10 Lessons from building video product tools in the newsroom.

Pietro Passarelli

Introduction

In 2016 I did a 10-month Knight-Mozilla Fellowship organised by Open News with the Vox Media products team in New York.

My interest was to explore possibilities for open source tools to help streamline the video production process. After spending a month interviewing and observing video producers across Vox Media, I narrowed my focus down to creating an application that enables more efficient and accessible video editing of interviews.

The final product is autoEdit (Passarelli 2016a), a Mac OS X desktop app that creates automatic transcription from a video or audio file. The user can make text selections and export those selections as a video sequence to the editing software of choice.

The success of autoEdit was beyond my expectation - it was downloaded over 700 times within the first three months after its launch. It is currently being used by video producers and journalists across the Vox Media brands, at the Financial Times, CUNY Journalism School, Holocaust Memorial Museum, and many other companies and organizations.

Following are 10 lessons I learned from building video product tools in the newsroom. The knowledge and expertise that lead me to approach product design the way I do is the result of my unusual background.

My education background includes a B.A. in Anthropology from Goldsmiths, University of London, a MA in Documentary Films from London College of Communication, and a Msc in Computer Science from University College London. My professional background is also a unique mix, I worked in documentary production for BBC, channel 4, and discovery channel, I also had the opportunity to work as a newsroom developer at The Times & Sunday Times in London, where I created quickQuote, a tool that automatically generates interactive video quotes, which won Guardian's Student Media Startup of the year award in 2015, and worked with the Guardian Video team.

In the following section, I am going to distill the key takeaways for journalists and developers interested in using user-centered design and rapid prototyping to create products that gain user traction.

User Research

One of the biggest challenges when doing research to develop a new product is how to get useful actionable intel from users while iteratively developing a prototype.

Let's consider a few key ideas that can help when defining a course of action.

The Lean methodology, pioneered by Eric Ries(Ries 2011), defines an approach to startup and product development initially inspired by the Toyota quick cycle of manufacturing. The focus is on learning, and to iteratively define and test hypothesis to expand understanding of the users and the problem we are trying to solve, rather than on building and deploying a polished product.

This goes hand in hand with the idea that there are going to be early adopters that you should spend some time identifying as a defined group and focus your initial research on (Blank and Dorf 2012) .

The idea of "participant observation" is a key element of ethnographic research in anthropology. In social sciences context you don't just observe, measure and quantify, instead, you need a better way to gather qualitative insights in a rigorous way. Participant observations consist in embedding yourself with the subjects you are studying, taking part in their routines and activities to better understand the world from their point of view, and see what that reveals to you about their knowledge systems and beliefs.

As anthropologist Bill Watson, from Kent University, often says, anthropological research has also a strong emphasis on looking at "what people do do, and not what they say they do".

Closely linked with these ideas is the concept of "The Mum test" (Fitzpatrick 2013) . In a nutshell, it says that to get actionable insights to the users you have to root the questions in the past. Using past behavior as more reliable predictor of future decisions. And avoid asking leading questions.

For example if you want to figure out if your users would be interested in using automated transcriptions as a part of their video workflow, Rather than asking "Would you use an app that does X & Y?" first things first, you'd have to learn about their current workflow. "Tell me about the last project you worked on, what was that like from start to finish?". It is then down to you to drill down to the granularity that gives you the most insights. Then you can ask what the biggest bottleneck is and what the users have done to try and simplify things.

In user-centered design approach, building personas should not feel like a chore. Personas are abstract representations of users that embodies some of the common characteristics shared across your pool of early adopters. These are useful in the development and prototyping stage, when defining non-functional requirements.

For instance, asking the users about their work, and what gives them the most satisfaction and what the most tedious part is, is also a great way to get a more rounded sense of who they are and how they think about things. This will also come in handy when considering how to position the product to them in a relatable way.

Last but not least, anthropologists would encourage you to have a healthy distrusts for questionnaires as a research methodology to gather insights. However, if you use the "Mum Test" when phrasing the questions, it can be a useful tool to identify early adopters.

Be a domain expert, but if you are not that’s ok too.

A domain expert is someone who has expertise in a certain field, knows and understands the intricacies of contradictory and conflicting theories in that field, and can juggle those in their mind.

For example, a youtuber might be an expert in doing social videos, but if they don’t know about ethics and other epistemological implications of the medium, they might not be a domain expert in documentary production.

If you are not a domain expert in the area where you are looking to use user-centered design to build a successful product, that’s ok too, there’s a few ways you can make up for that.

Once you have defined the domain you are going to be exploring, the first thing to figure out is what are considered as best practices, “expert view” on the topic. For instance in the area of using transcriptions for video production, a paper-edit is still considered as the best practice although having fallen out of fashion because of the tedious and time-consuming process in its analogue form.

You also need to identify out who is regarded as expert, what are the different school of thoughts, etc. Then as a separate step, find out how your users feel about these experts and school of thoughts, which one they respect, which one they dismiss, and why. For example, in the case of the paper-editing workflow, most video producers either do a simplified variation of it, or don’t do it because it is too time consuming. While sometimes video editors dismiss it as working with text not being visual enough for their liking.

Observe and learn from your users. Don’t take their word at face value, they might not always know what’s best for them. In this sense there is a strong parallel to documentary production where you have to research, analyse, interpret and draw your own informed conclusion to define a course of action. For example one of the feedback I had from a video producer at Vox about autoEdit is that it “Goes above and beyond what I thought was possible”.

Be language agnostic

As a technologist, you should figure out the best technology to use for the problem you are trying to solve and to factor in the learning curve that comes with it.

In computer science domain, It makes your life a lot easier if you have a focus on understanding the evergreen underlying knowledge, as it moves a lot slower than the latest hip framework that changes every 3 months.

The tools and framework that we use shape the mental models we create to solve problems (Cook 2017), and some might be more suited than others. For instance, in the first version of autoEdit (Passarelli 2016a) I used ruby on rails with SQL as a database. This lead me to model the transcription after a captions, srt file, with line level granularity, which had severe limitation when trying to select text at word level. However in autoEdit2, when working with node and nosql databases, it was easier to reason at a word level granularity and enable more possibility for digital paper-editing (Passarelli 2016b).

As the popular saying goes “if you only have a hammer everything becomes a nail”.

Last but not least, Eric Ries has also argued that, in line with user-centered design you should also start with the users and the problem they are facing rather than trying to fit a specific technology to a specific problem (Ries 2011).

Map the problem domain

I also found it particularly helpful to map the problem domain as a strategy to deal with ever changing requirements (Winder and Roberts 2006).

“A problem domain is the context in which a particular problem exists. For example, the problem domain in which a specific route plan exists is that of maps, route planning, travelling and strategies for moving around. Critically, the problem domain is relatively stable, changing only slowly, while specific problems to be solved are transient and change regularly. If you are able to capture the problem domain as the core of the design of your program, then the program code is likely to be more stable, more reusable and more easily adaptable to specific problems as they come and go.” (Winder and Roberts 2006, 351)

“Consider wanting to know how to travel from A to B. You could ask someone for a route plan and get a list of instructions containing statements such as ‘go to the end of the road, turn left, then turn right at the third turning on the left’, and so on. With a basic understanding of how each statement is interpreted (e.g. how to count turnings to find the third on the left), you can follow the instructions and get to your destination”.(Winder and Roberts 2006, 354)

“A given route plan may work but it is very specific. To find out how to travel to a different destination you have to go and ask for a new list of instructions every time you want to travel. An alternative strategy for dealing with your travelling problem is to make use of a map. This will allow you to travel between any two points on the map and do your own route planning. The route planning can even be done while you are travelling. There is a cost for using the map, you have to learn how to read and interpret it, as well as develop strategies for planning routes. However, in the longer term the benefits more than outweigh the initial learning costs. Moreover, much more of the map strategy is reusable—it is easy to transfer the solution strategy from one problem to the next.” (Winder and Roberts 2006, 355)

use component-based design

Another helpful technique is that of component-based design when writing the code of your program (Winder and Roberts 2006)

“A component is typically implemented by a small collection of classes, with one class acting as an interface to the component. Rather than designing a program from scratch, it can be built out of a set of predefined components which are linked together with small amounts of new code. This approach is termed component-based design. The connections between components are enabled using the mechanisms of inheritance, interfaces and dynamic binding. Users of the component make use of the public interface to call the component’s methods but need not be aware of the details of the component implementation.”

(Winder and Roberts 2006, 265)

Thinking in terms of reusable components has several advantages. For instance autoEdit “front end” is written in web technologies in such a way that the demo on the project website is the same code but with a hard coded database. This means that if I were to make a web version of the app I could reuse that part with minimal adjustments. It ultimately comes down to making a judgement call between premature optimization vs being strategic. Often a thin line.

Use an R&D Approach

Once you have got some findings from initial user research, you are ready to take the next step to identify, learn, and understand possible workflows. Then you can treat it almost like a lego project.

For example, for autoEdit, an understanding of the paper-editing workflow and how the users relate to that and/or a variation of it was crucial.

To get a better feel for this, I run a series of workshop on paper-editing, where participants were learning interview story crafting and I would get the type of insights you normally get through focus groups, focusing on the underlying workflow and not the app itself. (Passarelli 2017) (Passarelli 2016c)

Once you have the workflow figured out, next step is to divide it into parts. Of those parts identify the granularity of what are the components that make up that part. For those components, find out which ones you have available and to what degree you are familiar with their workings.

As well as which ones you don't know. Priority is given to the ones you don't know to verify how they work and whether they make the whole possible.

Once you have all the components you look at the interfaces and the communication between that, do you need to do any conversion or adjustment eg to get the output of one as input of the other etc.. This also helps you to think about data models representations that are most suited for the overall system.

This allows to work on parts and components in isolation and then combined them. Don't leave the combining and integration experiments too late tho because sometimes they are just as important as the building blocks.

Threat everything as an hypothesis and prioritise which one to be tested first.

Learn how to formulate questions as a way to get over unexpected roadblocks during development

It is not a question if you are going to get stuck, but rather a matter of when. It is therefore important to think about how you are going to get yourself out of it and use your time productively.

Face with an unexpected problem, think about what you know, the limits of what you know, what you don't know and the limits of what you don't know. Then describe what you know about the current setup, and inner workings. Describe the symptoms of what's not working as expected. Write it all down. This takes you already half way to find a solution, as you can now share it and communicate with others more efficiently when asking for advice. It's important to also spend some time identifying the right vocabulary to express the problem. You can then iteratively make hypothesis about the root cause, and run experiments to test it out. This will help you narrow it down until you find a solution.

Be brave like Pixar

I once watched this documentary about Pixar, The Pixar Story (Iwerks 2007), and was fascinated by how Toy Story team had the courage to throw away all the work just a couple of days before deadline, and started over from a blank canvas because the story and its main character was not quite working out.

This might seem daunting but If you had mapped the problem domain, followed component-based architecture and user-centered design, you really are just throwing away only the material representation of your ideas. And you are actually in a good place to start over, with all the excitement and liberating anticipation of a blank canvas.

Human vs machine?

In the context of where to draw the line in the automation debate in replacing people's jobs, especially with the recent buzz around AI, in studies that compared the performance of a team of humans, a team of "machines" and a mixed team of humans and computers, have found that the mixed team seems to consistently achieve the highest result (Allègre L and Matthias 2013).

I've been dwelling on the ethical issues around the development of autoEdit, for a while now, there is a possibility that, especially if combined with more advanced cognitive services, it could replace video editors.

After all IBM Watson can edit a video trailer (Smith 2016), with pretty good results.

There are already examples of Earthquake bulletins written by machines without human review and the same has happened for economic reports (Jenkin 2016).

In this context It's important to recognise that there is always a bias, of some sort, gender, race etc.. of whoever wrote the algorithm (Devlin 2016) .

In the current version of autoEdit, the aim was to remove the tedious parts of the paper-editing workflow to enable the video producers to concentrate on the story crafting.

Convenience trumps quality every day of the week

In certain contexts of automation, convenience seems to win over quality every day of the week. Increasingly leading to a polarized scenario with either high volume production with low quality output or low quality production with high volume distribution, and no room for anything in between.

An example is the use of text to speech to to do voice over of videos, where quality of the speech to text services is not as good as human narrators but can scale faster and cheaper.

In the interest of adding value for the users, I would argue it is your job as an application developer to address both convenience and quality. Identify convenience hooks to get traction with your users, but be concerned about raising quality of the output. Don't expect an increase in quality to be sufficient for new user adoption, however counterintuitive this might seem.

For example, with autoEdit the 5 minutes turnaround time for transcriptions is a very important cutoff point that was instrumental in gaining traction with early adopters. And even automated speech to text service alone is not as good quality compared to transcriptions services with 24 hours turnaround time with services that provide human or human + machine transcriptions. But the convenience wins over quality. And therefore some users will put up with the decrease in quality of the transcription they get for the added value of a fast turnaround.

Conclusion

In this article, we explored 10 lessons I learned from the development of autoEdit, and walked through how a mixed approach that combines lean, user-centered design and anthropological insights with key computer science techniques, such as mapping the problem domain and component-based design, can be instrumental in developing products that gain user traction. I would like to encourage readers to think deeper and further about how they can leverage some of the techniques I have explained to add value and increase quality of their products for their users.

Bibliography

Allègre L,. Hadida, and Seifert Matthias. 2013. "3 Humans + 1 Computer = Best Prediction." https://hbr.org/2013/05/3-humans-1-computer-best-prediction.

Blank, Steve, and Bob Dorf. 2012. *The Startup Owner's Manual: The Step-By-Step Guide for Building a Great Company*. 1 edition. Pescadero, Calif: K & S Ranch.

Cook, Blaine. 2017. "Annotations Models." https://pietropassarelli.gitbooks.io/textav/problem-domains/d83d-dd2a-2705-2b07-fe0f-annotations-models.html.

Devlin, Hannah. 2016. "Discrimination by Algorithm: Scientists Devise Test to Detect AI Bias Technology The Guardian." https://www.theguardian.com/technology/2016/dec/19/discrimination-by-algorithm-scientists-devise-test-to-detect-ai-bias.

Fitzpatrick, Rob. 2013. *The Mom Test*. 1 edition. CreateSpace Independent Publishing Platform.

Iwerks, Leslie. 2007. "The Pixar Story: Leslie Iwerks: Amazon Digital Services LLC." https://www.amazon.com/Pixar-Story-Leslie-Iwerks/dp/B006DQP64A/ref=sr_1_fkmr0_1?ie=UTF8&qid=1501371860&sr=8-1-fkmr0&keywords=The+Pixar+Story+documentary+dvd.

Jenkin, Matthew. 2016. "Written Out of the Story: The Robots Capable of Making the News Guardian Small Business Network The Guardian." https://www.theguardian.com/small-business-network/2016/jul/22/written-out-of-story-robots-capable-making-the-news.

Passarelli, Pietro. 2016a. "autoEdit." www.autoEdit.io.

———. 2016b. “autoEdit User Manual.” https://pietropassarelli.gitbooks.io/autoedit2-user-manual/content/paperediting.html.

———. 2016c. “How to Craft Compelling Stories Out of Video Interviews?” http://pietropassarelli.com/wip_london_july2016.html.

———. 2017. *How to Tell Compelling Stories Out of Video Interviews*. https://www.gitbook.com/book/pietropassarelli/how-to-tell-compelling-stories-out-of-video-inter/.

Ries, Eric. 2011. *The Lean Startup: How Today’s Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. 1 edition. New York: Crown Business.

Smith, John R. 2016. “IBM Research Takes Watson to Hollywood with the First ‘Cognitive Movie Trailer’ - THINK Blog.” https://www.ibm.com/blogs/think/2016/08/cognitive-movie-trailer/.

Winder, Russel, and Graham Roberts. 2006. *Developing Java Software*. 3 edition. Chichester, UK ; Hoboken, NJ: Wiley.