Important examples of modern team collaboration problems and published user interface software solutions

Martin Kraft

Abstract— Designing programs and user interfaces for multi-user, multi-device and multi-application (mmm) setups demands very specific attention to problems that single user applications do not have to cope with. In this paper I discuss a few of the challenges involved in one of the major tasks of mmm programs, the modern software and project development with teams of multiple collaborators using different hard- and software.

Index Terms—multi-user, collaboration, development process, design process, user interface

1 Introduction

In the last 25 years the standard procedure for a team of collaborators working together from different locations, using different hardware and software has changed dramatically. The requirements for programs and user interfaces for multi-user, multi-device, multi-application setups and the interaction with the programs had to be developed from scratch and led to the discovery of important yet very different aspects, solutions and problems. Different published papers have identified and mapped many problems onto the defined requirements. In this paper I would like to talk about the following problems and present published solutions trying to solve them.

- Ownership hierarchy: When people work together in a shared live environment to achieve progress every user has to know and understand what space belongs to him, what possibilities he has and how he can interact with the existing elements presented to him. [1] [7].
- User input management: The complexity of managing constant inputs from different users in a shared live environment with getting the right actions done, updating all the users of the general progress and getting specific types of interactions and feedback back to the right user. [2] [7].
- Version control: In case the collaborators are not working simultaneously on a live canvas they need to be able to synchronize their personal achieved progress and merge this progress together to create a working, data conflict free version. [1] [2].

2 Ownership hierarchy

Existing ideas reach from users exclusively owning parts of the overall screen space to create and present single ideas [1] or all users working together on one big surface to create and finish a collaborated concept. [7] Both concepts have their unique problems. When giving out equal spaces to all users, very active collaborators can run into physical boundaries when they used all their existing space while unused space is being blocked by inactive users. When the space is granted dynamically pressure can build on collaborators who want to take their time or aren't that active because their available space decreases. [1] If people don't know or understand the interaction with the features they could potentially not only miss out on creating new ideas but also critically damage existing progress by deleting other user's content or shared content on a general canvas. User interface elements in an mmm setup must be easy to understand and simple to use. They need to help preventing possible catastrophic errors by starting easy and only handing out complex functions in more difficult layers to professional users who really need those settings. [2]

The Google document collaboration *(see Figure 1)* shows an easy way to display live collaboration by offering limited functionality for a multi-user text editing software.

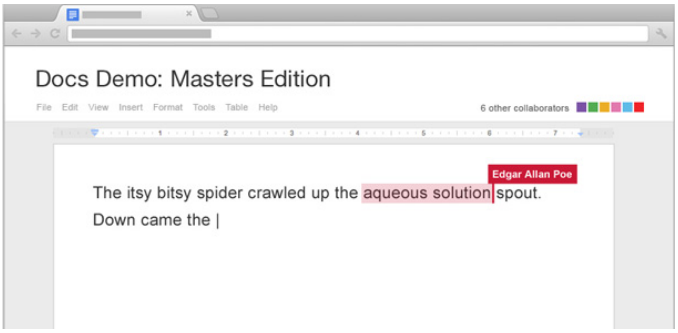

Figure 1. Users working together on a shared text document via the Internet [3] with the active highlighting of a user action on the screen

Users are able to work on a shared text canvas. The functionality is limited and focused on a very specific task so it is easy to use. The visual feedback for user interactions with the canvas is clear and visible so it is able to prevent errors and support the work flow. However it lacks a more profound skillset and if a larger amount of users tries to manipulate the document at the same time the overview and simplicity turns into a more chaotic state.

So the solution provided by Google is to reduce complexity and present focused functionality for easy handling. Different users can access the live canvas from everywhere using their own hardware (eg. notebooks, tablets, smartphones) and software solutions (eg. operating systems, browsers). The reduced complexity is also shown in the lack of any form of differential user permissions. All users are given equal rights so everybody can correct anything from anybody and everbody can use the whole existing canvas. Like mentioned above this benefits the idea of rapid content creation for a simple functionality. However the live online mode requires a permanent Internet connection and the rising potential of chaos when working with more and more users doesn't show as a fixed user limit from the start but more of an uprising limit in productivity.

3 User input management

What kind of feedback will be triggered when a user interferes with the interaction of another user? What will happen when two users try to do the same action at once? These are only 2 of the questions which come to mind immediately when thinking about multi-user input. Multiple

- *Martin Kraft is studying Media Informatics at the University of Munich, Germany, E-mail: kraftm@cip.ifi.lmu.de*
- *This research paper was written for the Media Informatics Proseminar on "The software architecture and user interface design can become complex in multi-user. multi-device, and multi-application setups", 2015.*

users, espacially in a live enviroment, produce a huge amount of data which needs to be handled correctly. The user interface elements need to support the user to understand what's going on, which changes are being made where by whom and how he can support this development. [1] A well-engineered program should present itself to active users in a more supportive role to help them finish their ideas but should also be able to protect less well trained users by offering cognitive feedback and very clear instructions for basic functions. [2] People need to understand the boundaries of the user's given permission. Am I able to manipulate other user's content? Am I able to use this feature for that purpose? Can I achieve what I wanted to create without changing elements by mistake? [7] The user interface must be presented generously and understandable. If a user interaction is triggering a series of unwanted events attached to the desired action the false input rate will become a constant danger. Especially in todays mobile Internet environment almost every user is able to be connected and informed in real time to react instantaneously via mobile devices. [7]

In the case of the live collaboration wireframing tool Lucidchart *(see Figure 2)* the complexity of user inputs is increased and it represents a more complex environment compared to the Google collaboration tool. Every user has the identical set of tools in his private instance *(see Figure 3)* and works on a canvas which is shared between all collaborators. Every user is able to manipulate any items on the canvas and add or delete any items he or she wishes to manipulate.

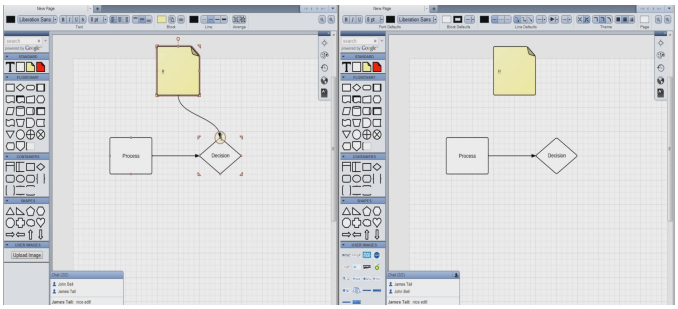

Figure 2. Adding new objects to the canvas [5] yet to be seen by another user who could potentially do something completly different in the meantime

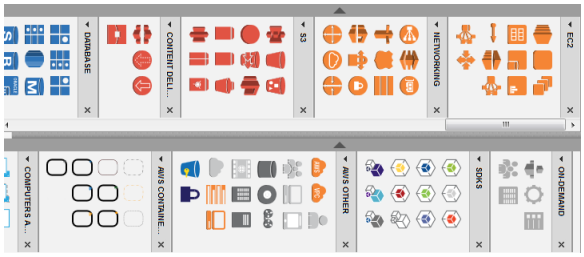

Figure 3. Complex and too similar looking functions can frustrate beginners but give expert users more defined tools at their hands. [4]

The benefit of the complete freedom ensures a quick and unproblematic collaboration experience and quick progress in the development process. The disadvantage of such uncontrolled collaboration is the lacking presence of supervision so the quality assurance is not given. Good ideas could be deleted accidently or changed without general notice and would be lost. The program offers a vast majority of complex functions and possibilities which can be too complex for beginners and the cognitive feedback lacks in visual clearness. The live collaboration helps to prevent different project states because only one up-to-date project state exists on the live canvas.

So the solution provided by Lucidchart also uses a live canvas to handle user input but offers more usable canvas space to diverse the input and offers rich functionality to give users with different skill levels diverse difficult options to create content. While the frustration potential of users is low due to the lack of permission restrictions the fact that everybody can change anything becomes a real danger for the complex content structures and the minimal optical feedback of other users input on the screen is provoking accidental misclicks and false input.

4 Version Control

The modern development process is almost never a straight line but consists of diverse and expanding developing technics (wireframing, scrum, local hill problem etc.). Smaller project sub-teams could detach and focus on a specific part of the development and need to resynchronize later in the project. The ability to create project branches and merge different project states together into a functioning complete module is a vital part in preventing errors [1], documenting progress and bringing personal progress together [2].

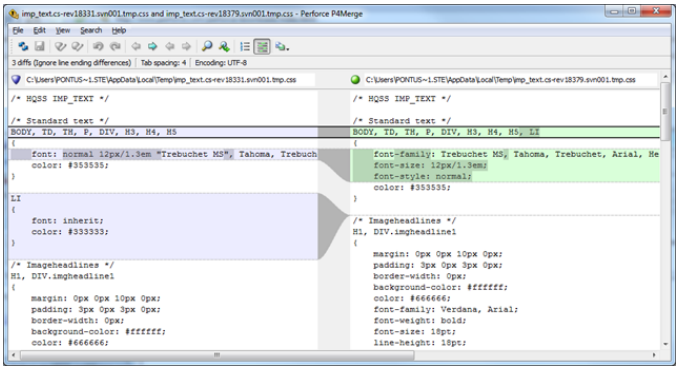

Figure 4. By showing clear visual feedback through highlighted changes the user is supported in his decision and the possibility of accidental errors is reduced. [6]

In the case of Github *(see Figure 4)*, a software collaboration program with a version control system, users are able to create copies of existing intermediate results to work on and are then able to merge improvements into the existing files without losing any progress which was achieved in the meantime. The benefits compared to simple file synchronizing services are immense. The disadvantage is a more complex system of functions and tasks which need to be handled correctly to be used.

Github (or version control in general) moves the users away from a live canvas to a more independent and private workspace but still offers the benefit of working on one big set of (synchronized) data. The potential of accidental errors or false inputs is solved by the ability to reverse earlier project stages and benefits less versatile users and larger teams of users to achieve progress without more chaotic states. Using the service is not binded to any combination of applications and devices and can be accessed with any of these eg. via an installed desktop program or Internet browser. However version control is no basic prerequisite and demands a used external service or internal build.

5 Conclusion

After looking at a few problems of multi-user, multi-device and multi-application setups and the user interface designs of the presented programs the conclusion can be called: mmm programs are very versatile and can have very different purposes besides this focused area of application. Important things to consider when designing a user interface for the discussed setup are the visualization and management of user input on a live surface, the important need to clarify functionality and purpose of tools to the user to prevent misusage and possible damage to the general progress and to help avoid frustration by guiding the user under the influence of possible user permission modes and different user skill levels.

REFERENCES

[1] E. A. Bier and S. Freeman. Mmm: a user interface architecture for shared editors on a single screen. *UIST '91 Proceedings of the 4th annual ACM symposium on User interface software and technology*, pages 79–86, 1991.

[2] C. Forlines, A. Esenther, C. Shen, and D. Wigdor. Multi-user, multi-display interaction with a single-user, single-display geospatial application. *UIST '06 Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 273–276, 2006.

[3] G. Inc. Docs demo. `http://www.google.de/imgres?imgurl=https%3A%2F%2Fwww.thinkwithgoogle.com%2Fthink%2Fimages%2Fgoogle-apps-docs-demo-masters-edition_campaigns_02.jpg&imgrefurl=https%3A%2F%2Fwww.thinkwithgoogle.com%2Fcampaigns%2Fgoogle-apps-docs-demo-masters-edition.html&h=333&w=711&tbnid=rWjv2KQ8aUWOyM%3A&zoom=1&docid=GMtWil76pQC0uM&ei=82eEVfPlLYLoywOdmaGYAQ&tbm=isch&iact=rc&uact=3&dur=252&page=2&start=38&ndsp=44&ved=0CJQCEK0DMFA`, 2015. visited 24.06.2015.

[4] Lucidchart.com. Lucidchart shapes. `https://www.lucidchart.com/documents/`, 2015. visited 24.06.2015.

[5] Lucidchart.com. Real-time collaboration - lucidchart tutorial. `https://www.youtube.com/watch?v=OINRyw7azJs`, 2015. visited 24.06.2015.

[6] C. Mode. Using a cool merge tool with svn or git. `http://www.google.de/imgres?imgurl=http%3A%2F%2Fpontusm.files.wordpress.com%2F2010%2F01%2Fp4merge2.png&imgrefurl=http%3A%2F%2Fpontusmunck.com%2F2010%2F01%2F30%2Fusing-a-cool-merge-tool-with-svn-or-git%2F&h=343&w=640&tbnid=_4tMDH9KqQVkyM%3A&zoom=1&docid=C5BzLk_P04fl9M&ei=-ueKVezxE6L8ygPc9oxg&tbm=isch&iact=rc&uact=3&dur=476&page=5&start=158&ndsp=36&ved=0CNQBEK0DMEU4ZA`, 2010. visited 24.06.2015.

[7] A. Wiethoff, T. Bauer, and S. Gehring. Investigating multi-user interactions on interactive media façades. *MAB '14 Proceedings of the 2nd Media Architecture Biennale Conference: World Cities*, pages 92–100, 2014.