A User Interface Toolkit for Microsoft Mediaroom

**Project Sponsor**

Gurminder Singh and Puru Purushatham

**Faculty Coach**

Daniel Krutz

**Project Overview**

In the modern world, new technologies are popping up all the time. Some of these technologies are short lived, but others end up changing the way the world works, but it takes foresight to recognize these upcoming technologies while they are still in infancy. Currently one of the major pushes in the software industry is to unify all available digital media: Internet, Cell Phones, and TV. Netflix is a prime example because they have released their software on the Internet, on multiple Cell Phone platforms, various Tablets, and on different interactive TV platforms.

MoSoNex is developing social media-sharing software that they are distributing to their first customer SingTel who are the largest internet, phone, and TV providers in East Asia, having 383 million of clients. MoSoNex’s product is going to to be distributed on as many platforms as possible. Currently MoSoNex is planning to release their product on the Internet, iPhone, Roku Set-Top Box, and on Windows Mediaroom (for IPTV).

MoSoNex contacted RIT’s SE Department in order to get help polishing their IPTV (Windows Mediaroom) application. The purpose of the project was to identify User Interface shortcomings in the IPTV application, then propose and implement solutions to the problems in order to improve the usability of the application. The specifics about scope were kept vague because MoSoNex wanted the students at RIT to identify the problems themselves.

The goal for the first half of the project was to identify and prioritize a set of tasks to develop over the second quarter. This first task required that a lot of time devoted to learning not just the
IPTV Platform (Windows Mediaroom 2.0), but also the competing platforms: Roku Set-Top Box, GoogleTV, and Yahoo TV Widgets. The team also had the Roku implementation of MoSoNex, which was used to compare and contrast with the IPTV implementation.

The second half of the project was spent prioritizing and implementing the various tasks identified in the first half. This involved creating new screens/functionality to the product, as well as overhauling the current look-and-feel of the product. Half of the work was done with code, and the other half was done in Photoshop.

Basic Requirements

The initial goal set by the customer was that this project should improve the usability of the MoSoNex IPTV application. Through a series of meetings, the RIT team and the customer came to the agreement that this is best done in two steps. First, the RIT team should compile a usability report that identifies a set of tasks that can be done to improve the quality of the User Interface of the IPTV application, and second the tasks need to be prioritized and implemented.

The User Interface requirements were simple and straight forward. This document needed to contain the following sections:

1. Feature Comparison between the implementation of MoSoNex on the Roku Set-Top Box and the IPTV version.
2. Usability Heuristics Evaluation which analyzes every page of the IPTV application for violations of usability heuristics.
3. Application Comparison between the Roku Set-Top Box and the IPTV Version (specifically layouts, color schemes, etc).
4. Widget Comparison between the widgets in Windows Mediaroom, Roku Set-Top Box, GoogleTV, and Yahoo TV Widgets

In addition to the usability report, a document that specifies a list of tasks to improve the usability of the application was required. Each task entails a small individual unit of work that would fix one of the violations identified in the Usability Heuristics Evaluation section of the Usability report. Each task would be given a number from 1-10 representing the impact it would have on Usability, as well as a number for 1-10 representing the relative complexity of the task.

Each of the tasks was then considered a requirement for the development portion of the project. Every week we would prioritize a set of tasks to accomplish that week, and every week we would review the previous weeks changes with the customer. Tasks were broken down into sub-tasks when the original task was deemed too complex.

Constraints
There were many constraints that dramatically effected the outcome of the product. Most of the constraints were pages or themes that we were required not to change. There was never any written constraints, but the following constraints were discussed and agreed to by both parties:

- No changes to the application can be made without direct Customer Approval. There are no exceptions to this constraint.
- Color themes may not be changed.
- Label text cannot be changed.
- No new functionality can be added.
- No changes to the Friends Page, Add Friend, and Friends List pages are allowed. The one exception is button textures can be redone.

**Development Process**

Our team originally tried to use the Rational Unified Process for the project. The original project proposal suggested that we would be doing a lot of development work to build a new product for the customer. However after many meetings with the client, we were able to determine that what the client actually wanted was for the RIT team to just modify the existing client. This did not lend itself well to using RUP, so we switched instead to SCRUM.

Once the scope and requirements were agreed upon, we realized that the first half of project was not doing software development but just domain research and extensive requirements gathering by building a thorough User Interface Analysis report. Using this report, we built up a large set of User Stories that identified tasks to improve the usability of the product.

Looking back on the project SCRUM would have been the correct choice to begin with. Since the switch to SCRUM our productivity has increased and we have been better able to manage the tasks using the process.

The weekly meetings associated with SCRUM were the perfect time to meet with the sponsor and our faculty coach. We combine our kickoff meeting with our wrapup meeting so our sponsor was always in the loop with the work we were doing and was able to help when we needed and give advice on situations that arose that we may not have considered.

In addition to weekly SCRUM meetings we also had weekly meetings with our sponsor to help us prioritize our tasks to best suit the customers needs. We wanted to integrate the customer into this project the best we could to make sure that the customer was getting the product they wanted. To do this we proposed a list of tasks with priorities already associated with them and had the customer review them. The customer gave that list back with some changes and then we planned out a timeline because some tasks were dependant on others and we did our best to make sure the customers high priority items were taken care of first.

**Project Schedule: Planned and Actual**
The project was planned out in 1 week sprints. We planned each sprint at the beginning of the week based on what the sponsor felt was most important to get done. See the metrics section for more details on the work we did each sprint. The actual schedule seemed to go well with the exception of a couple features running over their individual deadlines but the features got implemented.

We did run into a couple issues regarding a 3D border the customer wanted as well as a button association that customer wanted. We agreed to do these tasks and planned them out but after looking into them with the system the way it was, it was not realistic for us to implement these items within the time frame. Instead of just tossing out the tasks we spoke with the sponsor about another possible solution; we wrote up technical documents to describe the issues we faced and what we tried to do to get these items to work and delivered those documents to the customer with an outline of a potential solution.

We planned on many tasks running over since this technology was knew to us to we had a mitigation strategy in place here to make sure we could handle this. To try to help resolve this potential issue we spoke with the sponsor about starting out with smaller tasks that may not have been high priority so we could get our feet with the new technology. That would give us smaller hurdles to overcome and help adjust us to working with the system. Also, during sprint 1 we were waiting on the sponsor to get us some materials so we could begin development so we didn’t really have work to do regarding development so decided that each of us would write a GUI based application that would have similar base functionality as the MoSoNex application; again, this helped us get adjusted to the new environment.

System Design

Design of original system
The original codebase provided to the team was designed for the application's immediate functionality, but with less concern for addition or alteration of features. The application was designed for a linear application flow, with History1.aspx (the main page) functioning as a 'home loop'. A handful of pages act as functional resources, and handle all web calls, search processing, and other functional processing.

Though the implementation is not strictly or properly implemented, the application does follow the MVC mentality in its overall design. The vast majority of the data used by the application is held elsewhere, with only a single XML-style file holding local preference and setting information. These functional pages, which had little to no display logic, operated as the controllers of the application and were the primary interface for the view pages to interact with the data. This interaction functionality was outside of the usability analysis definition of the project as it was given, and was deemed to be sufficiently functional in the state it was given to the team, therefore we did not give it a considerable amount of attention.

Although the functionality was at a usable level, we found several issues regarding code repetition across similar functionalities. Multiple pages did extension swapping; the implementation of this function was done using manual case checking (producing nearly a dozen lines for each extension as all combinations of upper and lower case were explicitly
stated) and was copied directly into each page's codebehind rather than held within a widely accessible method. These practices generally did not impact the system's usability, but the team had intended to improve these elements over the course of the project.

**Design of document**
The Analysis report was separated into four sections- Features, Heuristics, App Comparison, and Feature Comparison. The team was informed by the project sponsors that user interviews and related usability investigation had already been done, and thus the Analysis should in the team's estimation focus on a comparison of the application against other similar and dissimilar applications. The Mediaroom application was compared against the MoSoNex Roku application, the default interface of Boxee, and the interface of GoogleTV. The sections were selected based on the team's history with usability analysis approaches and the knowledge that the project sponsors had concerns over the program's functionality level and the implementation of those features based on their requests for new features.

Each section was filled with as much relevant content as possible based on what exists within the Mediaroom and Roku applications. Features investigated every feature present within Mediaroom or Roku applications and how they act. Heuristics looked at every page within the Mediaroom application, including pages that have minimal functionality. App and Feature Comparison sections looked at what Mediaroom and Roku contained in terms of features and implementations, but also looked at what other applications and systems offered that had no equivalent on Mediaroom or Roku.

**Design of code changes**
We did not find any compelling reason to alter the system's MVC-style implementation. Any refactoring or rewriting of the application would have been done using the same design paradigm, but altered to more strictly adhere to the definitions of MVC and to improve general coding practices. We unanimously felt that MVC is the most suitable architectural design for a web-style application (and it is the most common in the industry, with technologies such as Ruby on Rails requiring such an architecture) and thus did not give considerable attention to other architectures regardless of their potential suitability.

However, because of deadline restrictions and the wishes of the maintaining development team, plans to design and implement a refactoring of the system were not pursued. Additions to the code in existing pages were implemented in-line, without any intentional changes to the system's overall design. New functionality within a page was implemented through new functions within the codebehind files, then called during the page's normal execution flow at a stage where similar activities are executed based on the original design.

The two new features (the Media Grid and dedicated Slideshow) were intended to be placed within the overall design such that calls to the original Image Display page would instead become calls to the Media Grid, with the Media Grid being the only means of accessing the Slideshow page. This direction was taken because each application flow that went to the Image Display page was either directed by the Display page itself (an action which was replicated by the Media Grid) or by a search action that is intended to produce a set of images as a result. The new features were developed by breaking apart the existing Image Display page into its components which controlled thumbnail retrieval and display and its components which
controlled the slideshow elements. These elements were interwoven in the code due to the highly coupled nature of the existing code; however, the elements were also sufficiently separate that they can be and were broken apart without requiring significant rewriting. The code was not largely refactored outside of what was deemed necessary. The timeline allowed the potential for these elements to be rewritten with an altered design, but due to other restrictions the benefits were deemed to be too minor for implementation.

The new graphics were designed and titled based on a freshly-designed template image. The template was meant to closely, yet not precisely, resemble the existing button graphics in the application, with the intent that the usability research done by MoSoNex that led to these button designs would not be lost in the recreation. The alternate state graphics (hover, click, alpha, etc) were considerably different from the original alternate state graphics but were designed and titled so that they better represented their purpose and provided the same visual effects, improving only the readability of the buttons.

**Process and Product Metrics**

The team used the sprint burndown to track the progress of the project. This was put together in an excel sheet and we kept track of project velocity as well. Our average velocity for the project was 27 points per sprint.

Overall The project went smoothly with regards to getting the work done. The burndown charts show that we completed nearly all of our tasks or at the very least resolved all of our tasks. We have two burndown charts, one that is "ideal" and one that is "real". The "real" chart shows the work that actually got completed and was integrated into the project. Unfortunately there were two tasks that were too much to complete in the time frame. To resolve these issues we spoke with the customer and came up with a middle ground on them which was documenting a proper solution and delivering that. However, since those tasks were not completed by the standards we set for ourselves we have taken those tasks off of the completed list for. Since those tasks were resolved to some degree we also have a burndown chart that shows those as completed.

We also have charts showing points per task, which averaged out to be 6 points/task, the points we attempted per sprint and the number of tasks we attempted per sprint. The latter two charts helped show us sprints where we really didn’t make much progress on the project such as sprint 3 where we had 1 task and we completed 0 points. Being able to look back at these items helped us gauge if we were on the right track to completing the project because we knew how far we had to go so we could gauge what our velocity should roughly be on a weekly basis.

Overall these metrics would have been most valuable to a team that would be working together again since the metrics are meant to be viewed at the end of the project more-so than during the project. The biggest help we received from these metrics was building the foundation to track them. When putting together the spreadsheet we began looking at how things needed to work in order for us to complete the project on time and we began planning things out.

Charts are collected at the end of this document. Data is available within the project artifact collection.
Product State at Time of Delivery

The product is in a usable state as promised to the customer. We were able to implement nearly all of the features promised to the customer and we achieved all of the features that the sponsor initially requested. Unfortunately, one feature that the sponsor requested was not implemented due to time constraints and a lack of understanding on the teams part of the depth of the feature.

Usability Report

The customer asked for a User Interface analysis document. The RIT team went above and beyond this task and created a fully comprehensive Usability Analysis that analyzed the product from every possible angle. The recommendations we provided were based on standard usability heuristics, and our list of application improvement tasks proved to be exactly what the customer wanted. The Usability Report was delivered at the end of the first quarter. The customer originally wanted the document to be delivered in the first half of the quarter, but due to a lack of support, we were unable to get started with the analysis (we were not given the product until 5 weeks into the quarter).

User Interface Development

We accomplished almost all the tasks on the list that we created in the Usability Report. For the tasks that we were unable to accomplish, we gave the user an in-depth report on as to why the task cannot be accomplished. Specifically, we are referring to the task about the 3d Border to add to the image on the Main Page. Due to the limitations of the back end code (which we were not permitted to modify), we were unable to create a dynamic border. The customer was well aware of the difficulties we had with this task and was not surprised when we finally said that we could not accomplish the task. We went down 5 different possible implementation ideas, but each proved to be impossible due to either limitations on the platform or limitations from the customer. All the ideas were documented and submitted to the customer. In addition, we determined what backend changes were necessary in order to fix the problem, and we submitted the solution to the customer so that they could finish the task with their engineers.

Project Reflection

What went right?

The team successfully delivered a product that satisfied the customer. We were very successful in differentiating between what the customer asked for, and what they actually wanted. As a team we investigated many possible solutions to the problems that the customer was having
with their product, and the recommendations we made to them were solidly based on our education here at RIT.

We were also fairly successful at self-analysis. We periodically analyzed what we were doing, and how we could be doing it better. We talked often with our faculty coach in order to make sure we were always on track. We were given a relatively easy project, but had many huge obstacles that were almost impossible to overcome. As a result, our team was often incredibly frustrated with the customer and with the product itself. We feel personally that we did a great job in managing all these frustrations.

What went wrong?

Communication was a big failure for our project. Our customer lacked the desire to fully support our project, and as a result we ended up having tons of problems related to communication. For example, we asked once a week for 8 weeks to get the documentation (API) for Windows Mediaroom 2.0. The customer kept promising to give us the documentation, then they told us they didn’t have it, and finally after 8 weeks they gave us the document. This set us back almost an entire quarter, and we feel was completely unacceptable.

Communication also ended up in a lot of wasted time in coding. The team and the customer would work together every week to pick a list of tasks to accomplish. On one occasion, the RIT team was working on a task, and after two weeks of work, finished the task, except were unable to test the product due to not being on MoSoNex’s internal network in Singapore. The customer then determined it would be too much work on their side to open up access for testing, and that the RIT team should just throw away the new code/task.

We had a very rocky start at the beginnning of our project. We felt really lost and unsure of the direction we should head. We were originally tasked with “learning Windows Mediaroom”, and we took several wrong turns along the way there. Originally we waited to get the documentation from the customer. After 3 weeks of wasted time, we asked one of the developers at MoSoNex to recommend a way to get started. He recommended we learn C# and Windows Presentation Framework. After 3 weeks of studying this, we learned that it has virtually no reflection on the work we need to do with the product. Finally, in the 8th week we got the documentation and were able to start developing on the project. This initial churn was a major hurdle to overcome in the second quarter.

These types of problems ended up causing the RIT team much frustration and wasted time. As a result, we were not able to accomplish everything that we wanted to accomplish. We felt very insulted by the lack of effort/support given by the customer.

What would you do differently in the future?

Unanimously, our team agreed that in the future we would all be much more insistent with the customer. Our original approach was to kindly ask for support, and just work around any
resistence the customer applied. For example, instead of demanding the documentation, the RIT team just submitted a new request every week for it. In the future, we would write out an email earlier on explaining how we cannot work without the documentation, that it is an absolutely necessity, and that the customer must get us the required documents by the end of the week. Any pushback from the customer, and we should have gone to our project sponsor or the department head to report that the customer was being uncoorporative with us.

We feel that many of the problems we encountered were due to the unprofessionalism of the customer. As a team, we feel that in the real world, a customer would not be so unwilling to work if they were actually paying for the product.

Appendix: Charts
Points Attempted/Sprint

Sprint