Microsoft User Flow Improvements at AppDirect

Ron Hyman
Systems Design Engineering
Department of Systems Design Engineering, University of Waterloo

This report focuses on analyzing the engineering process and results of the main project during my product management internship at AppDirect in the summer of 2021. Over the span of four months, I led the design and engineering team through the discovery, design, and Agile implementation of 3 new central features, two of which were shipped into production within my time at the company. The objective of the project was to implement an improved flow for users selling Microsoft products. With Microsoft products accounting for nearly 80% of the company's sales, the engineering design process had to be well thought out. By conducting user interviews, mapping out user flows, and collecting relevant user data, I was able to thoroughly understand the user experience and its pain points. From there, designing and validating the newly created solutions with internal customers and users ensured that the improvements were complementary to the current user flow, resulting in an improved user experience. After the 3 features were shipped into production, the next step is to further confirm this hypothesis with real customers using the newly improved flow.

Keywords – product, design, software.

I. SITUATION OF CONCERN & PROJECT OBJECTIVES

As the world veers into a digital era, more and more subscription-based software as a service (SAAS) type products emerge. AppDirect is an E-Commerce platform that allows businesses to sell subscription software to their customers. AppDirect's platform can sell hundreds of software products, but the most common one is Microsoft 365, which accounts for about 80% of AppDirect's sales. Furthermore, there are two ways that companies can use AppDirect to sell products. Companies can create a self-serve marketplace that resembles an online shopping experience – just like Shopify, but instead of selling t-shirts or diapers, companies sell software. However, this accounts for only a fraction of AppDirect's sales. The other method is called assisted sales. This is a more traditional method for buying products, where there is an 'expert' sales agent on the phone who purchases the product(s) on behalf of the customer. Surprisingly, this is the most common method of purchasing subscription software products on AppDirect. This is partially because most of the purchases relate to larger businesses and an expert is often appreciated for large sales.

This background information is relevant to my project because I worked on improving the part of AppDirect's platform that agents use to sell Microsoft products. With Microsoft products accounting for the majority of all product sales, and assisted sales being the most common method of selling products, this flow is integral to AppDirect's business model. It is one of the most common cases for customers and must therefore run flawlessly.

The problem was that it did not. Functionally, the flow worked – but not optimally. The user experience was subpar and had multiple barriers for agents who rely on the software to sell products efficiently. There were misleading and repetitive errors, a form that did not allow the user to save, and multiple 'limbo' states without proper feedback that confused the user. I was responsible for analyzing the flow for pain points, creating, and validating solutions, and even implementing and shipping the new features into production. This was no simple task because by the time the project scope was defined, there were only 3 months left in my co-op term. Fortunately, I managed to efficiently define the problem, create a solution, and implement it with the engineering team. But did the newly implemented solutions improve the user flow for agents selling Microsoft products? This report will focus on this question and dive into the design methods, results, and recommendations.

II. DESIGN METHODS & ANALYSIS

The design methods and analysis of the product flow one of the most important steps in this project. This process defines the way that the pain points are identified and verified.

Understanding the flow

The first step in the engineering analysis process was to understand the flow. What does it look like? Who uses it? This was especially important for me, considering that I was a new employee who has never been exposed to AppDirect's complicated product. One cannot expect to make changes to a product that they do not fully understand. This process included a few aspects such as interviewing co-workers and customers, as well as testing out the product myself as a user. However, the most important step in this process was visually mapping out what the user flow looked like. This meant testing out all the use cases of the product flow as a user and drawing it out on a tool such as Miro. After documenting all of the use cases on Miro, I validated my analysis by presenting the visualized flow to real users. This allowed users who have used the product and understand it to confirm that my understanding of the flow is accurate. I repeated this process with multiple users until my Miro flow accurately represented the real user flow.

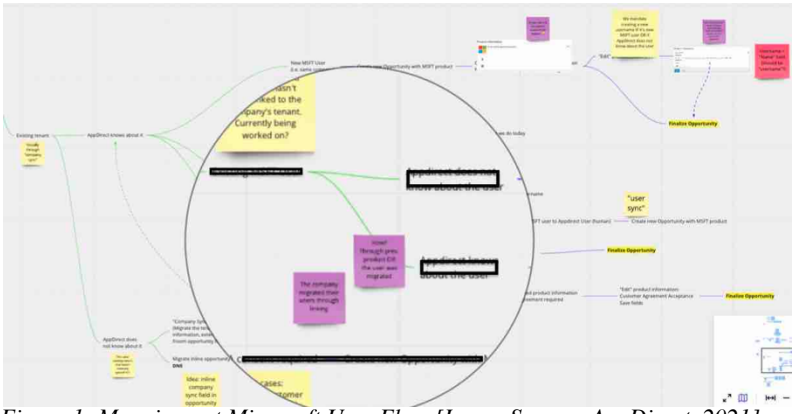

Figure 1: Mapping out Microsoft User Flow [Image Source: AppDirect, 2021]

Identifying pain points

After having a in depth understanding of what the user flow looks like, I moved on to identify the pain points that exist. This was done in two ways. First, I personally identified the 'low hanging fruit'. This meant finding the simplest and most obvious flaw in the user flow on my own, and then validating it with a real user. This gets the obvious pain points out of the way. To dive a little bit deeper, I interviewed active customers of the platform, as well as managers

that oversee the use of the platform. I interviewed these users starting with vague questions, and slowly used the visual flow to ask more specific ones. This method is what the head of product at the company calls 'Day in the Life', where the idea is to sit in the customers' shoes and try to understand their needs and pains. In fact, the product team hosts quarterly 'Day in the Life' sessions to maintain customer relationships consistently validate the team's knowledge. This helped not only identify pain points that I would not think of, but also help me understand what the users experience. For example, I learned that agents use email to keep track of potential customer information before filling out a form on the platform. This was because the form did not allow agents to save until they fill out all the fields, and thus agents had to wait until they had all the information necessary to save. This is a great example of how the product did not reflect the actual agent workflow.

Designing a solution

After identifying the most painful aspects of the user flow, the last step in the engineering analysis and design was to create a solution. With a large product management, design, and engineering team, there were many resources at hand. However, with a large team comes processes and standards. The design process revolved heavily around the UX team. The first step was to explain the problem to a senior UX member. This involved walking them through the old user flow, summarizing customer requests, and expanding on why the pain points are a priority. Once the team understood the problem, I worked with the UX designer to map out potential solutions. The chosen flow was turned into a medium fidelity mockup using InVision, which then goes through multiple iterations of feedback and edits. This process was repeated for all prioritized pain points.

III. DESIGNED SOLUTIONS

The final designs were split into three separate epics. The first epic was called 'Microsoft Partial Saving' and included a newly designed form for the user to fill. The main improvement allowed the user to fill out only some of the required fields, and still save the form. Other improvements included inline user feedback, added form functionality, and more. These improvements would not have been possible if customer interviews were not done in the design process. Mapping out the previous user flow as mentioned in the above section was evidence that the form should behave in a way that compliments the actual user journey. After implementing this solution, it was presented to real users (both internal and external customers) and was received with great anticipation.

The second epic focused on 'Microsoft Error Handling'. The idea was to transform the way field errors were handled in the flow (when an agent types the wrong input into a field). The new features provided a clear indication of which fields are missing or incorrect, something that the previous errors did not communicate. Further, visual feedback was added to indicate the validating state of the form. These improvements related to the ease of use of the form as well as clarity. To test the impact of solutions, a test subject (a fellow intern) with no prior experience using the software was asked to go through it for the first time. This was then repeated with the new solution. Although it took the user the same amount of time to complete a perfect form (~3 minutes), when an error came up, they understood what to do in the new flow, as seen in Table 1.

Although this test was not a perfect and bias free scenario, it was concluded that the new solution improved the clarity of error and how they are communicated.

Table 1: Time taken for new user to complete a form

	Flow 1 time (old)(s)	Flow 2 time (solution)(s)
Error – Free	170	182
2 errors (missing + invalid)	300	450

The third and final Epic focused on an entirely new concept. The details cannot be shared in this report, but the idea was to introduce new functionality into the already complicated product. The newly designed solution took nearly 2 months to create, and even then, was waiting approval. The feature allowed the user to complete a set of actions (syncing new company) within one flow instead of two. This is great because it would be more efficient and simplify the experience for the user at a surface level. The final mockups were in their final stages, but development did not start until after the co-op term.

IV. DESIGN VERIFICATION

As mentioned previously, the Microsoft reseller flow is one of the most important components of AppDirect's business model. The objective of this project was to create and implement an improved flow for agents selling Microsoft products. More specifically, the goal was to improve the way errors are handled and ensure the product flow is representative of the actual agent experience. Did the newly implemented solutions improve the user flow for agents? To answer this question, user testing should be conducted. The features were shipped into production only one week after my last day, so there was no opportunity to collect information and data from real customers. However, the following is what I would have recommended. Firstly, conduct A/B testing [2] on the new features, comparing the old flow to the new one. This was done briefly for the second epic as seen in table 1. This is effective but also time consuming. Another approach would be to manage and collect real customer data. Considering that the product architecture is advanced, the product team is capable of tracking and viewing customers' use of the platform with detail. An example of this can be seen in Figure 2. This includes error rate, number of customers and sales, as well as time spent on the platform. With this data, I would come up with specific metrics to track and analyze as the new features are released into production. This would help gauge the impact of the improvements. Finally, interviewing real customers a period after the features are released is a great way to collect feedback. This could even be done through a survey. With these best practices, the team could then validate and even quantify the impact of the new features on valued customers.

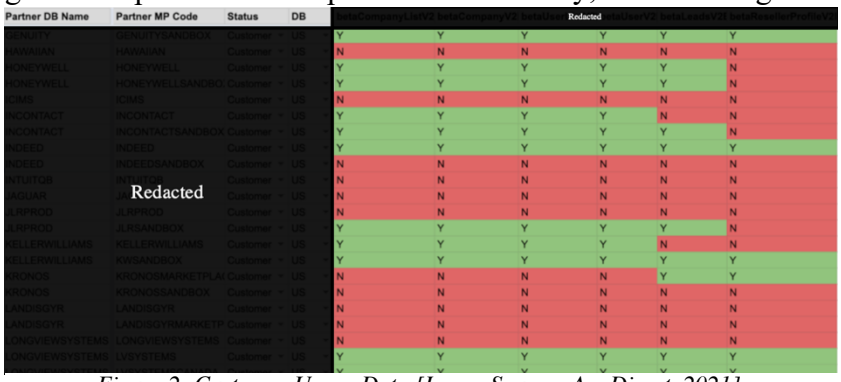

Figure 2: Customer Usage Data [Image Source: AppDirect, 2021]

V. LIMITATIONS OF METHODS USED AND DESIGNED SOLUTION

The methods used to identify the problem and design a solution allowed me to go from problem definition to shipping two new features in under 4 months. This proved to be very efficient, and as discussed in the results section, also impactful. However, every method has their pros and cons. For example, the 'Day in the Life' method was used to understand customers and identify the most highly requested features. It was used in this scenario to identify pain points that impacted both epics 1 and 3 in my project. However, it is important to note that this method is fairly structured. This means that the customer interviews and processes were ridged in terms of format and involved a lot of intervention (such as questions). The alternative method would be to observe the 'Day in the Life' of a customer without interfering at all and asking few questions throughout. The structured format allowed the team to clarify pain points and ask questions so we could properly understand the customer flow. What this can sacrifice is the number of pain points observed; interventions may alter the normal experience for a customer and some pain points and feedback might be missed. Therefore, the team conducts these sessions consistently and with some variation, to make sure that all aspects of the flow are documented.

Another method that was used and has limitations is visually mapping out user flows. This method was used to not only help me understand customer journeys, but also communicate it and validate it with other professionals. This proved to be efficient and useful, however, there is a balance between accuracy and efficiency. For example, the Microsoft flow was complex and involved multiple use cases. I spent a lot of time mapping it out for my own understanding. However, when presenting it to my colleague, they were overwhelmed with the amount of information and could not really provide feedback or validation. Overall, this method is very useful for most scenarios, but should be limited to simpler and shorter user flows. If a user flow is long and complex, it may be a good idea to break it up into smaller, simpler visualizations.

VI. CONCLUSIONS

Overall, it can be concluded that the newly implemented solutions improve the user flow for agents at a surface level, as proven by the design validations done before engineering implementation. However, more testing and research needs to be done with real customers using the shipped features to confirm that the improvements impact the customers in a positive way. This outcome is optimal considering the short duration of the project time (less than 4 months). Furthermore, future testing of the new user experience may lead to new discoveries and customer requests.

VII. RECOMMENDATIONS

Given the conclusion that more testing needs to be done, it is highly recommended that post-feature release user testing be incorporated into the feature development methodology. By ensuring that testing is done with real customers after the features are shipped, it would provide a consistent and accurate measure of the customers' satisfaction with the improvements, and whether the project has achieved its goal. Following this recommendation would increase the workload that comes with releasing new improvements which takes more time and resources (at least 8 hours more per feature). Some might say that making the release process more lengthy means that the team would be able to work on less features. Despite this, it is worth adding it to

the development process because it allows the team to quantitatively measure the results of the project, and thus understand it's impact. Furthermore, it would provide feedback from real customers, which could be incorporated into future improvements or projects. Overall, the current feature development process is well-structures, but by conducting user testing after releasing the features, it would increase the quality of deliverables.

ACKNOWLEDGMENTS

Full credit should be given to the product team at AppDirect that made it possible for me to contribute to their product and make an impact. The UX design team took full ownership of designing a solution while still collaborating and communicating with me and the PM team, making the process very smooth. The engineering team also developed the features before the deadlines while being proactive and flexible. And finally, number of my colleagues and supervisors ensured that I had all resources I needed.

REFERENCES

[1] "Microsoft CSP," AppDirect. [Online]. Available: https://www.appdirect.com/resources/glossary/microsoft-csp. [Accessed: 14-Sep-2021].

[2] "A refresher on a/b testing," Harvard Business Review, 27-Nov-2017. [Online]. Available: https://hbr.org/2017/06/a-refresher-on-ab-testing. [Accessed: 10-Sep-2021].

Images presented in this document created by the authors are indicated by author initials and the year of creation.