The FACETs System

Team FACETs

Chris Booth  Kevin Borgnis  Jonathan Hatalla

Project Sponsor
Dr. Edward Hensel Jr.

Faculty Mentor
Dr. Stephanie Ludi
Table of Contents

Project Overview ............................................................................................................. 3
Basic Requirements ......................................................................................................... 3
Constraints ....................................................................................................................... 4
Development Process ...................................................................................................... 5
Project Schedule: Planned and Actual ............................................................................. 6
System Design .................................................................................................................. 7
  3-Tier Architecture – EDGE .......................................................................................... 7
  UI Design – HCI ............................................................................................................ 7
  Scripting Engine ........................................................................................................... 7
  Database – ERD ........................................................................................................... 8
Integration of the Final Product ....................................................................................... 9
Process and Product Metrics ........................................................................................... 10
  Process vs. Development vs. Overhead – Team ......................................................... 10
  Process vs. Development vs. Overhead – Individual .................................................. 11
  Backlog Management Index ...................................................................................... 12
Product State at Time of Delivery ................................................................................ 14
  State of the Product .................................................................................................. 14
  Missing Features ....................................................................................................... 14
Project Reflection .......................................................................................................... 15
Project Overview

The FACETs project was presented as a senior project idea by the Mechanical Engineering Department Head, Dr. Edward Hensel. Dr. Hensel is the creator of the FACETs design methodology which is the motivating factor behind the FACETs System. This methodology focuses on twelve phases of product development and stresses that these phases may be actively addressed concurrently as opposed to a linear progression in the various stages of product development.

The FACETs methodology employs many tools which facilitate the progression of the various stages. These tools have long been used in the Mechanical Engineering Department as well as in the Multi-Disciplinary Senior Design course. To organize the research and data of the multi-disciplinary senior design course, Hensel employs the EDGE system which is a web front-end for a data repository.

Our project was to design a system which conformed to Hensel’s FACET methodology and incorporate the system into the EDGE system. The system is currently planned to be used by students in the senior design program, however future goals include the FACETs System to be an open-architecture system which will be used and improved upon by others in the industry.

Basic Requirements

The requirements of the FACETs System were broken up into two core areas. The first area was the administration functionality. These set of requirements dealt with the permissions and tool and toolbox names as they related to a FACET project. The second major area of requirements for the FACETs System was the tools themselves. These tools each had their own unique requirements as each tool provides a different benefit. For full requirements, read the respective document relating to the part of the system the reader would like more information on.

The administration requirements focus on aspects that are more relevant to administrators and facilitators than engineer and guests. A few key requirements are that a user of administrator or facilitator level must have the ability to create new
sessions. These sessions allow users to access the various tools which were developed, such as Affinity Diagramming. Another key feature is that administrators must be able to assign or remove specific roles of users in a project. The roles are Administrator, Facilitator, Engineer and Guest. One of the last key features for the administration requirements is that administrators or facilitators must be able to import data into a session. Examples of imported data are teasers for the Brain Ball tool. For families of related products, starting from scratch is something that rarely happens. These types of products usually have preexisting projects that are either built upon or there is already some core idea or information that is being used to start the project. Importing this data provides the tools will useful information and a starting point for teams to begin their active development of the product.

The requirements for the tools are largely independent of other tools. The following requirements are those that relate to all tools in general. Tools must be able to have data imported into them. While the functionality to add data to session is reliant on administrators and facilitators, the tools themselves must process the data that is being received and serve that content to the engineers in the best possible way. Tools must interact with engineers to provide an experience that mimics the real-world tools. The tool must also store the interactions and propagate those interactions to other engineers working with the tool. For storing information, the tool must store information to the database such that it is both permanently saved and is available for future retrieval. Requirements specific for tools can be find their respective requirements documents.

**Constraints**

Operating Environment (Debian Linux w/Apache)
Implementation Technologies (web friendly languages/scripting languages – PHP, JavaScript, and XHTML)

The biggest constraint on the project was the fact that it was intended to be an extension of an existing system. This meant that several architectural decisions had already made. Since the existing system was already in place built on Debian GNU/Linux running Apache, PHP and MySQL, the current project should be built on the same technologies. A second condition that played a large role in the design process was the
hope by the sponsor that this project would eventually be turned into an open-architecture project. This caused the team to focus on technologies that were very common and have a wide base of support, both by browsers and a developer community. Because of this the project would be coded in PHP for application logic with Javascript and XHTML/CSS handling the user interface. All communication between the client and server was handled by a JavaScript library.

Kate Nordland and Dr. Hensel had already detailed the majority of the database structure. This simplistic structure made it difficult for us to do some more advanced things that Dr. Hensel would have liked to see with at least the Affinity Diagramming tool. Specifically the ability to place groups of ideas into other groups. We were limited from doing this in our ability to determine which data entry in the database was a grouping and which was an idea. Our current method is assuming that any data entry that is not a child in the relationship table is a grouping. This works for our current implementation, it poses problems when we have groups within groups. In this scenario we could expand our grouping definition to allow for any data item that is not a child in any relationship or any data entry that is a parent to be a grouping. The only case where this logic falls apart is when an empty grouping exists as a child of another grouping. This is a problem we are leaving for future developers to work around.

Development Process

The process used for development was a staged delivery process. The process was discussed with the sponsor and it decided that it would be a beneficial process for both the customer and the team. The sponsor had not had previous experience with a software engineering development team; therefore the processes used for developing software were unfamiliar to the sponsor and no process mandate was issued. The process provided iterative stages which allowed specific stages of the product to be shown to the customer.

There were many roles which were assigned for the project. The process manager role prepared agendas for meetings and kept the schedule up to date with tasks that were completed and those that needed to be completed. Other roles which were identified for the team were website administrator, technical lead, development lead and server administrator. Due to the small size of the team, members were required to fulfill many
roles. They were assigned based on past experiences and how those experiences made them best qualified for a specific role.

**Project Schedule: Planned and Actual**

The first three weeks of the project were dedicated to learning the domain. After having a handle on what we would be developing, and deciding upon a process that would work well for the project, an initial schedule, available here <FACET-ProjectPlan.doc>, was created by dividing the estimated work into stages, allowing for one more week of development during the first stage. With this schedule, the project would be broken up into three specific stages, one for each toolbox, and an overall system requirements gathering stage.

Each stage had the goal of delivering a new integrated toolbox into the edge system. The activities that would achieve this goal were not detailed until the start of that stage, after its' requirements phase had taken place. During the first stage, no document or schedule was created to define when tasks would be completed. Realizing the significant impact this had, in the second stage we created a task list with specific deadlines and descriptions. Although a better format than none at all, the task list was rarely referenced, thereby causing the same slippage in schedule that we had experienced during the first stage.

At the start of the third stage, a meeting was held to resolve the issue of a constantly slipping schedule, and lack of quality documentation indicating both what is required, and what has been completed. This resulted in the creation of the final schedule, which can be found here <Schedule.xls>. This schedule is a detailed, itemized list of what tasks were required of the stage, including information on start and end dates, time requirement estimates, actual man-hours, descriptions, owners, and current task status. Keeping this constantly updated allowed us to see what items were in danger of slipping, and act on them.
System Design

3-Tier Architecture – EDGE

The FACETs system is an extension of the EDGE framework, and as such, makes use of the existing architecture already in place. The EDGE framework is designed with a standard three tier web architecture, composed of a presentation, application, and data layer. In addition to making use of the general architecture, the FACETs system also made use of the technologies already in place. A MySQL database stored all the data for both the EDGE and FACETs systems. The application layer consisted entirely of PHP scripts, and the presentation layer was handled with the combination of JavaScript for interaction and CSS for visual display.

UI Design – HCI

The system interacts heavily with users therefore the design of the UI for each individual tool was important. The affinity diagramming tool features a drag and drop interface which most users are familiar with. The objective tree tool is less intuitive to the mental model of the user, therefore we provide a help document to mitigate any difficulties the user might encounter otherwise. All tools provide a minimalistic interface which is designed to provide the user with fewer distractions.

Scripting Engine

With EDGE there was already a structure to how the scripting engine belonged. The bulk of the engine was stored in a folder on the file system. The remaining data was stored in a subversion repository that the engine accessed directly. The engine largely consisted of two different types of scripts: database access scripts and request nodes.

The engine determined what type of request was being made of the server. Then, the engine loaded the appropriate type of request node and passed the request to the node for processing. The FACETs system added two types of nodes: ListToolsNode and ToolNode. The ListToolsNode is very similar to the ListScriptsNode and both are responsible for presenting the user with a listing of tools/scripts available to them. The ToolNode is very similar to a ScriptNode and provides an interface for the tools to handle requests. While ScriptNodes are housed in each super project individually,
ToolNodes are housed centrally in the FACETs project. This allows for immediate access to all tools for all FACETs enabled projects.

The database access scripts all extended the DB_Tables class. EDGE made use of DB_Tables for easy access to the underlying database without exposing the SQL to the developer. DB_Tables are an extension to PHP and are part of the PHP Extension Application Repository, or PEAR. Because the EDGE system required that these database access scripts were all located in the same directory, the FACETs system added it’s own DB_Tables to the existing directory of EDGE DB_Tables.

Though the tools’ PHP scripts, coupled with the EDGE engine, provided the structure of the HTML page produced, when it comes to the usage of the design activities handled by the tools the page is dynamically created using JavaScript and CSS. The JavaScript handles creating and displaying the XHTML used to show ideas, groupings, and comments for each of the tools.

**Database – ERD**

The data for the FACETs system is stored in a MySQL database located on the EGDE server. The desire of the FACETs database was to provide as abstract a storage repository as possible to allow for maximum tool interaction. To this end the relational database described in Kate Nordland’s [web paper here](#) was used as a starting point. With this design, there was a central data table where all items would be housed, associated with other data items through a relations table. In addition to this, a member table provides access to separate areas of the FACETs system based on permissions.

While this design was used as a starting point, it evolved a number of times over the course of the project. Since all data was being stored in the same table, specific session information was needed to keep track of which tools and which projects were associated with particular data elements. All this information was stored in the data table. Due to the limited nature of a PHP system and its’ inability to push information to clients, an activity history table was created. This allowed clients to poll the system, looking for any changes that may have occurred since the last poll.

The team realized that this structure was causing the data and history tables to become bloated. Every time a slight change was made with a specific data element, a new
history element was created. With a large number of projects, eventually the amount of information being processed during each poll would cause a significant delay in server response. The first of the changes was to remove the history table by adding a timestamp field within the data and relations table, and a text field to show the last editor of the specified item. This timestamp would reflect the last time that particular element had been changed. These two changes were replicated on all tables as a means to track activity in the system.

The next refactoring that took place was again on the data table, which was storing information that did not directly pertain to the data of a tool, but to the session. This included the project of the tool, the tool type with which the data was created, and the name and problem description which uniquely identify the session. Also at this point the team included two timestamps to serve as an open and close time. This would prevent any changes to the data from taking place outside desired hours of operation.

Initially this database was accessed through the use of the MDB2 PEAR extension. With this setup the team wrote the SQL queries used to perform basic CRUD functions on the facet database. Towards the latter half of the second quarter, the team switched db access over to PEAR's DB_Tables, which is the method of access used by the EDGE system. This allowed for associative indexing of the tables for more readable access.

**Integration of the Final Product**

The initial prototypes for the system were done as stand alone XHTML pages using direct access to the database through PHP's standard functionality. The prototypes were not as flexible in their use of the database as the final product turned out, and the database usage was sometimes clunky, with a majority of the SQL being written by hand as opposed to dynamically generated. However, since JavaScript was not a strong language for anyone in the group, the prototyping allowed for the functionality to be built without having to worry about the integration aspect of the project.

After the integration was begun, it became apparent that hard-coding each of the SQL queries was not the most efficient way of doing things. As the integration progressed, it came to be understood that EDGE was using DB_Tables for access to the database and that a lot of the needed functionality was available without hard-coding SQL queries. During the integration process, each of the tools were transitioned to using DB_Tables
for database access which improved the quality of the overall system and maintainability while reducing repeated code.

Additionally during integration there was a need to make FACETs tools integrate nicely into the EDGE system. The design decision was made to make them similar in structure to EDGE scripts since their behavior was very similar. Some changes needed to be made, however, as the FACETs Design Tools were intended centrally located in one project while EDGE scripts were existed only in the Super-Project in which they were created. Because of this, some support needed to be added to the EDGE engine for the ToolNodes and ListToolsNodes to be able to function as intended.

**Process and Product Metrics**

The team used a combination of metrics including Process Time vs. Overhead Time vs. Development Time, and a Backlog Management Index (BMI). A third metric which was gathered but never used was Team Time vs. Individual Time. Below three charts are provided which give a visual depiction of the data gathered.

**Process vs. Development vs. Overhead – Team**

The team spent the majority of the man-hours on the project, nearly half, in development and testing of the system. This includes constant maintenance and integration of tools into the edge framework. The second largest consumer of time was process work. This is deceptively large because much of the time included in it is due to meetings, which included unaccounted for development and overhead activities. It is estimated that approximately half of this time can be attributed to either of these two categories, with a majority of that time going towards paired development.

The third and final category that time was recorded in is overhead. This accounts for the smallest percentage of time. This includes activities such as process and project management, system configuration and maintenance, document upkeep, and domain learning. A large majority of this overhead, approximately 20 percent, comes from the initial setup and learning the domain during the first few weeks of the project.

While a majority of the time was spent during development, this chart clearly shows the large amount of time spent between process and overhead. The time not accounted for due to miss-recorded hours during meetings shows that the team was not keeping
adequate documentation of their time. The fact that 50 percent of the time was spent on development goes to show that although a set amount of time is set aside for a project, to assume that only half of that is available for actual work to be performed. This chart alone, while interesting, does not provide much information along the lines of how things are going, aside from if too much time is being spent on one of the categories, leaving the others neglected.

The chart above shows the percentages of man-hours that each category of tasks took. It should be noted that the remaining week 10 and 11 of the second quarter for the project is not included due to deadlines. The majority of this time would fall between process and overhead, but not contribute a significant amount to either.

**Process vs. Development vs. Overhead – Individual**

While the above chart was a collection of all times for the team, it did not show an individual breakdown of effort in each area. On the individual level, Chris put in 109 hours for development, 71 hours for process, and 40 hours for overhead work. Kevin put in 78 hours for development, 41 hours for process, and 45 hours for overhead work. Jon spent 123 hours in development, 73 hours in process, and 53 hours of overhead work. The significant difference between the hours spent in overall process between Kevin and the remainder of the team can be attributed to the method of recording data.
Kevin did not record meeting times in his time and effort after the first few weeks of the project.

It is interesting to note that being assigned the role of process manager did not make a significant difference in the hours spend in overhead. Again this can be attributed to a large portion of the hours associated with overhead being spend during the beginning of the project. It was noticed that for more meaningful data to be obtainable from this sort of a metric, a more constant, standardized data collection scheme is needed. As it was the data was collected on a weekly basis, but not added to the metric document until the end.

![Chart showing man-hours per team member](chart.png)

The chart above shows the man-hours put in by each of the team members in the respective categories of process, development and overhead.

**Backlog Management Index**

The Backlog Management Index (BMI) is used to show the ratio of tasks being completed vs. tasks being assigned. The formula for this is:

\[
BMI = \frac{\text{Number of tasks closed during the week}}{\text{Number of tasks opened during the week}} \times 100
\]
If the ratio is greater than 100%, the backlog has been reduced. If the ratio is less than 100%, the backlog has grown. With this metric, the team can see just how effective and efficient they are at completing tasks on the schedule.

The above chart shows the number of tasks opened, number of tasks scheduled to be closed, and number of tasks actually closed each week. Due to the lack of a detailed schedule with open, expected close and actual close dates during the first quarter of the project, this information relates only to the second quarter, assuming a clean slate. This is not completely accurate, but the existing open tasks were added during the creation of the schedule.

This chart can be used with the detailed schedule to check areas where the number of scheduled closed tasks and actual closed tasks differ. The tasks at that location can then be examined to determine what the cause for the delay was in attempts to better predict times in the future. This chart shows that the team strayed from the schedule on a number of occasions during the month of September, causing a propagated backlog that had to be handled at the end of the project.
Product State at Time of Delivery

State of the Product

The product which is to be delivered will include all major features that were planned after the re-scope. These features include three design tools. These three tools are the affinity diagramming tool, the objective tree tool, and the brain ball tool. The affinity diagramming tool features drag and drop components, while the other two feature a minimalistic UI which satisfy the needs of the customer. Three administration tools are also slated for delivery. These tools are the role administration tool along with tool and toolbox management. The role administration tool allows FACET specific roles to be assigned such that a given individual can have permissions set which affects functionality throughout FACETs. Probably the most important aspect of the product that is slated to be delivered is the framework and documentation that supports said framework. The framework allows both design and administration tools to be developed and integrated into FACETs with relative ease. The framework isn't without difficulty, which is why the product also includes documentation that will significantly decrease the time required for a future team to become proficient with the system.

Missing Features

The product is complete to the final scope. Originally planned were three toolboxes containing a total of six design tools. There was a vague understanding that the product needed administrative functionality, but it was never planned into the schedule. After a re-scope, the product became a collection of three design tools in two tool boxes to showcase the concept of multiple toolboxes. Additionally three basic administrative tools were added to an administrative toolbox. Additional missing functionality is drag and drop for Objective Trees. This was originally planned to be drag and drop like the Affinity Diagrams, but the complexity of making this work across multiple browsers made this feat impossible within the given time period. Therefore is was modified to work by first clicking on the idea to move and double clicking on the idea you want to place it into.
Project Reflection

A lot of things went right in this project. Integration with the existing EDGE system was relatively straightforward. The server was well maintained resulting in a 99.4% uptime. The team delivered everything scheduled after the re-scope in the second quarter. The framework was also well documented such that future teams should be able to hit the ground running developing tools.

The team started off slow, which resulted in the first two milestones being completed behind schedule. A factor that contributed to the slow start were overlapping vacation and the poor planning associated with this. Another factor that contributed to this was the lack of domain knowledge and technical knowledge and the time it took to learn these items. Time management was a major hurdle that needed to be overcome to complete the project. In the last quarter the team actively addressed this issue by creating a detailed, itemized schedule and creating the role of a process manager to watch over it and prioritize tasks. This allowed the team to complete the project as initially scheduled after the re-scope.

In the future, the team would be more proactive in assigning tasks and staying on top of the project. The team would also start the project sooner and better schedule vacation times such that they did not interfere with the active development of the project. Pushing the sponsor for an additional contact knowledgeable about the requirements would be another move that would have ensured that project progress was not interrupted. While the team’s process provided a general structure, further defining the details of the process would have enhanced the effectiveness of the process.

A major point that the team learned was the importance of a well-defined and maintained schedule. The team also acknowledged the importance of prototyping as a means to understanding the technology and problem domains.