Project Overview

The FACETs project was created by the Mechanical Engineering Department Head Dr. Hensel, and is currently in its second stage of development. FACETs is built upon the existing EDGE system, which is a web-based design environment currently used by students within the multi-disciplinary senior design course, but has the goal of becoming an open source system used by any design team.

FACETs role in this system is to provide the design teams with development tools, facilitated by the FACETs design methodology created by Dr. Hensel. This methodology
focuses on twelve stages of product development that can be universally accepted by any engineering practice. Each stage of this process is represented by a toolbox, and each of these toolboxes will provide tools that design teams can use to aid with a team's process at any given time during development.

Our project will be the second version of FACETs, and will focus on restructuring and renovating the existing version to allow FACETs to run with the live EDGE system. In order to be usable on the live server, tools that are currently implemented on the system will be more thoroughly tested through various live user test sessions. Based on the user feedback, the tools will be modified to allow for better usability. In addition, new tools will be developed as well to foster the growth of the overall system.

Basic Requirements
Our project focused primarily on renovating the existing structure of FACETs to allow information to be shared between tools. Tools are placed into toolboxes, which are named after the twelve stages in the FACETs design methodology. Each tool allows a user to create a session, where a session is given a name, a problem state, a start date, an end date, as well as a moderator. A session is uniquely stored with this information within the system. A created session may also be modified at any time, including modifying the times to reopen a closed session, changing moderators, and changing the name and problem statement. While that session is open, users with the proper privileges may join the active session and use the tool.

Individual tools will vary with regards to the kind of information that is input. Many of the tools in the various toolboxes involve inputting solution ideas for a specified problem statement. This data will therefore be input as a string, and output to the database as such. It is assumed that users have some significant level of education, since the primary target users are those on an engineering design team. Therefore, the interface of the tools will be intuitive to use for users with basic computer skills. Although each tool created has its own set of requirements, the interfaces for the tools are primarily drag-and-drop, with simple text areas for inputting data, and buttons for directing the user on actions they can perform.

Constraints
The largest constraint was the fact we were working on a system that had already been built, and had to extend from that system. The existing system had already been created using PHP, Javascript, and MySQL, and changing to any languages or other tools that may have made things easier for progress would have taken too much time.

Development Process
Dr. Hensel requested several target dates throughout the project that would involve user test sessions of the product. He requested we use an incremental development process for the project, focusing on completing milestones corresponding to those dates. We were fortunate in that our sponsor was able to meet with our group on a weekly basis, where we were able to clarify and negotiate on the requirements for our project. We kept track of goals for each increment by defining tasks within the Mantis bug tracker tool, where
each goal was described in detail and assigned to individuals on the team to complete. Before each increment, we held a testing session of our own to ensure the tools worked properly before the live user test sessions. After each increment, we were allowed to account for feedback provided by the test sessions, and revise the scope for the next increment by deciding which features were important to modify for existing tools.

Our team roles were divided based on the experience of the team members, involving having some team members working primarily on the back-end portion of the project, and some team members working primarily on the front-end. We had an assigned project coordinator that kept track of the weekly schedules, and made to update and track the various tasks assigned to everyone throughout the project. At our weekly meetings with the sponsor, one person took down all the meeting notes, and one person primarily spoke with the project sponsor about our updates and concerns. This person was also the person that contacted the project sponsor whenever necessary, such as to confirm the weekly meetings and send an agenda. Everyone updated their artifacts to the team website as necessary, as this role was never clearly defined to any individual.
Project Schedule: Planned and Actual

Due to the planned out user test sessions from our project sponsor, we set each of those dates as our major milestones. Each of these milestones ranged from three to five week long increments, and our schedule was created around the set dates. This resulted in a total of six increments as a whole.

Our first increment was the longest, so we scheduled this to give us the time to learn the domain of the project, as well as analyze the existing structure of the system. The remaining increments were originally scheduled to allow portions of the team to work on updating the existing tools, and another portion of the team to work on a new tool for each increment. Also, our third increment we aimed to test the installation of FACETs alongside EDGE on a clean server to ease the transition to the live EDGE server at our last increment.

By the end of the first increment, we were only able to successfully refactor the system for only one toolbox, as we greatly underestimated the task. This meant we had to rework our entire schedule as soon as our first increment did not match up with our planned activities. Our second increment focused on completing the refactoring for the remaining toolboxes, and beginning to work on a new tool, and update the tool that was tested after the first increment.

System Design

Coming into the project, we were tasked with working on an existing design and implementation, both for the FACETS modules, and for the underlying EDGE system on which FACETS was based. Our design role was therefore limited to a moderate redesign of the existing FACETS components, with a primary goal of extending the existing design to encompass new types of tools and methods of communication between tools.

The existing FACETS design had a class hierarchy that was both overly simple and rigid, that did not allow us to easily add new functionality or properly abstract general tool behaviors. Where the old hierarchy contained a single base class, we took the opportunity to add a second layer to the class hierarchy for each type of tool implemented, which allowed us to further segment and separate behaviors that were common to all tools, a subset of tools, or individual tools themselves. The resulting design is a combination of the existing design from the EDGE system, the existing FACETS system, and our new improvements to FACETS.

At the base of the FACETS architecture is the EDGE system interface ProjectNode, which represents a single submodule or page in the application. From this interface, the base class ToolNode contains generalized behaviors that are shared by all tools in the system, such as how to display forms via the FormFactory interface. This class also maintains the base code for integrating tools into EDGE in a standardized format.
Extending from this base are the basic tool types, NeedsToolNode and ConceptToolNode, which we have segmented based on the various toolboxes in the FACETS development process. This decision was based on interaction with Dr. Hensel, where we determined that tools in a toolbox would generally contain a subset of shared behaviors and data models that were not shared with other tools or toolboxes. This “middle” layer of the design allows each tool type to abstract actions, such as modifying shared data models or handling generalized Javascript actions, away from the individual tool implementations.

Building off the primary tool types are the individual tool classes: ObjectiveTree, AffinityDiagram, and Brainstorm. These tools define a concrete implementation of the corresponding tool type, and contain only the behaviors that are specific to that tool, including the presentation of data models to the user. The individual tools also contain their own Javascript behavior definitions that are loaded as part of the page to handle asynchronous updates and user actions by communicating back to the server via JSON.

In the case of the Brainstorm tool, the progression of stages was implemented directly within the Brainstorm tool, although the data model is abstracted enough to be implemented at a higher level in the class hierarchy. This decision was made as there are no other tools currently implemented that require more than a single stage, and it was best to isolate the feature in Brainstorm rather than risk adverse effects to the other existing tools. In the future, if other tools are developed that require this behavior, it would be ideal to abstract that behavior into a higher level class rather than duplicating the implementation.
Process and Product Metrics

For our scheduling purposes, we kept track of actual time versus estimated on individual tasks for each person. These times were kept track on a weekly basis, with estimates being made at the beginning of each week. The estimates for similar tasks were based on the trend of actual times from previous weeks in conjunction with predicting how much time any new tasks may take. Tasks involving documentation and research were often times overestimated; however, tasks involving development general were underestimated. This may have been the result of not putting enough time into our documentation and planning before developing, resulting in more hours being put into the development phases of our project later.
Open tasks versus closed tasks were tracked as defects for each major milestone in our project. Each task was flagged with a severity level as well, making sure to mark anything that had to be fixed or completed first with a higher level than the others. This allowed us to track who was working on which portions of the project, and see which open tasks were major errors that should be worked on and closed before creating new tasks. At the end of each milestone, we kept track of how many of these tasks remained open to measure our progress as we worked from one release to another.

Often times, many of our tasks assigned remained open at the end of a milestone, and these tasks would have to roll over into the next release. However, since many of these tasks that were not completed had a low severity level, it did not hinder any major advancement from one release to another. Tracking the tasks in this manner helped visualize our progress, and keep our major goals in sight throughout the project.

**Product State at Time of Delivery**

The FACETs system is a combination of many individual products which can and are developed independently of each other. Read for wide area release are the general improvements to the FACETs system which our team implemented, the Affinity Diagramming tool, the Brainstorming tool, and the Objective Tree tool. There are, though, many subtasks which are still outstanding for each of these items. With careful consideration we determined these to be less critical to the initial deployment and will be logged here and in the issue tracking system for future consideration.

There were some tasks which we eventually decided to scrap plans for and revert our work. The notable example is the update to the EDGE system to utilize ID numbers instead of text to identify projects in the system. This was outside the original scope of the project but we chose to take it on as an upgrade to the EDGE system to ensure its continued scalability and adherence to accepted standards for data storage. Over the course of the quarter we discovered the depths in the system which relied on the existing inefficient method of data storage, and the risks in changing such core portions of the system. We encourage continued effort to be put in to optimization and cleanup of the existing base system.

There are notable items which were not completed:

- The BrainBall application was in an inoperable state when we began the project and through changes in requirements and scope was never picked up and repaired.
- Miscellaneous Brainstorming bugs which are noted in the bug tracker. The product is ready for use, but as always, can be better.
- Miscellaneous Affinity Diagramming bugs, none of which are blocking to a release.
Project Reflection

From the start, we did not define a software requirements document. We loosely agreed that we would be refactoring the existing system to accommodate creating additional tools on the system. Also, we knew from the start that we would be bringing FACETs to a live server by the end of the project. Fortunately, Dr. Hensel was able to meet with us on a weekly basis. Before our second meeting, we made our estimates on how long it would take to refactor the existing project. However, our estimates proved to be inaccurate, and our scope quickly slipped before the first major milestone.

The existing FACETs structure would not accommodate many of the new requirements that new tools would have. Many of the existing had major usability flaws as well. This meant we had to rework our schedule. After another meeting with Dr. Hensel, we agreed that since the system would be going to a live server, it would be in best interest to update what has already been completed, and focus our schedule on making these updates. In addition, we would create one new tool, and if time permitted after the updates to the existing tools, begin work on other new tools to leave for future teams to finish.

Having no set requirements from the start and having given poor estimates on the schedule definitely hindered progression, and made for a slow start to the project. However, we were able to recover and negotiate which requirements would be most important for release to a live server, and this proved to be beneficial. Our task tracking was kept up to date, and this proved most helpful in staying within our planned schedule through much of the project.

Different members from the team brought different areas of expertise, which was helpful when deciding on who would perform which tasks. Having these roles decided upon from the start made it easy to track who was working on which portion of the system. Also, we were all able to attend most of the planned weekly meetings, and work as a group. This allowed us to bounce development ideas back and forth freely, as well as work out any major problems more rapidly than could be done if we did not meet as often.