Facets Mobile  
Team 6g  
Bryan Sullivan  Victor Calderon  Robert Van Tyne  
Ryan Chadwick  Mark Gatesman  

Project Sponsor  
Dr. Edward Hensel  

Faculty Mentor  
Dr. Stephanie Ludi  

Project Overview  

The purpose of this senior project is to take as many of the currently implemented tools in the FACETs web system and port them to a mobile device – specifically the Droid line of phones. The EDGE system, which stands for Engineering Design Guide and Environment, is a web system used by the RIT Engineering department to manage engineering senior projects, among other things. It is primarily a document version control system and project management system. It allows you to create projects, create users and assign users to user groups to provide security, and then within projects manage project documents. A subset of this is FACETs, which stands for Formulate, Assume, Chart, Execute, and Test, with the s referring to the iterative nature of development.  

Previous SE senior project teams have built 5 different tools to help engineers go through the FACETs development process. They are Brainstorming, Affinity Diagram, Objective Tree, Function Tree and House of Quality. The ultimate goal of our project was to take the tools that were made on the web, and make a version that works on the phone. Due to time constraints, we were only able to accomplish moving two tools: Brainstorming and Affinity Diagram.  

The other decision that had to be made was whether or not to make a native Android application – Android is the operating system that the Droid devices run on – or a mobile website. The decision was to make a native application because it allowed for far more feature rich content, which would make the application more useful given what it would have to do.  

One of the major requirements was that any tool that we made – be it a mobile website or a native application – would have to work with the existing system. This meant that while adding features to the current web system was out of scope, fixing any bugs in the web system that prevented users from being able to use both the mobile version and the web version simultaneously would need to be fixed.  

The result of the project will be an Android application for both the Brainstorming and Affinity Diagram tools that work with the current tools on the website. Along with the tools themselves, there will be Session Management functionality – that is the ability to create and edit sessions for both Brainstorming and Affinity Diagram. There are some
tool-specific options, such as a vote limit in Brainstorming and the stakeholder groups in Affinity Diagram. Other shared functionality – such as RIT LDAP authentication and the database connection – will also be completed.

**Basic Requirements**

The overall requirements of the system are to create an environment for users to participate in using tools of the FACETs system. Currently there exists a website with several tools implemented. The goal of this project is to take existing tools and convert them to Android applications. This will allow users to be free from the constraints of their desktop computers as well as being in the same room as the rest of the participants.

The Android application that was developed needed to support the existing FACETs tools so users could participate no matter which platform they were using. Since the web tools supported real-time concurrent collaboration that was the main priority for the Android application as well. The Android application needed to support the number of users in a normal classroom environment. Another major requirement was to keep track of all inputs to the system. The system is used for document management and all user entered data needed to be saved for future use. After all inputs have been entered, the system needed to display the results to its users. This was done on both the Android application as well as in the existing website.

The project also needed to follow the common principals that are designed for creating Android applications. There are many design guidelines that are found on the Android development website that needed to be used so users of the application have an intuition on how to use it.

All other specific application requirements are outlined in the Software Requirements Specification document.

**Constraints**

The main constraint on the system is resources. As described in the overview, the original project specification was to complete all 5 tools on a mobile device. Due to time constraints, we were only able to actually create 2. In addition, time constraints limited us in the amount of testing and bug fixing we could do. While the delivered system is in a very good state, there are a couple bugs and minor features that were not implemented as well.

The Android OS was a technology constraint in limiting what we could use for database and server technology. Due to Android not supporting Remote Method Invocation (RMI), we decided to use Java Servlets, with the connection to the server using http requests. Another advantage of using servlets is that it is very easy for other GUI implementations (such as a website) to also use the same data structure, if that were ever considered in the future.

The Android OS was also limiting in the fact that we had to use their packages for UI components and how the components worked together. A limitation that caused no
implementations problems but is a maintenance headache is that you cannot organize the layout and menu xml files into folders. This caused the single folder to become extremely large and annoying to deal with, and will continue to not only be annoying to future teams, but will grow as well as the other tools are implemented.

The limitation of our knowledge of Java and cryptography prevented the user from being able to use the same superuser access that is encrypted in the .htpasswd file on the server. While Java can do an MD5 hash, the htpasswd command on the server uses a salt. It’s essentially an extra key, and what it does is allow the same text to be encrypted in different ways. Without knowing what the salt is, the encryption couldn’t be duplicated. Also, the encryption is one-way, meaning it can’t be decrypted either. Verification is done by encrypting and comparing the hashes.

The major constraint on the system implementation was that because all the phones would be constantly polling the database for updates in several of the phases, that a large database and/or server connection bottleneck would occur. By using a database pooling system, as well as making a single database call and then the phones accessing the result of the database call, reduce the overhead to manageable levels. However, large numbers of users simultaneously will cause issues as the server gets overloaded.

**Development Process**

We initially chose the spiral model for a couple of reasons. The first being other teams in the past have chosen it, and the second for its primary focus on risk. In the end this turned out to be a bad decision as our schedule and mindset was in line with iterations. When it came down to what we wanted to do, it was clear iterations were the way to go. We wanted to use heavy prototyping to avoid rework, and deliver multiple releases to get continuous feedback from our sponsor. Looking back I think this was a good choice as it increased the satisfaction of our sponsor. Iterations also allowed us to divide the project up much easier. Our sponsor approved the change to an iterative process when we explained our motivation for wanting it.

Our teams schedule has us releasing something every week which was nice because we were able to show this functionality to our sponsor. It allowed a regular schedule for our sponsor and provided him a means to know we are actively working on the project. Continuous feedback is always useful because if any changed to come up we are able to address them quickly. If any rework needs to be done then that can be completed for the next iteration.

Based on this iterative process we were able to create the project schedules. We then created teams for each portion and divided up the project. Team members then bid on their preferred project. This allowed team members to get to work on what they wanted to work on and seemed to increase project satisfaction. Brainstorm, for example, was divided up as follows:
**Iteration 1: System Prototype**

In order to ensure that all elicited requirements match the customer’s needs and expectations of the final product, a mockup system prototype will be created. This prototype will not contain any functionality, but will consist of screen mockups of the Idea Generation, Idea Grouping, Voting, and Results phases so the customer can see a close representation of the final product as early in development as possible. This will be done to minimize developer time wasted on unnecessary features, as well as minimizing design rework to implement missed requirements.

**Deliverable Content**
- System Prototype Mockup Screens
- Planned Features and User Stories, each mapped to a mockup screen

**Iteration 2 – Idea Generation Implementation**

Iteration 2 will be the first iteration to provide a functional application that can be demoed to the project sponsor and undergo usability testing. The application will not authenticate users nor connect to the FACETs Brainstorming tool database, but function stubs will be made to “fake” that functionality to allow for testing and presentation.

**Deliverable Content**
- An installable Android application that contains the following features:
  - Creating a new Brainstorming Session (only in on-phone memory)
  - Editing a Brainstorming Session
  - Participating in the Idea Generation phase for a chosen Brainstorming Session

**Iteration 3 – Voting Implementation**

Similar to Iteration 2, Iteration 3 will contain a functional application that can be demoed to the project sponsor and undergo usability testing. In addition to the Idea Generation functionality provided in Iteration 2, Iteration 3 will allow users to participate in the Voting phase of a Brainstorming Session. Sorting similar ideas into groups will not be included in this iteration, but the system may automatically create groups to allow for thorough Voting phase testing, and ease integration of future Grouping phase development. Partial database or credential functionality may exist in this iteration release based on the speed of “Session Management” development. The application will continue to use function stubs to “fake” any remaining database and credential functionality yet to be developed.

**Deliverable Content**
- Creating a new Brainstorming Session (only in on-phone memory)
- Editing a Brainstorming Session
- Participating in the Idea Generation phase for a chosen Brainstorming Session
- Participating in the Voting phase for a chosen Brainstorming Session.
Iteration 4 – Full Implementation
Iteration 4 will contain all features planned for the end release of the product. Idea Generation, Grouping, Voting, and Results Viewing will all be included in this iteration. The user will also be able to login using their RIT DCE account and access, create, and edit Brainstorming Sessions in the EDGE database. This Iteration will undergo thorough integration and usability testing.

Deliverable content
• Creating a new Brainstorming Session
• Editing a Brainstorming Session
• Participating in the Idea Generation phase for a chosen Brainstorming Session
• Performing the Grouping phase activities for a chosen Brainstorming Session (Moderators Only)
• Participating in the Voting phase for a chosen Brainstorming Session.
• Viewing the Results of a completed Brainstorming Session
• Shifting between adjacent phases of a Brainstorming Session (Moderators Only)

Iteration 5 – Final Release
Iteration 5 will contain a fully featured, tested, and documented Android application package. This will be considered the first “Product Release” of the project intended for end users. No new features will be added between Iteration 4 and Iteration 5, but any defects and usability issues will be corrected.

Deliverable Content
• Contains all features planned for the release of the product
• Is digitally signed with a certificate that expires after the expected lifetime of the product
• Includes help documentation with information about application installation and usage
• All bugs found in Iteration 4 are fixed or otherwise mitigated
• All usability issues found in Iteration 4 are fixed or otherwise mitigated
Project Schedule: Planned and Actual

During the first few meetings we spent a great deal of time determining what we would deliver to our sponsor. We knew we would need some sort of management system for the FACET states so we were left with determining how many FACET tools we would have time to implement. Our initial estimates seemed high but we reached a decision of implementing two tools. This decision came from discussions with our sponsor about the average number of tools completed by previous teams during their senior project. Based on this result and RIT’s quarter system, we were able to create our bulk release dates. Brainstorm would be delivered by end of quarter one and Affinity Diagram would be delivered by end of quarter two. This division made it trivial to create milestones dates and focus on each project individually and left the summer break if we needed it.

The major portion of this first release was the session management functionality. Key activities for this include LDAP verification, session selection, and session creation. These activities were designed to be completed in parallel to the Brainstorm application. Key activities for Brainstorm include idea generation, voting, grouping, and results. We had spent the first three weeks learning the Android operating system so we had some means to estimate the total length of development. We believed that when the Brainstorm portion was complete, the initial session management functionality would be complete as well. We divided up the team where two people were working on session management and four people on Brainstorm. Based on this we created the schedule and estimated development would be done by week seven.

We met our schedule for development of Brainstorm but severely underestimated the testing and defect resolution. We had scheduled two weeks for defect resolution but this lasted over the break. We originally had left the summer break open, meaning if we needed to then we would work so this did not impact our schedule.

When our team met to discuss the second quarter schedule we knew we wanted to be done by week seven to allow time for our poster and presentations. We unfortunately had a team member who could not continue so at this point we were down to five members. We took this into account and adjusted the schedule accordingly. Key activities for Affinity Diagram were needs generation, grouping, chat, and sentence creation. Along with Affinity Diagram we had to finish session management and finish any remaining bugs in the system. We spent the first week learning about Affinity Diagram, gathering requirements, and creating prototypes. We split up the project in a similar way as the first quarter. One person worked on session management, and the other four people worked on Affinity Diagram. We missed the deadline for this project by a week and a half. This was probably due to poor estimation for the refactoring. We had some extra time set aside in the beginning for poster and presentation work so this did not affect us too much. It would have been better to get a final demo for our sponsor sooner but in the end he was satisfied with the tool.
System Design
The team decided on a client/server architecture for this project. The data we were supposed to manipulate and represent was already housed in a central location for use in other projects. This data also had to be synchronized on both platforms (the Android and the Web). Since the existing website was implemented in PHP and directly manipulated data we decided to write our own database functionality for manipulation by the phone. We added very little new functionality to the database. The functionality we did add only affected the phone implementation and therefore no change to the website was needed.

Client
In order to help understand our design we must first explore a few basic Android design guidelines. All user interaction is based around a concept called an Activity. Each Activity represents a specific context or state which defines the way a user can interact with the program. It then becomes obvious that when the designer wants to change the way the user interacts with the current context then it would warrant a new Activity. This correlates directly to the way that each Android based program is designed. Each screen is a representative Activity in theory and therefore should extend the base Activity class.

With the above in mind we decided to take a modular design for our views. The team designed and built each activity in order to suit the linear nature of our program. The program flow was meant to narrow down what session the user wanted to access. To move forward in this progression we merely needed to swap in the Activity that asked more specific questions about the desired session. This can be seen in the diagram below (Figure 1).

![Diagram](image)

Figure 1
Each part of the diagram in Figure 1 from Login to Session Selection Represents an Activity that will help the program logic decide which session and phase it should display to the user. Each of these is encapsulated in its own Activity class that houses logic and view for that screen only. Essentially what this comes down to is a state based system that changes with user interaction.

The first four activities (Login, Project Selection, Tool Selection, and Session Selection) make up the first part of our system: Session Management. This part of the system makes up all the functionality inherent in the EDGE project management system that FACETs requires. This includes user authentication and session management as well as navigation to any specific session type in FACETs.

Once we had the groundwork for session management in place we could start on the tools for FACETs itself. We decided again on the same modular approach for the FACETs tools. Each tool represents a process like Brainstorming that we would use to refine products or ideas. Each process can be broken down into steps which define what a person should do in that part of the process. This again correlates well to the idea of Activities. Each step can be represented by an Activity class. As the user progresses through the steps we merely need to change the Activity in order to change how the user interacts with the data.

**Server**

The team chose to use JSP in order to access a SQL database housed on the ME departments EDGE/FACETs server. Our server design is subdivided into two parts. They are: HTTPS servlets for handling requests and database “Apps” that interact with the data.

![Figure 2](image_url)

The servlets are used to handle HTTPS requests and responses to the Client. There is a servlet implementation for each FACETs tool. This implementation defines what communication is necessary to interact with the server. When a request is received the
servlet identifies if it is a valid request and then starts the corresponding operation in the form of a database App.

The databases Apps encapsulate all interaction with the database itself. There is a database App for each FACETs tool. The Apps consist of data interaction like updates, inserts, and removals based on requests coming through the servlets. They also spawn and despawn threads that cache lists of data for each FACETs tool in order to keep the number of databases accesses down when sending data to multiple clients.

**Process and Product Metrics**

The two main metrics our team planned to measure were estimation accuracy and requirements density. Initially we also planned on measuring requirement-to-release cycle time. In the beginning of the first quarter, we thought that we would write more than two tools during the two quarters, so cycle time would show how long it would take to release each tool. However, we eventually settled on only making two tools, and focusing on one tool per quarter, so the cycle time metric was dropped.

**Estimation Accuracy**

Each week, team members independently filled out an activity tracker for that week’s tasks. At the beginning of the week, each team member estimated the amount of hours needed for each task, and at the end of the week would enter the actual hours spent. Meeting times, testing, and presentations were also entered. The following is the sum of all estimations and actual hours spent of team members for each quarter.

<table>
<thead>
<tr>
<th>Quarter 1</th>
<th>Estimated: 541</th>
<th>Actual: 564</th>
<th>Difference: + 4.1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quarter 2</td>
<td>Estimated: 498</td>
<td>Actual: 482</td>
<td>Difference: - 3.3%</td>
</tr>
</tbody>
</table>

Note: For ease of comparison between Quarter 1 and Quarter 2, Arnold King's estimated and actual hours were removed. For reference, the full team of 6 values for Quarter 1 are as follows:

| Quarter 1 (With Arnold) | Estimated: 623.5 | Actual: 613.2 |

According to the data, we were very accurate in our estimations for man hours spent per week. We also underestimated in the first quarter and overestimated in the second quarter. The 7.4% swing in estimation is likely from overcompensating while attempting
to more accurately estimate how long features would take to implement. We also concluded that this difference is from the team getting more adept in programming for the Android OS - much less research was required when adding functionality.

**Requirements Density**

As a product metric, requirements density helped to keep track of code conciseness. An increase in requirements density would imply that we are adding more requirements without adding much code, which is favorable. If the requirements density falls significantly, we would have to investigate the cause and decide if taking action, such as refactoring to reduce code bloat, is necessary.

**Quarter 1**
- Lines of Code: 6.9K
- Requirements: 62
- Requirements Density: 8.99 Requirements / KLOC
- Lines of Code per FACETs Tool: 6.9K

**Quarter 2**
- Lines of Code: 13.1K
- Requirements: 86
- Requirements Density: 6.55 Requirements / KLOC
- Lines of Code per FACETs Tool: 6.55K

When briefly looking at the changes in requirements density from the first quarter to the next, one would assume that code bloat increased over the duration of the second quarter. The requirements density changes from roughly 9 requirements per thousand lines of code down to about 6.5.

When the team realized that the requirements density was steadily dropping, we briefly investigated the causes. After discussion, we attributed the change to the fact that 21 of the requirements are UI specifications, and 26 of the requirements are shared between tools - more than half of the requirements are shared between Brainstorm and Affinity Diagram tools. This means that, while we nearly doubled the code size to add Affinity Diagram, many of the requirements were simply ensuring that the Affinity Diagram tool was following the same constraints as the Brainstorming tool. Seeing that there are 20 Brainstorm specific requirements and 19 Affinity Diagram requirements, the lines of code per tool-specific-requirement stayed roughly the same.

While code bloat did not increase from the first quarter to the next, the fact that the lines of code doubled when adding another tool showed us that we did not have enough code shared between tools as we should have. The lines of code per FACETs tool decreased only by 350 when adding another tool. All of the session management code was shared between both tools, but all of the adapters and activities for all 4 phases of both tools were mutually exclusive. The team has attributed this weakness to initial lack of understanding of Android OS programming during design phases.
Product State at Time of Delivery

All major functionality stated at conception has at time of delivery has been completed. We completed all three major functional points of the project: Session Management, Brainstorming, and Affinity Diagram. The platform for session management is capable of all functionality that is present at this time on the website. The two tools tasked to us for completion in Brainstorming and Affinity Diagram have both been completed and work in unison with the current website.

There was only one function we were unable to complete before the end of the project, the super user password. We were to add a super user password access based on the root access of the EDGE server but we were unable to finish this feature before delivery.

We did some testing with our sponsor’s students. Afterwards, it became apparent that not all functionality was intuitive in use. We added an extra help feature that was not planned that explains each screen.

Currently we have deployed the FACETs mobile server we created to the development machine designedge.rit.edu. We have created several deployment scripts that can be utilized to move the server over to the production server edge.rit.edu.

Project Reflection

Overall the project went very well. We were able to finish with a working product. Although there are a few minor issues the project is in working order and can be used by current mechanical engineering students when they are using the FACETs tools throughout their classroom instruction.

The team was able to effectively communicate through email, phone calls, and face-to-face interaction. Most of the issues that we had were resolved in our weekly meetings when everyone was able to contribute and track down the problems. Everyone was well focused and able to complete their assignments on time. There was a high level of organization and determination to get the project done right and working on time. If anyone needed help to get one of their tasks done there was always someone who was willing to step up so the project could move forward.

Having a team of 5 people was the perfect amount for this project. The tools that we worked on logically broke up very well into 5 separate pieces so everyone was able to develop their part and integrate it with the others. This allowed us to develop the application phases all in parallel. This also allowed us to give responsibility to specific parts of the application.

There wasn’t much that went wrong throughout the project. There were things that we could have improved on if we had more time but that is always the case. The impact of losing a team member only meant that the rest of us had to completely focus on development throughout the project. This left less time to be able to effectively manage the documents that are necessary for a successful product. If we were to have the 6th member of our team we would have been able to keep the document more up to date. This just meant that we had to do more work towards the end of our project.
As a team we came up with a set of Metrics at the beginning of our project. We were able to calculate them but we might have been able to choose better ones once we changed our process after the initial few weeks. The metrics we chose did not require too much extra effort throughout our project. We had weekly activity trackers to keep track of the progress, effort, and time spent on the project. If we had broken down the activities more we might have been able to generate more meaningful statistics.

If we had to do this project over again it would be very beneficial for everyone on the team to have experience with the Android phone. Playing around with the phone for a week or so greatly increases the user’s knowledge of how the phone behaves, and what its users expect when they are using an application. This could have considerable design impacts as to how to best capture the user’s interaction with the device. Another thing that we should have all done was go through and do the tutorials that are provided by the Android developer’s website. This would give the team a quick overview of how to develop an Android application and jump start the team to get developing quicker. At first it seemed to have a high learning curve, but once you jump in and understand what the system provides for you it becomes very easy to pick up. Another thing we could have.

As a team we feel that we were able to effectively create an application for mobile devices that have small real estate as far as the viewable area is concerned. This was the main driving factor in our decisions on how the users would perform the same functions they would on the website. We learned a little more on how to design the program and take into account many different usability concerns. Another thing we learned was how to make an extensible and maintainable system. Since this project will be used across many future senior project teams, and their continuing effort to the application, we needed to create a system that they could just pick up from where we left off. This was a learning process and we spent a considerable amount of time debating the consequences of different approaches. In the end we came up with a system that would be familiar with students pursuing software engineering. This will allow for the project to continue and be successful in the long-run.

**References**

1. Android Developers – developer.android.com