How do people learn to use spreadsheets? (Work in progress)

Advait Sarkar
Microsoft Research
21 Station Road, Cambridge, UK
advait@microsoft.com

Andrew D. Gordon
Microsoft Research
21 Station Road, Cambridge, UK
adg@microsoft.com

Abstract

How can we help users discover and learn spreadsheet features? In this work-in-progress report, we present findings from semi-structured interviews with seven participants, which asked questions on the themes of learning and adopting spreadsheet features. We find that feature adoption in spreadsheets is informal, opportunistic, and social. There appear to be three components to feature adoption: discovery, expertise acquisition, and attention investment. Users fall on a spectrum of intrinsic motivation for learning. We close with a few reflections on how features might be designed with adoption in mind.

1. Introduction

Large, complex applications such as Microsoft Excel are feature-rich, but for users, discovering and learning to use features is a challenge. Designers have several methods for alleviating this problem, e.g., callouts ('look at the new features in this version!'), tooltips, help menus, and automatic suggestions. But how well do these really solve the problem? Poorly delivered suggestions can be viewed by users as intrusive and annoying. Principles for mixed-initiative user interfaces have been proposed (Horvitz, 1999), including 'minimize the cost of poor guesses,' 'consider the status of a user's attention,' and 'employ socially appropriate behaviours' – but it is unclear what constitutes 'socially appropriate.'

The ability of users to discover and learn features is key to the success of applications like Excel. Our experiences of speaking with users suggested that rather than benefitting from prompts within the application itself, users pick up spreadsheet tool expertise informally, opportunistically, and socially. This motivated us to investigate whether it is possible to design for social discovery and learning.

2. Method

We recruited a stratified group of 7 participants, 1 female, aged 35-65, using convenience sampling. Participants varied in spreadsheet expertise from minimal usage with only occasional formulas, to high expertise usage including the development of customised add-ins. Participants used spreadsheets for a wide variety of applications such as budget management, energy modelling, futures trading, climate change assessment, accounts reporting, and examination marks processing.

We conducted semi-structured interviews using contextual inquiry (Holtzblatt & Jones, 1993) at the participants' workplaces. The interview was structured around the following questions:

- How did you learn to use spreadsheets?
- How do you learn new features and techniques?
- Can you give me an example of when you learnt a new feature or technique?

Our choice of the phrase 'features and techniques' was intentional; we are interested not only in features (e.g., charts, formulae, formatting, etc.), but also techniques for applying those features. By 'techniques' we refer to craft practices and design patterns (such as the conventions of layout and formatting associated with 'good practice').

3. Findings

3.1. Learning in Excel is informal, opportunistic, and social.

Learning is informal because it is not typically in a classroom setting or with formal learning materials such as textbooks, Excel help, and documentation.

> *[P4, small company accountant] I'm sure people within the accountancy company taught me. They probably showed me… I was a functionary, I was a temporary worker. So I sat down, and they said 'this is what you need to do, input these figures' etc etc. So I didn't even necessary pay attention to what I was learning because I didn't care about the job. The job was a means to an end. But of course when it comes to starting a theatre company, or what have you, or doing my own personal taxes, I realised I have a skill. I have a knowledge base. So in some ways it's being told, in some ways it's watching, and of course you ask questions. You get interested in how people do things. So you ask questions. But I never did a course in Excel. I've never taken a formal learning in Excel.*
>
> *[P5, large company accountant] I don't think there's enough in the Excel documentation online. Looking through the forums you can find people who have similar scenarios to that which you're looking for. And so I've tended to find that a bit more helpful. Sometimes in the forums people will include formulas that you can literally copy and paste.*
>
> *[P6, professional accountant] To figure out a regular expression I would go and look at an O'Reilly book. I still have an O'Reilly book on regex. But my colleagues would just Google it. And that's the experience that I see my children and others going through. It's not one of a formalised reference book. When I learnt SAS, there was a bookshelf and you pulled these tomes, these Dickensian volumes off the thing and wade through the syntax. They don't do this anymore, they just Google it, they copy the code and stick it in.*

Learning is opportunistic because it happens as and when learning opportunities become available, either motivated by a problem which needs new expertise to be solved, or through observation/inheritance of others' worksheets.

> *[P2, examination marking overseer] I know what I know how to use because it's what I've needed. Effectively, I've learnt the functions as I've come across a need to use them, rather than just learning things abstractly.*
>
> *[P6] I knew very little about Excel until I got into a finance course in my second year at University [...] We were studying the arbitrage pricing model, and I tried to implement that in Excel on a Macintosh 2SE. [...] There was this part of the APT (arbitrage pricing theory) that required a normal distribution and there was no function in Excel to do that. And given an obstacle, I then had to get around it, because it was a University deliverable. Given that obstacle, many hours were spent beating my head against the glass pane of that Mac trying to work out how to do it, and as a consequence, I learnt Excel.*

Learning is social because unless the query can be very specifically formulated, users are more likely to seek help from others than from online resources. Users often learn from their co-workers, who see themselves as 'helping' and not 'teaching.' Seeking help from others allows users to learn in-context, because they don't have to re-interpret any examples (e.g., from online tutorial material) for application in their own domain:

> *[P3, energy demand modeller] I didn't have any experience with Excel, so I learnt Excel over a period of 4 years or so. Consultancies work on an apprenticeship model, so there's usually someone more senior on the project, who will advise. [...] you can access advice from people in your team and other gurus around the company [...] I normally ended up on*

analytical modules on projects because that was my background so I ended up using Excel quite a lot. People of that kind in my company sort of pride themselves on their ability to use Excel effectively and efficiently.

[P4] I remember there was this very clever guy who knew a lot about computers who was also part of the team, and he was always willing to help. This was almost pre-email, early 90s. Sometimes I went to them and said 'can you look at this, I don't know how to do this, can you give me a hand' and often they'd just do it and show me. And I became quite friendly with this particular person. So I felt able to disturb him. In this tedious environment, you had to find ways to actually be engaging with people. So it kind of helped to do that.

[P5] I used to have a colleague years ago. He was a real whiz with spreadsheets. Occasionally he would point out 'oh, you can simplify this by putting this in there' and I guess subliminally you take these things in.

[P5] Maybe with certain formulas that I'm including that they haven't come across before. A few people have asked me 'how could I work this out?' and so I might talk them through it. Some of these things I appreciate might seem fairly simplistic when you've done them a few times but other people haven't done them before so you know, helping them to understand how to do that. And then if they don't get it the first time, going back and helping them until they get it. If they can refer to something in prior spreadsheets, it makes it easier for them to understand rather than just having to start from scratch by themselves.

[P5] A lot of this I've picked up on the job. There'll be terms or formulas that I won't understand because they're far advanced beyond my level, so I will have to ask my colleagues to explain things to me. And sometimes because it's so much their day-to-day work and they find it simple to do, they can come across a little frustrated that I don't get it. And they have to explain these things to me. I have to not be worried about asking because I'm not going to learn if I don't ask.

[P6] So the first thing you teach someone coming into the company even with Excel is you teach them the [proprietary database] etiquette. 'This is how we pull it out. This is how the sheets need to look in order to interface with [proprietary database] so that we can use these sheets and other people can use them' and you would guide them. It's almost an apprenticeship, somewhat medieval in its nature. And then people would start to work and then you'd guide them for two or three weeks, through what is deemed 'good practice'. And that includes for example annotations, it includes how you break up the sheets, how you title and label. It can include showing people how to build up an audit trail in the sheet. How to use begin and ends, for example. So you'd coach them through that so they wouldn't try to start and do something stupid like use individual cell references. You'd help them through that process in order to make this stuff more reusable and less brittle.

[P6] We don't write things down. That's a deficiency in the profession. We do try and build personality cults, and that's tacit. There's no handbook on personality cults. If you're good at your job then people will emulate you. If you win their intellectual respect, then they will emulate you. If you're good at your job and you're explaining it and they're seeing the return, they will simply adopt those practices. And a good boss will drive those practices quite deep if they're successful.

Because of the social nature of learning, design patterns and idioms often percolate from influential groups or individuals. Users often learn about a feature, technique, or design pattern when they see it in a spreadsheet someone else has made, and therefore visible features percolate better:

[P3] So for example at some point there was introduced SUMIFS. I must have just seen that in a spreadsheet somewhere and thought 'what the hell is that?' and at some point come back to it and looked at it in help, and decided that was the right thing to do.

[P3] Array formulae was just this sort of mythical thing, it was just sort of this secret incantation that experts knew and you sort of heard of from time to time, until you finally went and talked to them and said 'what the hell is this?' For me, someone gave me a spreadsheet with a funny curly brackets in, and I inadvertently deleted certain cells, and thought 'what the hell is this?' and so, and then you go talk to someone, and they say 'that's array formulae' and you look them up on the web. As far as I know there is no official documentation of the underlying algorithm by which Excel calculates array formulae. By trial and error you learn that some functions like IF behave a certain way in array formulae, but other functions like SUM behave a different way.

[P4] I may have seen on other spreadsheets, notes and comments being used, and thought 'oh that's a good idea' but I realise that I need them. I use them as aide-memoires.

3.2. Adoption consists of discovery, expertise acquisition and attention investment.

There appear to be three distinct components to feature adoption. The feature must be discovered, the user must acquire enough expertise in order to use it correctly, and the user must be motivated enough (believe it gives them a significant enough reward) to invest their attention (Blackwell, 2002) into actually using it.

Moreover, users can be placed on a spectrum between two types: low intrinsic motivation and high intrinsic motivation for adoption. Many users have low intrinsic motivation to acquire additional tool expertise (Aghaee, Blackwell, Stillwell, & Kosinski, 2015). Excel's flexibility allows them to 'cope' in many ways: for instance, users can manually type computed data values instead of learning how to write formulae, users can employ arithmetic primitives (e.g., '+') rather than learn more general functions (e.g., `SUM`); users can merely lay out data in a tabular structure, rather than learning to use the more powerful, formal 'table' features; users can apply filters within tables and copy/paste the filtered view instead of learning to use pivot tables, and so on. We have observed that users' intrinsic motivation interacts with the three components of feature adoption in a manner that is summarised in Table 1.

For instance, users with high intrinsic motivation have the ability to realise the need for a feature and formulate search queries that allow them to learn from online fora:

[P1, climate change modeller] I went to those kind of fora, and then largely trial and error. So trying it out and putting myself in the developers mindset here 'surely this is how they would have done it' and you try out something assuming that this is how they've done it and see if it works as it's meant to work. And in most cases I've sort of sorted it out for me.

[P2] Usually by googling and finding out one of the forums which says 'ah, you need to use function such and such'. There are lots of websites telling you how to do things on Excel. So I come up with what I want to do, and then you google it. So somebody says 'ah you need to use this formula/function', or 'you need to do a macro', or whatever, and then it's you know, go off and make it work, basically.

[P2] Yes. I don't find help in Excel particularly useful as a way of discovering a function. It's very useful when you've worked out what function, to understand how it works. But I don't find it particularly useful for finding the right function to use. I think it works more as a reference manual for how a function works.

[P3] For instance in programming there's a concept of local variables, so certain variables have scope and that's often useful. So I think 'surely Excel has something like that' then after some hunting you discover that you can make named ranges local to a worksheet as opposed to global and that helps in certain cases.

High intrinsic motivation users have a lower threshold for attention investment:

> *[P3] I believed that the spreadsheet would in the end be more robust because your data changes, output changes, things change, and if you do things more generally you're a bit more robust to those changes. Also in some sense the job is not very interesting and it's more interesting if you're learning interesting things. And of course in the end there'll be tasks that were impossible to solve unless you know the more advanced features.*

Depending on motivation, users have different attitudes to the use of advanced spreadsheet features:

> *[P3] And my approach to Excel was always to... one was on any given task, to try and solve the task that was one step more general. So if you have a particular model to build or task to solve, you can do the very specific thing you were asked to do but almost inevitably, come tomorrow, you'll be asked to do something slightly different. Or you can solve a slightly more general problem than the one you're given which has the disadvantage that it takes a bit longer to get to the first answer, but has the advantage that it's faster to get to the second answer when things change. In pushing oneself in that way, you learn to use Excel.*
>
> *[P6] And a lot of my academic progress has yielded insights into that for example running macros and all the rest of it. Often in the workplace you don't have a lot of time to do a lot of macros because the world evolves faster than the macros do. Other people may give you a different experience but automation to me is a very fine double-edged sword. You can achieve efficiencies but it is significant cost initially and if the world changes an inch, you can negate any investment. So a lot of my colleagues never automate beyond formulae, if that makes sense. And they'll pivot table and they'll refresh but they won't do a lot of macros.*

	Low intrinsic motivation	High intrinsic motivation
Feature discovery	Passive discovery: features are discovered when apparent in a received spreadsheet or when informed by a colleague.	Active discovery: users are autodidacts, have the ability to realise the need for a feature and formulate specific queries to learn about features from documentation, online fora.
Expertise acquisition	Informal, opportunistic and social. Learning occurs when a learning opportunity becomes available, but is not sought, and may not be connected to usage opportunities.	Learning is still usually informal and opportunistic, but is less likely to be social. Trial and error is a very common learning strategy. Usage opportunities create learning opportunities.
Attention investment	Users need strong evidence of reward from using a technique or feature.	Users have a lower threshold for evidence of reward. A bricoleur/technophile personality may push some users to apply new features even when there is no direct benefit.

Table 1 – Differences in adoption practices between users with low and high intrinsic motivation.

4. Design implications

What can the insights from this study suggest about designing for greater adoption?

1. Design for percolation: a feature whose presence is visible in a spreadsheet is more likely to be discovered than an invisible feature. This is not about the feature itself being visible - for instance, we are not advocating that the button for the new feature should be placed prominently in the interface. This is about the usage of the feature being apparent - it should be clear, if possible, that the feature has been used. This can sometimes lead to tensions in design. For instance, consider sheet-defined functions (Peyton Jones, Blackwell, & Burnett, 2003), which enable users to define new functions using formulae already present in the grid. Designers might want functions created in this way to look exactly like built-in functions so that users don't need to think about whether they are using a user-defined function or a native Excel function. However, that would design against percolation of the sheet-defined function feature; making it invisible would reduce the propensity for users to opportunistically learn about it.

2. Design for explicit reward. For example, previous work has shown how it is possible to harness curiosity to incentivise users to write more complete tests in a spreadsheet-like end-user programming environment (Wilson et al., 2003). When designing new tools, consider whether it is apparent to the user how using it might reward them for their invested attention.

3. Include influencers as part of the user-centric design process. Microsoft's MVP program[1] shows how such individuals might be identified. If influencers are passionate about, and convinced about the utility of these new abilities, they are likely to be motivated to use them in their spreadsheets, and encourage others to use them too. These users are also likely to have excellent example use cases in which to ground the design process.

5. Related work

Small groups of people within organisations have been found to be responsible for sharing files, establishing and perpetuating 'informally-defined norms of behaviour' (Mackay, 1990). These people could be subdivided into two groups: (1) skilled, highly-motivated end-user programmers who were intent on experimenting and learning the software, and (2) less-skilled end-user programmers who were interested in interpreting needs of colleagues and creating files that solved those needs, facilitating communication between the technical group and the rest of the organisation.

Our data supports previous findings that while beginners learn spreadsheets mainly socially through colleagues, experts are more likely to further their knowledge using books, manuals and online resources, and in either case formal training is not common (Lawson, Baker, Powell, & Foster-Johnson, 2009; Nardi & Miller, 1990).

Our data suggested that the flexibility of spreadsheets permits a variety of 'coping mechanisms' for users to deal with low expertise, without having to acquire additional expertise. These coping mechanisms can be characterised as 'bad practice' (Lawson et al., 2009), which differentiates experts and non-experts; experts perform more planning and design activities when writing spreadsheets. A particularly interesting coping mechanism in highly collaborative environments is delegation: non-expert spreadsheet users collaborate with experts who can complete high-expertise tasks, therefore alleviating the need to learn, although this collaboration sometimes has an informal learning outcome for the non-expert (Nardi & Miller, 1990).

Our data also supports previous findings that spreadsheet learning tends to be goal-driven rather than structured (McGill & Dixon, 2001), an approach that could lead to lower quality spreadsheets as users do not acquire principles of design. Unsurprisingly, spreadsheet users are usually focused on understanding their problem domain, not programming, and therefore while they might become more familiar with

[1] `https://mvp.microsoft.com/`

their domain due to spreadsheet experience, they might not necessarily become better programmers with experience (Grossman, 2007). A related study of web designers found that the decision to learn is more often a matter of necessity than curiosity (Dorn & Guzdial, 2010). A think-aloud study of 10 participants (Reimann & Neubert, 2000) found that participants who self-explain while trying to learn how to use spreadsheets prove to be better problem solvers. Another study found that users learn spreadsheet skills more effectively through problem solving than through watching tutorials (Kerr & Payne, 1994).

General studies in programming expertise show that expertise in programming can manifest in a number of ways. For example, expert programmers have highly organised knowledge that allows them to do better at recall tasks (Wiedenbeck, 2005; Chi, 2006). Beginner programmers understand individual lines of code but not the relationships between them; experts see the more abstract, overall pattern of a program (Lister, Simon, Thompson, Whalley, & Prasad, 2006). Our data did not directly indicate that this was also true of spreadsheets, but would make for interesting future work.

Although we did not find evidence that users acquired expertise through learning scaffolds embedded in the spreadsheet packages themselves, previous work has found that tools that 'self-disclose' assist with learning end-user programming systems (DiGiano & Eisenberg, 1995; DiGiano, Kahn, Cypher, & Smith, 2001). Our observation that features whose use is visible in the spreadsheet creates learning opportunities suggests that scaffolding *social* interaction or stimulating information seeking behaviours might better integrate with users' existing learning practices.

A longitudinal study showed evidence for gender differences in attitudes to technology adoption (Venkatesh, Morris, & Ackerman, 2000). Women were strongly influenced by the subjective norm (expectations of society to be 'normal') and perceived behavioural control (someone's perception of how well they are able to control some desired behaviour of their own – a sense of agency). A key aspect of this study was that there was strong top-down motivation (from managers) to adopt a new technology. This may be the case for some Excel users, but for many there is no such compulsion to adopt new Excel features. Gender differences may be less prevalent in users with high intrinsic motivation.

6. Conclusion

In an semi-structured interview conducted with seven participants, we have begun to learn how people learn spreadsheet packages such as Excel. The preliminary findings indicate that learning is *informal, opportunistic* and *social*, and that feature adoption in Excel consists of *discovery, expertise acquisition,* and *attention investment*. Users differ with respect to their intrinsic motivation to adopt spreadsheet features. In future work we intend to study whether these observations are also true of larger samples, as well as investigate more deeply how users learn specific features, such as formulae.

7. References

Blackwell, A. F. (2002). First steps in programming: A rationale for attention investment models. In *Human Centric Computing Languages and Environments, 2002. Proceedings. IEEE 2002 Symposia on* (pp. 2–10).

Chi, M. T. (2006). Laboratory methods for assessing experts' and novices' knowledge. *The Cambridge handbook of expertise and expert performance*, 167–184.

DiGiano, C., & Eisenberg, M. (1995). Self-disclosing design tools: a gentle introduction to end-user programming. In *Proceedings of the 1st conference on designing interactive systems: processes, practices, methods, & techniques* (pp. 189–197).

DiGiano, C., Kahn, K. M., Cypher, A., & Smith, D. C. (2001). Integrating learning supports into the design of visual programming systems. *Journal of Visual Languages and Computing*, *12*(5), 501–524.

Dorn, B., & Guzdial, M. (2010). Learning on the job: characterizing the programming knowledge and learning strategies of web designers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 703–712).

Grossman, T. A. (2007). Spreadsheet engineering: A research framework. *arXiv preprint*

arXiv:0711.0538.

Holtzblatt, K., & Jones, S. (1993). Contextual inquiry: A participatory technique for system design. *Participatory design: Principles and practices*, 177–210.

Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 159–166).

Kerr, M. P., & Payne, S. J. (1994). Learning to use a spreadsheet by doing and by watching. *Interacting with Computers*, *6*(1), 3–22.

Lawson, B. R., Baker, K. R., Powell, S. G., & Foster-Johnson, L. (2009). A comparison of spreadsheet users with different levels of experience. *Omega*, *37*(3), 579–590.

Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the solo taxonomy. *ACM SIGCSE Bulletin*, *38*(3), 118–122.

Mackay, W. E. (1990). Patterns of sharing customizable software. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work* (pp. 209–221).

McGill, T., & Dixon, M. (2001). Spreadsheet knowledge: An exploratory study. In *Managing Information Technology in a Global Economy: 2001 IRMA International Conference.* Idea Group Publishing.

Nardi, B. A., & Miller, J. R. (1990). An ethnographic study of distributed problem solving in spreadsheet development. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work* (pp. 197–208).

Peyton Jones, S., Blackwell, A., & Burnett, M. (2003). A user-centred approach to functions in excel. *ACM SIGPLAN notices*, *38*(9), 165–176.

Reimann, P., & Neubert, C. (2000). The role of self-explanation in learning to use a spreadsheet through examples. *Journal of Computer Assisted Learning*, *16*(4), 316–325.

Venkatesh, V., Morris, M. G., & Ackerman, P. L. (2000). A longitudinal field investigation of gender differences in individual technology adoption decision-making processes. *Organizational behavior and human decision processes*, *83*(1), 33–60.

Wiedenbeck, S. (2005). Factors affecting the success of non-majors in learning to program. In *Proceedings of the first international workshop on computing education research* (pp. 13–24).

Wilson, A., Burnett, M., Beckwith, L., Granatir, O., Casburn, L., Cook, C., … Rothermel, G. (2003). Harnessing curiosity to increase correctness in end-user programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 305–312).