User experiences in a visual analytics business

Mariana Mărășoiu
Computer Laboratory
Cambridge University
Mariana.Marasoiu@cl.cam.ac.uk

Alan F. Blackwell
Computer Laboratory
Cambridge University
Alan.Blackwell@cl.cam.ac.uk

Abstract

We report on an ongoing ethnographic study conducted at a data analytics company and discuss the multiple facets of user experience observed in this context. We describe in detail two interaction episodes of analysts working with visual analytics software and characterize them through the Patterns of User Experience framework. We discuss the implications of our observations and make some recommendations for future tool design.

1. Introduction

The way in which businesses use data is changing, with more and more companies relying on visual analytics for monitoring, improving or shaping their business. This can be done either in-house, or it can be contracted out to specialized visual analytics consultancies. Together with the increased interest in the larger population for analysing data, it is becoming ever more important to build tools that support this work.

Ethnography and ethnographically informed methods have been previously used in studies of software engineering in order to understand and describe the social context and work practices of engineers in a variety of settings, for example agile development (Sharp & Robinson, 2004) and professional end user development (Prior et al., 2008). At PPIG, ethnomethodologically informed ethnography has been used to discuss programmers reading code by Rooksby and colleagues (2006). They have also been used to study end user programmers (e.g. Nardi & Miller, 1990). Sharp and colleagues highlight the importance of ethnographic studies in the context of empirical software engineering research (Sharp et al., 2016), but their discussion can be extended to other studies of professionals that work with computers on a daily basis. One such group are data analysts.

Recent work studying data analysts includes an interview study of 35 data analysts from various industries (Kandel et al., 2012). The study characterized the process of data analysis in a real-world industrial context, by categorizing three types of analysts on the dimension of tool use, and classifying the activities that the analysts engage in with regards to data processing. We add to their work through an in-depth study of a small group of analysts that specialize in visual analytics for exploring, understanding and reporting data.

In this paper, we use the Patterns of User Experience framework (Blackwell, forthcoming; Blackwell & Fincher, 2010) to characterize ethnographic descriptions, highlighting the experiences we observed the data analysts having whilst they worked working with visualisation tools. We also observe the commonalities between the work of the analysts and previously studied programming-related activities and behaviours. We present these observations alongside ethnographic "thick descriptions" (Geertz, 1973), a format which allows us to discuss the experiences of the analysts in the context of their work.

2. Methodology

The study started in August 2016. The first author was the single observer, and visited the office of Atheon Analytics one day a week, almost every week, for a total of 27 times until the submission of this paper. The observer aimed to take part in day-to-day activities, by attending meetings, having informal discussions with the analysts, observing them at work by sitting next to them, and working at her own laptop at a nearby desk in order to capture the shared office environment.

In the chosen setup, due to the researcher not being trained as an analyst and the relatively sparse visits, there was limited opportunity for more in-depth participation in the work activities of the team, such as working on models and dashboard development alongside the analysts. However, the ability to observe the team over several months offered insights into their work that a shorter but more dense study wouldn't have provided. For example, we were able to observe a wide range of activities and projects, as well as how the projects evolved through various stages of requirements gathering, planning, design, implementation and customer support.

3. The background

To situate the observed episodes that are described in Section 4, we first describe the context of the ethnographic study, the setting and the people observed, and then give a short description of Tableau[1], a commercial software for data visualisation used by the analysts.

3.1 The setting

The office is located in the Cranfield Innovation Centre, a one-storey office building within the Cranfield University Technology Park. The Park is located outside the Cranfield village, next to the Cranfield Airport, on the road that leads from the village to the university. The team first moved here in 2012 from Luton, and since then they have expanded to larger offices twice more, but within the same building. One such moved happened during the study reported here, in December 2016.

Atheon initially started as two separate businesses, Atheon Consulting and Atheon Software Products, which were merged in 2010 under a single name, Atheon Analytics. However, the dual nature of the company has been maintained over the years, as Atheon Analytics offers both consulting services and products for retailers and suppliers. At present, the company has around 22 employees, 4 of which are formally part of the consulting team, alongside the managing director. The study reported here has been conducted by observing this team.

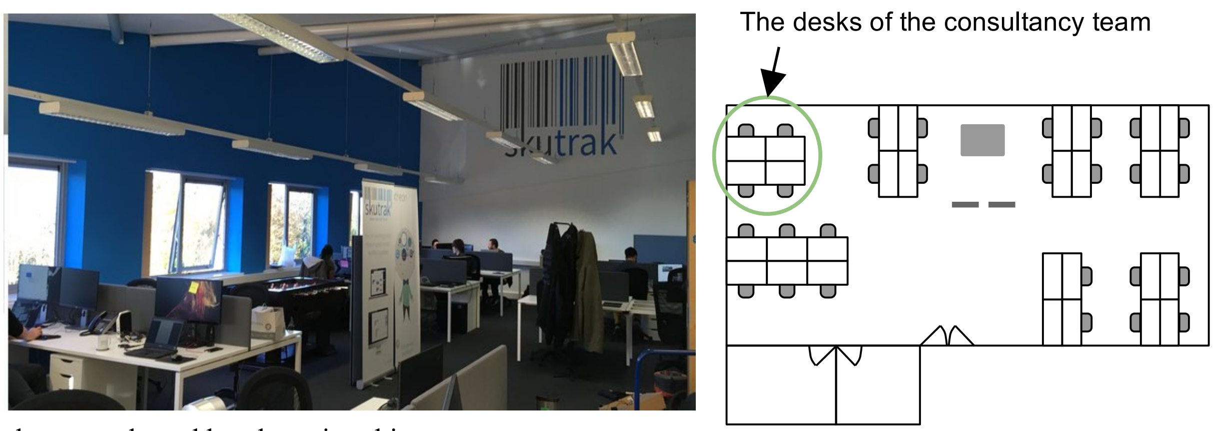


Figure 1. A photo of the office[2] and a sketch of the office plan.

The office is open plan, with desks having partition screens on the long edge. The partition screens were added after the last move and they weren't present in the office when the study started. Each desk of the team members has one or two monitors, but no desktop computers - all the work is done on laptops, with the external monitors connected for additional screen space. A photo and a sketch of the office can be seen in Fig. 1.

The office has a kitchen and a meeting room, both of which got larger when the company moved offices. Besides the usual appliances, the current kitchen has a whiteboard and a table, where people eat, chat or sometimes even have meetings (when the meeting room is occupied).

3.2 The consulting team

The consulting team is located in the far left corner, the 4 analysts working at a four-desk island (highlighted in Fig. 1). The managing director and CEO, who organizes most of the consulting projects and also contributes with analyses sits at an adjacent desk on the right side. The other technical teams are on the other side of the common play area.

[1] https://www.tableau.com/
[2] From http://atheonanalytics.com/news/2016/12/5/atheon-moves-to-its-largest-office-yet

The team has one weekly meeting, usually scheduled for Monday morning, sometimes moved later in the week, where they go through the projects that they're working on, update each other on progress, decide the state of the project for the upcoming week, and add new projects. Usually, there are between 5 and 8 projects that the team is actively working on, with another 5 to 8 in wait, depending on other people, and around 20 more possibilities of future projects.

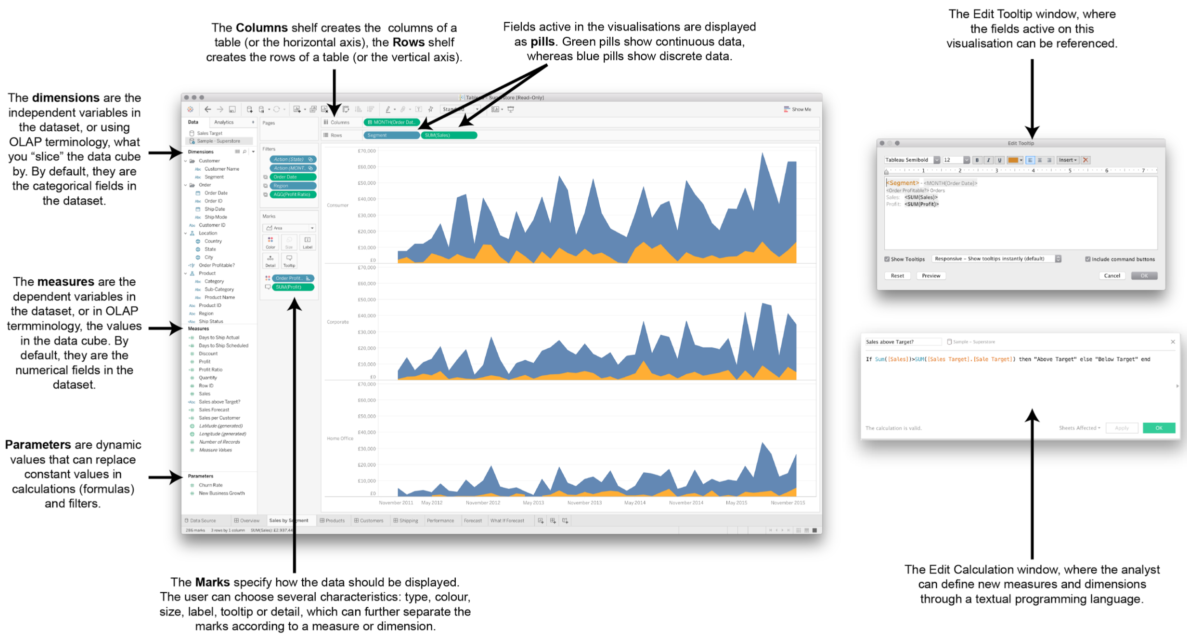
The projects are diverse, ranging from one-off analytical pieces where the output is a presentation of the insights, to building bespoke "tools" for clients that will be used in business decision making, and to maintaining and adding new features to SKUtrak[3], a Tableau-based product that the company provides as a service to a number of retailers and suppliers.

This wide range of projects is interesting for multiple reasons. First, the analyst takes on multiple roles. One is that of the typical data analyst where they're asked to analyse some data and report on it. Another set of roles are those related to building a software product, so the analysts themselves are gathering requirements, designing an interface, develop it and then maintaining it. From our observations, the latter is more prevalent in the work of the team we studied.

3.3. The software used by the analysts

The main visual analytics tool used by the team is Tableau, a commercial software package that allows rapid creation of interactive visualisations and dashboards of visualisations. Tableau is based on previous work visualisation techniques for data cubes and relational databases (Stolte et al., 2002).

The result of a project is often a Tableau workbook. Tableau's file organization is similar to that of Excel: a workbook is a file that contains one or more sheets, which can in turn be worksheets, dashboards, or stories. A worksheet contains a single "view" of the data, for example a table or a chart. A dashboard consists of one or more views laid out on a canvas area. Fig. 2 shows a screenshot of the interface for creating a visualisation worksheet, to aid understanding of the user interaction with the software we describe in the next section.


The **Columns** shelf creates the columns of a table (or the horizontal axis), the **Rows** shelf creates the rows of a table (or the vertical axis).

Fields active in the visualisations are displayed as **pills**. Green pills show continuous data, whereas blue pills show discrete data.

The Edit Tooltip window, where the fields active on this visualisation can be referenced.

The **dimensions** are the independent variables in the dataset, or using OLAP terminology, what you "slice" the data cube by. By default, they are the categorical fields in the dataset.

The **measures** are the dependent variables in the dataset, or in OLAP termminology, the values in the data cube. By default, they are the numerical fields in the dataset.

Parameters are dynamic values that can replace constant values in calculations (formulas) and filters.

The **Marks** specify how the data should be displayed. The user can choose several characteristics: type, colour, size, label, tooltip or detail, which can further separate the marks according to a measure or dimension.

The Edit Calculation window, where the analyst can define new measures and dimensions through a textual programming language.

Sales above Target?

```
If Sum([Sales])>SUM([Sales Target].[Sale Target]) then "Above Target" else "Below Target" end
```

The calculation is valid.

Figure 2. A screenshot of the Tableau data "view" interface.

Whilst most of the observations were conducted when the analysts were using Tableau, they are using a large number of other tools that help them for specific tasks. For example, Alteryx, a data flow programming language, is being used for data processing and data blending, Excel is being used for some analytical tasks and tracking requirements, PowerPoint is used for creating anticipatory sketches of a data visualisation or for presenting results, SQL for querying databases, and Python scripting within Jupyter notebooks for predictive analyses.

[3] https://www.skutrak.com/

4. Findings

In this section, we discuss two observed instances of analysts working with Tableau and other tools. We offer rich descriptions of each observed episode, and, in parallel commentary, a) discuss them using the Patterns of User Experience framework in order to describe the experience of the analysts as they're using their tools (green text), and b) highlight programming-related activities (blue text). The names in the descriptions are fictional.

Due to the context in which our analysts work, we can describe two types of experiences, depending on who the user is. First, we can analyse the *designed customer-experience*: the experience that the analysts aim to design in their tool for their clients. Second, we can observe and discuss the *analyst-experience*, that is, the experience of the analyst as they're building the tool. This dichotomy of experiences appears since both the analysts and the clients use the same underlying software, with the analysts having the knowledge to "program" against it (to create complex visualisations and dashboards, program filters, actions and interactions, etc.), whereas the client is using the resulting dashboards in an interactive, but read-only mode (they're interested in understanding today's data with the given visualisations, not in creating new dashboards).

4.1 Preparing for a workshop with a client

John explains to me what he's working on today: "I have a training session coming up later this week [so I'm now figuring out what they need]. [...] They also gave me a shopping list of stuff [that they want to do]" during the training session, so John is going through that list to make sure he is prepared for the meeting. I only notice this later, but he has a spreadsheet with the requirements from the client team, where he also takes notes of the things that should be brought up during the training session.

The spreadsheet of requirements fulfills an equivalent role to that of a *feature tracker* in professional software engineering.

For now, he's adjusting the width of some Filter widgets in a dashboard. He does this by following the same set of steps for each of the filter: from the menu dropdown of the filter widget he selects "Fixed width", then enters 180 as the new pixel width of the widget.

This activity could be described within the PUX framework as a *modification activity* - the widgets already exist, and the analyst is only changing their appearance. With regard to the designed customer-experience, the goal of this activity is to improve the experience of meaning, and in particular, *ME3: Similar things look similar*. From the analyst-experience perspective, the interface enables to some extent *IE2: Actions are fluid, not awkward* and *PE2: The steps you take match your goal* - the sequence of steps is easy to remember, but it is not supporting *PE5: Repetition can be automated*.

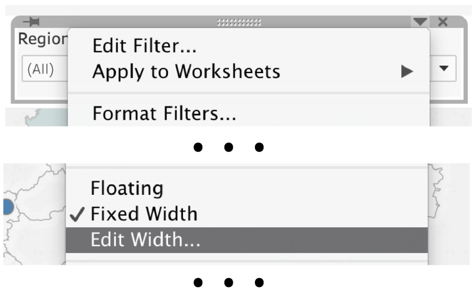


Figure 2. Menu for editing the width of a filter widget

Following this, he renames the sheet by adding " - Working" at the end, then duplicates it and starts to change the previous one more substantially.

We also notice a strategy for *version control*: the analyst creates a checkpoint by duplicating a sheet, renaming it with a description of it's current state, in order to make more substantial changes to the initial sheet.

I observe him doing something more complicated in order to get the title of a chart to appear at the top of the chart rather than at the bottom. After he's finished, I ask him what he just did, so I can understand what he did and why, as it wasn't straightforward from just watching him. He undoes part of the changes he made and walks me through them, explaining what he did.

Starting with a dashboard that has a table on the left side and a bar chart on the left, he goes to the sheet describing the bar chart. He duplicates a pill (Tableau's name for field names which comes from their visual representation, e.g. Region) that already exists in the column shelf, defining the bars of the chart. This creates a new visualisation side by side with the existing visualisation. He then makes the marks of this new visualisation fully transparent, and then merges the two visualisations into a single, dual-axis chart. He then hides the axis and title at the bottom (which corresponds to one of the charts) and he also hides the axis at the top, but maintains its title (this corresponds to the other chart). He explains that he did this so that the chart design could match the table in the dashboard where they are put side by side. John mentions that "in Tableau 10 you can have a grand total in a table and you can chose to have that at the top. But they haven't introduced it for charts."

Even though this is a more complex situation, the end goal is still a change to an existing dashboard - a *modification activity*. Analysing the designed customer-experience, the change would enable the experience *ME3: Similar things look similar*. The experiences of the analyst are similar to that of performing a work-around in the system to get what they want, which is reflected in their last comment. As such, the relevant patterns of experience that are not supported by the interface are *IE1: Interaction opportunities are evident*, *IE2: Actions are fluid, not awkward*, *IE3: Things stay where you put them* and *TE1: You don't need to think too hard*. However, the ability to employ such hacks to achieve a desired goal could be an encouragement for creativity (patterns *CE1: You can extend the language* and *CE4: Anything not forbidden is allowed* apply).

Another request from the clients is for him to give them an explanation for blending data sources, and John mentions that he's going to use a blog article he wrote, which discusses blending vs. joins as a starting point for the training session. John mentions at this point that the client team uses an Excel spreadsheet to select the stores they want to look at. He mentions that one of the disadvantages of using Tableau is that you can't enter data into it - data comes from external sources. However, "for something that works with databases, that's probably a good thing". The way that John is getting around this limitation is to use a spreadsheet file as a data source, join it with the other data sources, so changes to the spreadsheet followed by a refresh in Tableau updates the visualisations.

In this case, the analyst performs an *incrementation activity*, by adding a new data source and combining it with the existing ones. This would later enable *modification* as an activity for the client (changing the data in the spreadsheet and visualising the change in Tableau), as well as *sense-making* (analysing the new data).

Before discussing the patterns of experience, we should note that there is one feature of Tableau has the highest effect on the other patterns. This is the ability to work with data only in read-only mode. On one hand, this enables pattern *IE4: Accidental mistakes are unlikely*, as data cannot be changed accidentally in Tableau. On the other hand, it limits the things one can do in Tableau.

Looking at the example above, in the context of modifying data in Excel and visualising the change in Tableau, the experience pattern *IE2: Actions are fluid, not awkward* is hindered, as the user (whether analyst or client) needs to switch between multiple applications to generate changes in the visualisation.

4.2. Implementing feature requests from clients

Today I'm watching Emily work. Eric, the team manager who sits next to her, mentions that they have some quick fixes for a project that they need to do, asking if she would like to do them. She says "Yeah sure" and Eric continues that some of it is tooltip changes, and a few other things. He says that they don't have to do them, they're not urgent, but Emily switches and starts working on this immediately. Eric sends her the spreadsheet link on Hangouts that contains the list of the requested changes to the dashboard tool. They chat a bit about the changes she needs to make - the spreadsheet has several columns, including a description of the desired change, estimated time, status and notes.

We again notice the use of spreadsheets for keeping track of feature requests, similar in purpose to the *feature trackers* used by software engineering teams.

Emily starts with the first requirement, a tooltip edit. She goes to the visualisation whose tooltip needs changing, then opens the "Edit tooltip" window and makes some changes to the text and the format. She closes the window, then hovers over a few data points to see how the tooltip looks. She then opens the tooltip editor window again and makes some more changes to the formatting. She does this several times, until she's happy with how the tooltip looks.

Within a PUX description, the activity Emily engages in is *modification*: she desires to change the tooltip text and format.
The experience of interacting with the tooltip when hovering over a data point is characteristic of *IE1: Interaction opportunities are evident* and *IE2: Actions are fluid, not awkward*. However the experience of editing the tooltip is more problematic. There is a lack of *PE3: You can try out a partial product* - the interface doesn't allow her to interact with the tooltip in the visualisation whilst she's editing the contents of the tooltip. This also inhibits *TE5: You are drawn to play around*, as the back and forth between the tooltip editor and the visualisation can be perceived as frustrating.

Once she has decided on a format and content structure, she goes through multiple sheets that need to have a similar tooltip and makes the same changes there as well, one by one.

PE6: Repetition can be automated is relevant here: she doesn't have the ability to automate the change across all other similar tooltips, she has to manually edit each of them.

Once she's done with the tooltips, she marks it as done in the requirements sheet and moves to the next one.

In the meantime, the project manager of another team comes over to Eric's desk and they chat for a bit about another project. Emily takes a short break and goes to make herself a tea.

Once Eric finishes the chat, Emily asks him about some of the tasks in the spreadsheet which she's marked with "???" in the "Notes" column.

Emily taking notes of her progress is an *incrementation* activity, with the most relevant experience being *ME5: You can add comments*.

For the first one, Eric points to a chart which needs changing on a specific dashboard from the workbook.

For the second one he tries to explain what the client wants. He takes a piece of paper from his desk and draws a chart saying that this is what the thinks that the client expects. Emily seems slightly confused and comments that in the dashboard it's a rolling week, so the chart shows a full week of data, and that "[she hasn't] seen anything with missing data". She opens the calculation of the field that is shown on the chart, and they discuss what the formula is trying to do.

We observe here Eric sketching a chart in order to support the conversation (an *exploratory design* activity pattern), and the use of a different medium than Tableau: the pen and paper.

After some discussion, Emily says: "This is basically saying, it's doing the difference between the dates, and for some reason it's adding a 2".

When the Tableau visualisation is brought back into the focus of the discussion, the analysts collaboratively aim to understand the calculated fields (a *sense-making* activity within the PUX framework). From a programming perspective, the analysts are *debugging.*

Emily and Eric are talking about the calculation looking at their calendars, trying to figure out why “it’s adding a 2”. Emily opens the formulas for “<Project> week” and “Calendar week” repeatedly, so that they appear on top of the other successively, and they discuss the differences between them.

Eric suggests to see how the fields actually look against a daily date. Emily makes a table in a new sheet, with “Date” as first column, then “Max date”, and then the three similar calculated fields. This creates a table similar to the one below.

Eric’s suggestion to display the fields can be compared to *state tracing* in programming: displaying the internal state of the program with the purpose of debugging it.

The PUX activities are a combination of *Exploratory design* (exploring solutions for understanding the calculations) and *Transcription* (creating the table once the decision to create the table has been taken).

The relevant patterns of experience for the creation of the table are: *IE2: Actions are fluid, not awkward*, *IE3: Things stay where you put them*, *IE5: Easier actions steer what you do*, as Emily only took a few seconds to create the table.

However, the need to create the table in order to understand the results of a calculation suggests that visibility may be an issue (*VE1: The information you need is visible*).

Date	Max date	Week	<Project> week	Calendar week
…	…	…	…	...
../../11	../../28	-1	-2	-2
../../12	../../28	-1	-1	-2
../../13	../../28	-1	-1	-1
../../14	../../28	-1	-1	-1
../../15	../../28	-1	-1	-1
../../16	../../28	-1	-1	-1
../../17	../../28	-1	-1	-1
../../18	../../28	0	-1	-1
../../19	../../28	0	0	-1
../../20	../../28	0	0	0
../../21	../../28	0	0	0
../../22	../../28	0	0	0
../../23	../../28	0	0	0
../../24	../../28	0	0	0
../../25	../../28	1	0	0
../../26	../../28	1	1	0
../../27	../../28	1	1	1
../../28	../../28	1	1	1

After Emily and Eric look at the table and have discovered what the current behaviour is, they then discuss how the client wants it and what changes Emily should make. They discuss both how the visualisation at hand should be changed (Emily says that the clients probably want the weeks with value 0 and 1 to be in the chart, rather than -1 and 0 as it is at the moment), as well as how changing this visualisation would change the others that might depend on the same formulas.

This can be described as a *design* discussion, where the analysts evaluate what the expected behaviour of the visualisation is and how it can be implemented.

After deciding how the visualisation should look, Emily spends some time working out how she can implement the change in the calculated fields.

After some investigation, Emily turns to Eric and says “I’m going to put in a parameter for this screen”. She explains that many of the other charts in the rest of the model use the 0 and -1 check for the weeks to be displayed, so if she changes the formulas that compute “<Project> week” and the other fields, it would affect the rest of the model. She then pauses and comments that she would still have to edit all of them to put the parameter in.

This is a discussion that reflects experiences of structure, and in particular *SE1: You can see relationships between parts* - Emily had to take several minute to find out how the calculations were interrelated, so the relations were not immediately visible. Also, the fact that changing one sheet would affect the rest of the workbook signals a problem for *SE2: You can change your mind easily* - the information structure in the software makes it difficult to isolate changes to one sheet.

After a sidetrack in the discussion, Emily says that in order to make the parameter work, she will have to duplicate all the calculated fields on the current sheet to put the parameter in. Or she will have to rewrite all calculated fields with the parameter.

Eric says: “I suppose there’s no harm there, is there? That’s the more elegant way [the second option].”

They go briefly through the other dashboards to see what would be impacted by the change. They decide to put the parameter in. Emily finishes the discussion mentioning again why using a parameter would be a good idea: otherwise she would have to go through and change 1 to 0, and that would affect the whole model and it wouldn’t give her any flexibility.

The decision that the analysts need to make her is one that is quite familiar to programmers: the tradeoff between increasing flexibility at the cost of adding an abstraction and effort for doing so, or in an Attention Investment description (Blackwell, 2002), deciding between the pay-off that future work will be made easier and the risk that the parameter will never be used anywhere else. The decision is eased by the fact that even if the parameter is not introduced, Emily will still have to edit all calculations that refer to the week indices.

Emily returns to editing the dashboard. She creates a new “Current/Full toggle” parameter. She then edits a calculation using a large “if then/else” statement that depends on the new parameter. She writes the code in, then checks it against the helper table that she built, which is now on her secondary screen. Her work now involves editing the calculations, looking at the visualisations and seeing if anything changed, taking some notes on her notebook, and toggling parameters for testing.

Within the PUX framework, Emily performs *incrementation* and *modification* activities. She is familiar with the interface for adding parameters and editing calculations, so *IE2: Actions are fluid, not awkward* and *IE5: Easier actions steer what you do* apply for her.

She is also engaging in *testing* of the feature she just built, often by comparing the behaviour of the tool with her expectations (*comparison* activities and *SE4: You can compare or contrast different parts* and *VE1: The information you need is visible* are relevant here).

5. Discussion

We can observe that the analysts can be described both as end user programmers and professional programmers - on one hand they are end users of Tableau, and are building visual analyses for specific purposes, to answer analytical questions: the tool only matters to the extent that it allows them to achieve their own goal. On the other hand, the dashboards that they are building are then used by others, and the interactivity and ability to react to new data gives the analysts the ability to generalize the models and “productionise” them.

This creates an opportunity for building tools to support such visual analysis activities, by analogy to those which are traditionally found in software engineering. Previous research into introducing such support for spreadsheets (e.g. for debugging and testing (Reichwein et al., 1999), and for code smells (Hermans et al., 2015)) might be a useful starting point for improving such support for visualisation tools.

Further reflecting on our analysis above, we can observe that the current tool used by the analysts, Tableau, is sophisticated on some axes, but not on others. For example, the ability to do state tracing by displaying data in a table has a higher throughput than stepping instruction by instruction, as in a typical IDE for a textual programming language, and this can result in a quicker understanding of the code being debugged. In this case, the interface supports *IE2: Actions are fluid, not awkward* and *PE2: The steps you take match your goals* when the analyst creates the table, as well as *TE1: You don’t need to think too hard* when interpreting the table. Capabilities for finding out where a measure is used do exist in Tableau, as in modern IDEs. However, the global extraction of a parameter in the second episode is made harder by the lack of ability to search and replace through all calculated fields, resulting in poor experiences of structure and interaction (in particular *SE1: You can see relationships between parts* and *IE2: Actions are fluid, not awkward* are problematic). Refactoring features are available in most modern IDEs. Another area in need of improvement is version control -

as we have seen, duplication of a sheet is one strategy, but it is limited and prone to errors (as we have observed on a different occasion, a sheet was duplicated and renamed, but the modifications that followed were done in the wrong sheet of the two).

The analysts we studied were often engaged in maintaining software (Tableau dashboards) that they previously shared with others (clients, other team and company members). However, while clearly engaging in programming activities, the analysts refer to themselves not as "visualisation developers", but as "data animators"[4]. This suggests that they view their practice more creatively, and that the purpose of their work is not the tools they create, but helping their users gain insights into their data through the tools.

6. Conclusion

We presented two ethnographic descriptions of work typical for a visual analytics consultancy, taken from an ongoing study. We used the PUX framework as a tool for characterizing these rich descriptions, in order to discuss the experiences of the analysts as they go about their work. We discussed two types of experiences - the *designed-customer experience* and the *analyst experience*, and observed that these are often distinct. We also observed that the analysts engage in a number of programming-related activities, which could be better supported by future tools.

7. Acknowledgements

Mariana is a Vice-Chancellor's Scholar and is supported by an EPSRC industrial CASE studentship co-sponsored by BT. She is also supported by a Qualcomm European Research Studentship in Technology.

We would like to thank Atheon Analytics and their consulting team for agreeing to participate in this research and welcoming the first author to conduct the study in their office. We would also like to thank Luke Church for discussions of ongoing work and for his comments on this paper.

8. References

Blackwell, A. F. (forthcoming). A pattern language for the design of diagrams. In C. Richards (Ed.), *Elements of Diagramming*.

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

Blackwell, A.F. & Fincher, S. (2010). PUX: Patterns of User Experience. *Interactions* 17(2), 27-31.

Geertz, C. (1973). *The interpretation of cultures: selected essays*. Basic Books.

Hermans, F., Pinzger, M., & van Deursen, A. (2015). Detecting and refactoring code smells in spreadsheet formulas. *Empirical Software Engineer*, *20*(2), 549–575.

Kandel, S., Paepcke, A., Hellerstein, J. M., & Heer, J. (2012). Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, *18*(October), 2917–2926.

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). ACM.

Prior, J., Robertson, T., & Leaney, J. (2008). Situated Software Development: Work Practice and Infrastructure Are Mutually Constitutive. In *19th Australian Conference on Software Engineering* (pp. 160–169). Perth, Australia: IEEE.

[4] https://dataanimators.com/

Reichwein, J., Rothermel, G., & Burnett, M. (1999). Slicing spreadsheets: An integrated methodology for spreadsheet testing and debugging. *ACM SIGPLAN Notices*.

Rooksby, J., Martin, D., & Rouncefield, M. (2006). Reading as part of computer programming. An ethnomethodological enquiry. In P. Romero, J. Good, E. Acosta Chaparro, & S. Bryant (Eds.), *Proceedings of the 18th Workshop of the Psychology of Programming Interest Group* (pp. 198–212).

Sharp, H., Dittrich, Y., & de Souza, C. (2016). The Role of Ethnographic Studies in Empirical Software Engineering. *IEEE Transactions on Software Engineering*, *PP*(99), 1–1.

Sharp, H., & Robinson, H. (2004). An Ethnographic Study of XP Practice. *Empirical Software Engineering*, *9*(4), 353–375.

Stolte, C., Tang, D., & Hanrahan, P. (2002). Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, *8*(1), 52–65.

Stolte, C., Tang, D., & Hanrahan, P. (2002). Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, *8*(1), 52–65.