction – each click has a single predetermined effect. Although it can be used convincingly in demos by a researcher who knows where to click, it quickly becomes unconvincing when used in an experimental situation.

3.2 What we learned

The conclusion of our prototyping work was this: prototypes are often either too expensive or too cheap. While our prototypes helped us to refine our ideas and to articulate them to others, they were inadequate to test hypotheses such as "do users understand matrix values". More specifically:

- To support usability tests, the environment, or context, must be fully functional. It is no good having a new feature that "works" but in a simulation of an existing system that does not work as well as the existing system. This was the problem with Prototype 2.
- Hence the prototype must be based on the existing product, because the latter is too complex to simulate accurately. However substantial new features are likely to be hard to program as extensions of a commercial product, as we found in Prototype 1.
- Screen-shot prototypes, such as Prototype 3, are useful for demonstration purposes, but have obvious shortcomings for usability studies, meeting neither the requirement of a functional context nor a realistic implementation of the feature of interest.

In response to these experiences in prototyping, we combined the most successful features of our early prototypes to create a "middle ground" – the Champagne Prototyping approach described in the next section.

4. A "Champagne Prototype" study

The main burden of this paper is to present and evaluate a technique that allows early and cheap evaluation of proposed designs. The emphasis on cheapness is a feature, not a bug: 20 cheap studies of a variety of ideas and features are likely to be far more valuable than one expensive study of a particular feature. To give a sense of scale, we budgeted around 5 person-days and £100 in direct costs to investigate user responses to our proposed matrix-in-cell feature.

The main features of this approach are:

- A crisp question and a cheap prototype directed at that particular question (Section 4.1)
- A small number of highly credible participants (Section 4.2)
- Scenario-based interviews carried out by a researcher (Sections 4.3, 4.4)
- Coding using CDs and Attention Investment as the analysis framework (Section 4.5)

4.1 The prototype used

Recall that the question we sought to answer was *would our target audience understand the idea of a matrix value inhabiting a cell of an Excel worksheet?* Having learned from our mixed experiences with conventional prototyping techniques, we created a fourth low-cost prototype, specifically to assess user experi-

ence of matrices. It provided a close (but non-functional) visual simulation of the new feature, within a realistic interactive context, so that users could explore its benefits.

To create this prototype, we returned to a current release of Excel as an implementation base, but created pixel-level simulations (in Photoshop) of the appearance of each cell that would contain a matrix value (see Figure 2). The spreadsheet cells in which matrix values would be located were enlarged manually to accommodate the matrix image, which was then pasted into the spreadsheet. Cells cannot have image values in the current Excel release, so the image of the matrix value actually "floated" over the grid, but were precisely aligned so that the cell under the image was not visible. The actual cell value, hidden under the image, was a string containing a simulated matrix formula. Of course the current implementation of Excel does not support matrix syntax, so the string did not start with an "=" (which would cause evaluation, and lead to a syntax error), but with a space character, followed by the rest of the simulated formula.

As a result, when the user moved the cursor position to one of the cells containing the new feature, they would see a formula appear at the top of the screen that appeared to have generated the simulated matrix value. No users noticed the space at the start of the formula, or realized that the features they were exploring were not really implemented. Of course it was not possible for users to change any values in the spreadsheet (because the simulated image would not change, and the "formula" would not be recalculated), but the task was presented in a way that encouraged exploration and interpretation of the experimental spreadsheets rather than modification. This provides a far more realistic experience of programmable systems than is possible with conventional Wizard of Oz techniques. Users were able to explore the spreadsheet exactly as in standard Excel, with all of the usual appearances, options, menus, etc., fully operational. No user in our study attempted to change any values, so none of them realized the functional limitations of our simulation.

To summarise, our prototype uses a simple trick to *layer* new functionality on top of a working system, and it supports a "look don't touch" style of interaction, in which data can be *explored but not modified*.

4.2 Recruitment of participants

A central aspect of Champagne Prototyping is that highly credible participants can be used to evaluate the new feature. Research definitions of end-user developers generally emphasise that the end-user developer

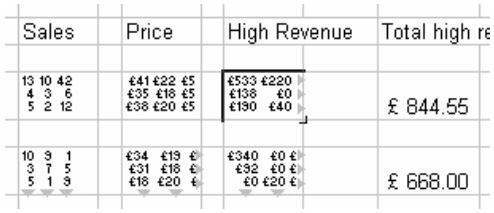

Figure 2. Proposed matrix-in-cell feature for Excel. (Small arrowheads indicate that the matrix extends further than the visible area of that cell)

is a mature expert in his or her own field, who happens not to be a professional programmer [7]. We therefore wanted to recruit study participants who were expert users of spreadsheets, and who had sufficiently sophisticated requirements that they were likely to use matrix data or develop reusable functions, but who were not professional programmers.

Note that we would not expect every Excel user to employ matrix values, even if these are included in future product releases. This is a relatively sophisticated feature, which would normally be used by users who are mathematically comfortable with matrix operations. We therefore needed to recruit participants who, while not professional programmers or computer scientists, were representative of this kind of advanced Excel usage. We chose the field of financial modeling as an ideal domain in which to find such users.

A challenge in recruiting expert participants for research studies is that they are very busy people, whose time is very valuable. Whereas students are generally willing to participate in an experiment for an incentive payment of a few pounds, academic experts in financial modeling are likely to value their consultancy time at hundreds of pounds per hour. We therefore needed an incentive more likely to catch the attention of members of our target population. When our researcher (the first author) entered the office of a prospective participant, he therefore offered a bottle of champagne.

Potential participants were selected from the finance group and decision science group at the Judge Institute for Management Studies. The interviewer approached participants without an appointment, but carrying the bottle of champagne in order to stimulate curiosity and motivate immediate participation. Although it is more common to make appointments for participation in usability studies, our experience is that experts of this kind are unwilling to make appointments other than for essential meetings.

On approaching a potential participant, the interviewer asked two qualifying questions to confirm that he or she used Excel, and was familiar with formulae in Excel. In order to confirm understanding of formulae, the interviewer asked what happens when

you copy a cell with a formula in it and paste it elsewhere. If the participant was aware that cell addresses in a formula would be updated relative to the new position, they were invited to participate in the study. Six participants were recruited.

4.3 Problem scenario and interview structure

The structure of our experimental task was largely determined by our concern with discovering whether users would be able to understand the concept of matrix values. We did not want to prompt participants in advance by making any references to matrices. Instead, we designed a task and interview structure so that users would encounter matrices in the context of regular spreadsheet usage, in much the same way that many users encounter advanced features—in a spreadsheet that has been constructed by someone else, and which they are trying to apply or modify when they encounter the new feature.

We started by explaining that the motivation of this study was to evaluate a proposed new feature of Excel, with the objective of finding out whether or not it is self-explanatory. The purpose of this wording was to focus the remainder of the interview on the product, rather than on user performance. Users often feel threatened when asked to interpret complex features that they do not immediately understand, and we wanted to minimize this possibility.

We then presented a problem scenario, using a spreadsheet that was constructed in standard Excel, with no additional features. A fragment of the sample spreadsheet is shown in Figure 3. The purpose of the sheet was explained as comparing revenue from a chain of stores, in order to determine what proportion of revenue is derived from high-priced and low-priced items at each of two chains during financial analysis for a possible merger or acquisition.

The interviewer drew the attention of the participant to the regular structure of the formulae in this spreadsheet, which involved similar calculations being repeated a variable number of times, dependent on the amount of input data (the number of stores in each chain). This is a scenario in which a matrix might be a more efficient way for a sophisticated user to implement operations on variable sized data, although our interviewer said nothing further at this point to suggest any such possibility. Participants were given the opportunity at this time to explore any aspect of the sample spreadsheet, and to ask any questions, if they wished to do so, about its design or behaviour.

Revenue				**High price item (over £19.99)?**			
	Trous	Shirt	Socks		Trousers	Shirts	Socks
Leeds	£340	£167	£4	Leeds	1	0	0
Cambridge	£92	£126	£15	Cambridge	1	0	0
Edinburgh	£90	£20	£41	Edinburgh	0	1	0
Bath	£62	£48	£20	Bath	1	0	0
Brighton	£155	£96	£15	Brighton	1	0	0
Durham	£0	£240	£32	Durham	1	0	0
Total sales			**£1,561**				
Revenue on high price items							
	Trousers	Shirts	Socks				
Leeds	£340	£0	£0				
Cambridge	£92	£0	£0				
Edinburgh	£0	£20	£0				
Bath	£62	£0	£0				
Brighton	£155	£0	£0				
Durham	£0	£0	£0				
Revenue on high price			**£668**				
Proportion of overall reve			**43%**				

Figure 3. Problem scenario in standard Excel

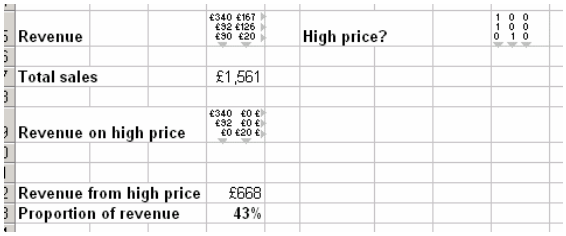

Figure 4. Problem scenario using matrices

Once participants were satisfied that they understood this standard Excel spreadsheet, a separate spreadsheet window was brought to the front. This window simulated the appearance of matrix values using overlaid graphics (Figure 4), in the manner described above. The interviewer gave no explanation to the participant, simply saying that this spreadsheet included the proposed new feature, and that we wished to know whether it was self-explanatory.

If the participant did not notice the modified cells, his attention was drawn to that cell, and he was asked again if it was self-explanatory. He was asked if he could explain how it was being used, and if he understood how the other cells in the spreadsheet were being used. In each case, participants were free to view the (simulated) matrix formulae in each cell. If they did not specifically mention the formulae, they were asked if they could interpret the formula for that cell. If they did not specifically mention the fact that larger matrices do not "fit" visually into a single cell, they were asked what was happening at the edges of that cell. These questions were all carefully designed to avoid giving the impression that we were evaluating the *participants*, instead emphasising that we were evaluating the *system*.

At the end of the interview, participants were asked to assess the utility of the feature, whether they would find it personally useful, and whether they would recommend that it be included in future versions of Excel. They were also asked about their own experience with programming languages, how long they had been using Excel, and whether they were familiar with matrices in a mathematical context.

4.4 Recording technique

The whole of each interview was recorded using a compact digital video camera that was placed on the participant's desk, on a small tripod behind the raised screen of our laptop running the Excel prototype. The video camera and laptop were both running before entering the office of the participant, so that they could simply be placed on his desk for an immediate start to the interview if he agreed to participate.

The interviewer sat next to the participant, in order to be able to point to the screen, and also change from the original scenario (unmodified Excel) to the evaluation version.

Each interview was transcribed by one of the research team, from the point at which participants were asked to interpret the scenario until the end of the session. These transcripts were then used as the source material for the remainder of the analysis.

4.5 Analysis and coding technique

Every transcribed statement from the participants was independently coded by two raters (the first and second authors), by assigning one or more codes from Table 1 and Table 2 to each of the user's utterances. This coding framework included both specific questions regarding comprehension of the proposed matrix feature and also more generic design principles. In the case of comprehension questions, we were interested in whether participants volunteered an observation with no prompting, after prompting, or not at all, as shown in Table 1. Generic design principles were coded by correspondence to the Cognitive Dimensions and Attention Investment frameworks, as shown in Table 2.

As noted earlier, we did not specifically prompt participants with regard to either CDs or Attention Investment. This is in contrast to the research approach taken in the CDs questionnaire [3], where participants are specifically instructed to adopt the analytic perspective of the CDs framework. In the current study, we instead elicited their understanding and opinions of the two "possible" Excel solutions to the same problem, without any use of CD-like vocabulary. CDs were not introduced until in the data analysis phase, as a systematic means of classifying the participants' observations.

4.6 Results of the case study

The results of our particular use of the Champagne Prototyping technique are presented here as a demonstration of the kind of information that can be elicited, even from such an inexpensive study.

Features of matrices that are recognised	Without prompt	With prompt	Not noted
Cells contain matrix values	●●●	●	●
Many cells may map onto one cell	●●●● ●○○○	○	
Matrices can be intermediate values	●●●○	●	
Feature brings labour saving advantages	●●●○ ○	○○	●●

Table 1. Comprehension of proposed matrix feature. Number of bullets shows the number of times each item was mentioned by a respondent. Where a respondent mentioned an item repeatedly, open circles are shown.

Cognitive Dimensions	
Visibility: how to see the remainder of the matrix	●●●● ●○
Error-proneness a) increases scope for error	●●
Error-proneness b) reduces scope for error	●
Abstraction tolerance: allow multiple operations defined at once	●●●
Abstraction barrier: only accessible to expert users	●●
Repetition viscosity: matrices reduce repetition	●
Domino viscosity: matrices reduce the consequent effects of change	●
Hidden dependencies: use of matrices may hide valuable relationships between data items	●●●
Closeness of mapping: relates to matrices users are familiar with	●
Consistency: matrix formulae work like cell formulae	●●●●○
Attention Investment	
Investment risk: what may go wrong when matrices are used	●●
Investment cost: attentional effort of using matrices	●●●
Investment payoff: effort to be saved as a result of using matrices	●●
Manual alternative: basis of decision not to use matrices	●●

Table 2. Frequencies of coded statements from CDs and Attention Investment frameworks. Note that some CDs were never mentioned, and are not included in the table, so this is not a complete list of CDs.

Participant profile. We interviewed six participants. Five had financial or decision science backgrounds, and all had at least 10 years experience of using Excel. The sixth had a more conventional accounting background, and had only 5 years experience of Excel use. He was included in an attempt to explore how matrices might be received by Excel users with less mathematical experience. In fact during the course of the interview, it became clear that he had never created spreadsheets for any purpose other than con-

ventional double-entry accounting, and used only very simple formulae to add and subtract columns. Although we did learn some interesting things about the work of this user, he was not within the target user base for the matrices feature, so is excluded from the rest of this analysis.

The remaining five participants stated (at the end of the interview) that they were familiar with the mathematical application of matrices. Three had some experience of conventional programming languages, all in FORTRAN "long ago". The other two had not used general purpose programming languages, but were familiar with specialist languages for statistical and econometric modeling. Several drew contrasts between the relatively mundane tasks for which they (or students) would use Excel, in contrast to the powerful facilities of these languages.

Comprehension of matrix feature: As shown in Table 1, all participants understood the significance of the proposed matrix feature, and nearly all without prompting. Only one participant failed to comment on the fact that Excel could now support matrix operations, and even this person, when asked if he had experience of matrices, said "yes; if someone hadn't they might not find it as easy," so clearly understood this connection. Several participants made unprompted statements about various labour-saving advantages of the feature ("this is very helpful").

Cognitive Dimensions: Our five participants made a total of 25 observations related to the concerns of the CDs framework. These encompassed both negative and positive implications of the feature, as would be expected by the importance within CDs of recognizing the trade-offs of design changes. A well-known trade-off in CDs analysis is that more abstract features can reduce viscosity ("allowing you to do a calculation just once"), at the expense of introducing hidden dependencies ("there's some hidden stuff going on here") and an abstraction barrier ("there would probably be power users that would use it").

Our visual design "compressed" matrix values into a single cell by using a small font, and indicating that the matrix extended further than the region visible within that cell. This is a key element of our design concept (we were motivated by the CD of diffuseness) but we were surprised at how many participants were concerned about the corresponding trade-off in visibility ("I like to see as much of the calculation as possible").

There were multiple positive comments about the way that our design used existing knowledge, and was consistent with the rest of Excel ("It's treating the cell as an array", "that's very clear").

Attention Investment: Participants made some reference to all of the motivating factors analysed in the Attention Investment framework, including the effort devoted to repetitive manual work ("I guess you'd still have to type in all these other lines as you did before"), the attentional cost of working at a more abstract level ("can't see that for a lot of users they would necessarily use it"), the resulting automation payoff ("then this part would be automatically updated") and the risks that result from working at this increased level of abstraction ("One issue I can see would be error tracking. It's difficult enough when you have the cells more simply calculated").

Commercial questions: Our sponsor also had an interest in the perceived commercial potential of the proposed feature. Two participants said they would recommend inclusion of this feature in the product, one that he would pay extra for the feature, and one that he would receive benefits from the feature via the improvements that it would allow to third party products. Although not directly related to our theoretical interests, these findings will help us to gain further support for research of this kind.

Reliability analysis: 66 coding assignments were made by the two raters. 39 of these were assigned identically by both raters without any consultation, while 25 were agreed when the raters compared their initial assignments. Calculation of Cohen's Kappa statistic shows that agreement between the raters (when corrected for chance agreements) is 88%. This is a relatively high degree of reliability for raters using a predefined coding frame (as opposed to a coding frame derived from the data).

5. Discussion and previous work

There are other low-cost evaluation techniques used in early stages of user interface design, many of them excellent. However, we believe that the Champagne Prototyping approach helps fill a gap that has not been well addressed before, namely evaluating design changes or additions to existing, real-world end-user programming systems.

A common early design step in the process of contextual design [1] is to observe users in their work context, systematically identify their unspoken requirements, and derive new feature proposals from those requirements. This step is useful for designing new features, but does not help to evaluate the new features once they have been designed.

When designers already have a concept for a new design, but it would be expensive to fully implement that design, a common strategy is "low-fidelity" prototyping, where paper sketches of the new design are

evaluated by asking potential users to "operate" them with a researcher acting as a Wizard of Oz to simulate the system response [6]. As we have argued above, in the case of familiar existing systems like Excel with complex internal state, it is extremely difficult to simulate the many features with which expert users readily explore that state.

Where a partially operating prototype is available, a common strategy is to use a "think-aloud" protocol such that users constantly introspect and report on the reasoning underlying their actions in using the prototype [4]. Champagne Prototyping can be thought of as a "neighbor" of this approach in the sense that it too uses freely-offered verbal data; however, its interview format is much more structured than the think-aloud protocol. The interview format directly elicits the exact information sought.

Finally, CDs and Attention Investment are useful analysis approaches, providing a discussion vocabulary for designers to discuss early concepts, but in that role they do not elicit information from users themselves. A CDs questionnaire [3] has also been developed to elicit CD-based feedback from users, but it is only relevant when users are highly expert with the features involved, and thus is not well suited to evaluating completely new features. By using them as a coding framework, we avoid one of the most difficult parts of content analysis, which is the development of a new coding frame based on respondent data.

The Champagne Prototyping technique is therefore complementary to these other methods. It is especially useful in the early design evaluation of new features proposed for complex, real-world end-user programming systems.

The technique requires some mechanism for tinkering with the system's appearances without interfering with the system's ability to execute normally in most respects. (The pasted-in Photoshop images and uses of strings to masquerade as formulas were the mechanisms used in our case study.) Using such mechanisms, the technique introduces the new feature in context, so that informants are able to assess how their work might change as a result of the new feature, including any costs, benefits, risks, and trade-offs they see associated with these changes.

6. Conclusion

This paper introduces Champagne Prototyping, a new evaluation technique for evaluating design changes to existing, real-world visual or end-user programming systems. Its main features are a crisp question, a cheap prototype aimed at the question, a small number of highly credible participants, interview, and evaluation of transcripts using CDs and Attention Investment. Our experience with it, described via the case study in this paper, has shown it to produce useful information at a relatively low cost. This is particularly interesting, considering the amount of effort that we ourselves put into previous attempts at prototyping – work that would have been unnecessary if we had known of this approach in advance.

Furthermore, our case study has helped to validate the CDs and Attention Investment approaches, by demonstrating that users take the initiative to point out these "cognitively relevant properties"—without prior briefing in these concepts or vocabulary.

7. Acknowledgements

We are grateful to Microsoft Research Limited for funding this research, and also to the study participants. Blackwell's and Burnett's work was also supported in part by the EUSES Consortium via NSF grant ITR-0325273.

8. References

[1] Beyer, H. & Holtzblatt, A. (1997). *Contextual Design: A Customer-Centered Approach to Systems Design*. Morgan Kaufmann.

[2] Blackwell, A.F. (2002). First steps in programming: A rationale for Attention Investment models. *Proc. IEEE Symp. Human-Centric Computing Languages and Environments*, 2-10.

[3] Blackwell, A.F. & Green, T.R.G. (2000). A Cognitive Dimensions questionnaire optimised for users. In A.F. Blackwell & E. Bilotta (Eds.) *Proc. 12th Annual Mtg. Psychology of Programming Interest Group*, 137-152.

[4] Ericsson, K. A. & Simon, H. A. (1993). *Protocol Analysis: Verbal Reports as Data*. MIT Press.

[5] Green, T.R.G. & Petre, M. (1996) Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *J. Visual Languages and Computing*, 7, 131-174.

[6] Landauer, T.K. (1987). Psychology as a mother of invention. Plenary Address, *ACM CHI*, 333-335.

[7] Nardi, B.A. (1993), *A Small Matter of Programming: Perspectives on End-User Computing*, MIT Press, Cambridge, Mass.

[8] Peyton Jones, S., Blackwell, A.F., & Burnett, M. (2003). A user-centred approach to functions in Excel. *Proc. Int. Conf. Functional Programming*, 165-176.

[9] Yang, S., Burnett, M., DeKoven, E., & Zloof, M. (Oct./Dec. 1997). Representation design benchmarks: A design-time aid for VPL navigable static representations, *J. Visual Languages and Computing*, 563-599.