FA E-Research Community
Team SFC
Christopher Greenawalt    Jeffrey Kurtz
Benjamin Taber          Thomas Whitmer

Friedreich's Ataxia Research Alliance
Jennifer Farmer

Hewlett Packard
Patricia Gimbel
Bill Hartnett
Ron De Mena

Faculty Mentor
Nicholas Bucciarelli

Project Overview
Friedreich’s ataxia is a rare disease and that faces several critical challenges to advancing research that will ultimately bring treatments and a cure. Time and resources are the main challenges facing the Friedreich’s ataxia community. Friedreich’s ataxia has only a small number of researchers working on the disease, approximately 300-400 worldwide, and the number of grants and funding opportunities for research are small when compared to more common conditions like Parkinson’s disease, Cystic Fibrosis, etc. Time is also a valuable resource – the time it takes to conduct experiments, apply for grants, prepare research publications, investigate new advances in the field, etc. Any efficiencies that we can bring to the research community that reduce the amount of time it takes them to conduct their work the more time is translated to patients battling this disease.

The project proposed is to create an e-research community that fosters collaboration, sharing of information, and a repository for research publications, protocols, directories and funding opportunities. It can take 6-9 months for scientific results to reach the research community through publications and not all research gets published. In-person scientific forums or conferences often only occur on an annual basis and only include a subset of the research community. Creating a secure e-research community where FA scientists and physicians can share information instantly would dramatic speed the pace of research. Researchers would not be working in their individual silos but rather sharing information about protocols, research tools and results – decreasing duplicative efforts and time to conduct experiments.

To solve this problem, FARA commissioned a social networking site that would allow researchers and doctors to share findings and updates in real-time over the internet. Researchers would be able to register with the site and post new content alongside fellow researchers. As updates come or research is completed, the researchers can post that information to the site to be shared amongst the entire FA community. The site would hold a calendar of events that would allow site users to see what events related to Friedreich’s Ataxia are upcoming. A news
aggregator would collect and sort relevant news items from external sites and research journals, and display it in chronological order on the E-Research Community for easy access. A database search would allow researchers to easily search external scientific databases and be taken to those results instantly. The site would host a wiki-style guide containing all the known information about Friedreich’s Ataxia. This information would be user-generated and created so that anyone who discovered new information about the disease that information could be instantly updated in an easily-accessible location for all to see. The site itself would be searchable by any user, allowing them to easily locate specific information without clicking through the site looking for it. These features will work together to make the site pleasing and easy to use.

To be considered a success, researchers will obviously have to use the site. The purpose of this site is to promote collaboration between researchers and accelerate the research process, which would not be possible if nobody is using the site. The site would need to be easy enough to use that typical researchers would not be scared off by the complexity for adding resources or performing a search. The site would have to work properly, providing accurate search results and storing documents and research results for as long as the user’s desire.

**Basic Requirements**

In this section, I will go over the most important features of the project that we undertook, to highlight the core capabilities that absolutely had to be in place to consider this task a success. While it will be fairly descriptive and address the main points the website was created to address, it is not intended to be a complete list of requirements that were established between the team and our sponsors at FARA and HP. For a complete list of the requirements that existed for our project, please refer to the requirements document that was created during the project's early stages.

The first requirement I will be discussing is the desire to have the website maintain the look and feel of FARA’s current properties. This was modeled mainly after the curefa.org website, which is the main landing website for FARA. The color scheme was required to be maintained within our website, to follow the looks and feel associated within secondary pages on this website. To try and describe it quickly, it uses red, blue colors for the navigation links contained on the page, and white central area to display content. Along with the color scheme, FARA wanted to maintain the same navigation process within our website, so that new users are able to quickly adapt to the new system and not have to learn a new navigation system to lower barriers to adjusting to the new content. In order to properly address this look and feel, we were able to utilize the files used to create the theme for the curefa.org website, as well as working directly with the sponsors to make sure it met their standard of appearance.

A second requirement associated with our project is to enable collaboration between researchers working on FA itself. This was designed around the general problem associated with the network of doctors and researchers working to cure this disease not communicating with one another for up to 6 months at a time. With this unacceptable gap of time between sharing of information, there is a very real chance that work can either be duplicated, or very beneficial findings will not be shared between the other researchers until these biannual meetings took place. This website was to help these men and women communicate with each other, and share information between each other in real time, instead of rely on the conventions that happen every 6 months. In order to bring collaboration between them, the website was designed to allow them
to upload documents, communicate on a forum, as well as other collaboration tools that allow
them to read stories that exist on other medical websites that match terms important to
Friedreich's Ataxia. These features allow them to share information, as well as ask questions that
may lead to quicker location of findings within this community.

The third important requirement that was established for this project was to create a website that
is easy to maintain by a non-technical administrator. This stems from the current format that
cureafa.org is operated under, which requires FARA to edit the HTML files directly to complete
any changes to the website. This would be manageable for a company or organization that staffs
an IT department, or some form of technical staff, but FARA is unable to staff these members for
their needs. This means that the current situation that is used to update their website is very
ineffective, and relies on work completed by friends or any potential volunteers that just happen
to know HTML for something like placing a new piece of news on the front page. For the
project brought to us, this would be deemed an unacceptable way to edit content on the website,
and we needed to make sure that the website could be maintained and updated by a non-technical
administrator instead.

Constraints

Key networking protocol standards such as TCP/IP and HTTP and the markup language HTML
have driven the explosive growth of the Web. For the next phase of growth, industry will
continue to use server side technologies such as PHP, or asp.NET. In the case of our project,
PHP was chosen over other options due to the teams researched knowledge of CMS's on the
market. Our team came to realise early in the project that the utilization of a preexisting
framework would be the only way we were able to deliver the site that FARA wants. Using the
web platform, we decided to use Drupal to get the largest return on our time. We saw that of the
CMS's we were looking at, that Drupal fulfilled, by far the most functionality that we needed.
This was done to enable the team to progress further and to enable more integration of all the
information available about FA. There were also quite a few issues with the database, due to
choosing drupal as a platform. The original choice for a database was MS SQL, but due to the
weak community interest in using other data stores with Drupal, and increased complexity of
installing drupal on a database other than MySQL, we realized it would be best to limit ourselves
to MySQL. This then gave us the opportunity to take advantage of Drupal's easy to set up
administration interface. The interface was chosen based on the ability for those with limited
technical knowledge could access the sites admin and make meaningful changes to the site.
FARAs original website used a certian look and feel, and we were asked to follow that as well
as we could. Look and feel didn't constrain our ability as much as our choice in navigation
throughout the site. Time was probably our biggest constraint. Most of our choices were to
allow our team to deliver as much functionality as we could in the limited amount of time we
had. But in the end we found that these technologies streamlined the development process for
creating an effective suite of online tools for the community of medical and research
professionals.

Development Process

Our team utilized a variation on the SCRUM process, with releases spaced out in two-week
sprints and two meetings each week. This was not mandated, but was encouraged, by our
technical advisor. This process worked well for us and the sponsors because the frequent
releases made it easy to show usable progress and keep the sponsor up to date on how the
The project was progressing. We would hold at least one conference call with the sponsor each week to let them know what the current project status was and how each of the team members was progressing with their assignments, and to solicit feedback on the previous sprints and feature progress up to that point. We would also email back and forth frequently to provide immediate updates and request specific items we discovered were needed between the weekly meetings.

Each member of the team served equally when it came to development, but we identified general team roles as follows: Chris was the Team Leader, Jeff was the Sponsor Coordinator, Ben was the Website Coordinator, and Tom was the Meeting Scribe and also the Scrum Master, since he had the most experience using that methodology. These roles seemed to fit well with each team member's personality and if issues arose with one member's role, the other members rose to the challenge to solve the problem.

**Project Schedule: Planned and Actual**

The project schedule was originally derived from the RIT schedule in terms of documentation. Based off when things were due for RIT, we planned our own schedule initially around that. This was mainly done for the planning stage of our project. Once we were closer to development, we prioritized the requirements and tasks that needed to be done and determined which sprint they would fall into. The way we prioritized our tasks were through sponsor feedback as well as a potential user survey to gauge which features the end users would want. Based on the feedback of those, we determined which features were the highest priority and which were the lowest priority and scheduled the higher priority tasks to be completed first.

Our major activities and milestones were the end of each two week sprint that we had where we would release a version of our product to the sponsor. Each sprint we would set out to accomplish certain things and have them finished by the end of that sprint. For the most part, we were on schedule with delivering what we said we were going to deliver and there were very few instances where we missed a deadline. With that being said, there were a few times where we experienced schedule slippage. Most of this is due to the fact that we were overly aggressive in our scheduling and thought we could get done more than was realistic. There was also a few times where there were difficult technical issues that we had issues with that caused us to miss a milestone. The way we adapted to this was to utilize other team members for assistance if needed or in extreme cases utilize the HP technical adviser that had volunteered to provide expertise to us if necessary.

As you can see below, we tracked the task burn down of our project. This allowed us and the project sponsor to quickly see how close we were to actually completing the project. As you can see, we kept a fairly good pace in terms of completion of tasks. Towards the end of the project our pace drops off mainly due to the fact that we were mainly focused on testing and server migration and no longer developing new features. Server migration took a long time because there were issues with the web host that our sponsor was using. We had to work very closely with the sponsor to get the project up and running on their boxes and to make sure they
understood how to migrate the site again in case that was necessary.
The primary architecture solution for our project centered around the Drupal architecture that we decided to use after examining the requirements presented to us. Since we decided to leverage the DBMS system that is being supported by an open source community, it limited our architecture to conform to their implementation. Conforming to Drupal’s standard allowed us to leverage their built-in security, content management capabilities, and other components that were deemed to be vital to the success of our project. The above diagram represents a view of how the Drupal DBMS is designed, and helps to understand what is going on under the hood of the project.

The bottom layer to this system is the Database API, which is used to allow all of the components to communicate with the MySQL database that stores all of the content on the website. This makes it so each of the other components just calls a command in the DB API, instead of creating a connection to the database, using it, and then tearing it down. It also allows the system to be more secure, as the credentials for the database are not passed around the various components, but controlled in one central location.

On top of the DB API sits all of the various components that make up the features that are built into Drupal by default. These are all displayed in the diagram above, but some of the more important API's will be described to help portray what they are used to accomplish. The first of which will be the Node API, which keeps track of how individual pieces of content are created. A node within Drupal represents each content instance, such as a forum post or a posted resource page. This API is used to help accomplish the various tasks needed to take a form filled out
within the website, and convert it into the necessary entries in the database to store it and display it once loaded by any user from then on.

The second API of note in the middle area is the Block API, which addresses the various components that are displayed on a given page. The idea of a block is basically a component some feature provides, be it a module you install, or from a core component of Drupal itself. For our website, you can see an example of a Block component by viewing the Search box that allows users to locate information on the website. The API supports placement in the header, footer, left panel, right panel, or content (or central) region on a given page. Along with the placement functionality, this API also provides functionality on when to display the given Blocks to the users. This ranges from controlling things based on user types, such as administrators or public view, as well as to specify which pages within your implementation will display a block.

Another key API is the User API, which handles all of the role capabilities within the website. This API provides the functionality to create users in general, as well as provide some other vital functionality like assigning roles to a given user. Without the use of this API, there would be no sense of security implemented in the project, so it was used properly to ensure sensitive materials are kept behind appropriate secure pages. At the same time, it allows you to implement any number of security roles to the different users that exist within the system itself. These roles are defined as you implement your Drupal instance, and you can provide different permissions for each role. With this in place, it allows you to generate different capabilities for each user type, and add some additional security options for the system.

Along with the mentioned API's that exist within Drupal, this layer also contains some other key components that are actually just implemented subsystems to help you deploy a proper website. The most important of these options is the Views system, which allows you to customize some of the default functionality with various visual touches. This can range from allowing forums to alert users which forum posts have not been viewed by them so far, to changing how different components are displayed on a given page. This customization capability is key if you desire to leave some functionality in place and not replace it with custom items, and was used in a few spots to make the Drupal website feel more appropriate for FARA's desires.

On top of these API's and subsystems is where the Themes system resides for Drupal, which handles the look and feel of websites. To give a general definition of the Themes system's importance, this is where the actual CSS and appearance of all times takes place. It comes bundled with a few different options for website creators to use on their creation, and remove the general Drupal scheme for colors and appearance. You are able to create your own themes and import it into this system, which is what we ended up doing to accomplish our requirement of maintaining the look and feel of FARA’s other properties.

After examining all of the default API's and systems that Drupal has bundled into it's DBMS, we then arrive at the further customization that took place by adding additional components to bring in features that do not come by default in this DBMS. The added items are called Modules in Drupal, and are accomplished by simply placing the necessary files into the Modules folder, and then Drupal will recognize this new component automatically. After the Drupal instance notices the new module is placed into the correct location, you then need to manage the setup of the new
functionality through the administration page. This is done by first enabling the new module, which is simplified for most situations as just placing a check mark next to it. At the same time, if there is more than one feature built into a given module, you can enable or disable just those that you are interested in, and not forced into turning them all on just to get a few of the desired components. Not all modules are this simple to implement, but that is the ideal situation for new components.

With the module turned on, you then are able to customize these features in a few different fashions. The approach that best suits those that do not have a lot of technical skills with PHP or HTML is to use the customization pages that appear once a module is enabled, which is contained in the administration page. These are implemented through the module itself, and give you text boxes, radio buttons, or drop downs to alter settings of the module. This approach does not allow for the absolute control that other approaches can provide, but for most situations it is adequate enough to meet the necessary alterations that an administrator is looking to complete.

The second approach that you can take to customize the modules you add to the Drupal DBMS is to actually modify the PHP or HTML files contained in the module's folder itself. This more often than not is well documented with comments, so you can understand what each component or section is implemented to handle. With this approach you can customize every single detail you ever wanted for a module, but requires you to understand the various languages used to complete the construction of these files. Allowing these files to be edited at will, along with proper documentation like comments brings the ability to leverage development knowledge, as well as provide a way to alter modules so it will properly integrate with the website.

These modules are designed to cover a wide arrange of capabilities, be it adding completely new functionality or altering the core Drupal components. No matter it's intention, all modules are built on top of the various API's that exist within Drupal, and leverage these libraries to convey the tasks being attempted into a format that Drupal properly understands. For those that introduce new functionality, it will use the API's to store or retrieve the necessary information to the DBMS, and the actual work is completed within the module itself. For modules designed to alter functionality, there is a few different approaches that can take place for how it completes this task. One way is similar to the new functionality way, and just hook into the API's, while disabling or not even touching the core component that exists within Drupal itself. The second approach is to communicate with the subsystems that exist within Drupal, and either override or enhance the code that it uses to complete the task. This alteration of default functionality allows for something like the Forum module to get a more improved look to it, or add other new and interesting capabilities that your website is interested in obtaining.

**Process and Product Metrics**

In addition to the required individual time tracking, for our project we tracked a burn down chart and clicks per operation.

The burn down chart let us know how well we were progressing toward the goal on a feature by
feature basis. Combined with an overall task list, we could keep track of exactly how many features have been completed, how many are currently being worked on, and how many have yet to be started/assigned. This makes it very simple for us to track our progress as we are moving through the project. It also allows us to produce a simple graphical representation of the progress in the form of a chart, which maps the features to be accomplished along the Y-axis and the sprint number along the X-axis.

Our second metric was Clicks Per Operation, which measures how many clicks each task takes to complete in the site. This is important for measuring ease of use, and since ease of use is one of our primary goals, this metric is ideal for our project. In our testing, we have collected an average number of clicks it took for the testers to accomplish each task. We have compiled a list of the ideal numbers of clicks in the following table:

<table>
<thead>
<tr>
<th>Test Name</th>
<th>CPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegistrationTest001</td>
<td>2</td>
</tr>
<tr>
<td>RegistrationTest002</td>
<td>2</td>
</tr>
<tr>
<td>RegistrationTest003</td>
<td>2</td>
</tr>
<tr>
<td>RegistrationTest004</td>
<td>6</td>
</tr>
<tr>
<td>LoginTest001</td>
<td>2</td>
</tr>
<tr>
<td>LoginTest002</td>
<td>2</td>
</tr>
<tr>
<td>LoginTest003</td>
<td>2</td>
</tr>
<tr>
<td>NewsTest001</td>
<td>5</td>
</tr>
<tr>
<td>NewsTest002</td>
<td>14</td>
</tr>
<tr>
<td>NewsTest003</td>
<td>14</td>
</tr>
<tr>
<td>RealTimeCommunicationTest001</td>
<td>6</td>
</tr>
<tr>
<td>DonationTest001</td>
<td>1</td>
</tr>
<tr>
<td>CommentsTest001</td>
<td>10</td>
</tr>
<tr>
<td>CommentsTest002</td>
<td>7</td>
</tr>
<tr>
<td>CalendarTest001</td>
<td>2</td>
</tr>
<tr>
<td>CalendarTest002</td>
<td>2</td>
</tr>
<tr>
<td>CalendarTest003</td>
<td>4</td>
</tr>
<tr>
<td>CalendarTest004</td>
<td>13</td>
</tr>
<tr>
<td>CalendarTest005</td>
<td>2</td>
</tr>
<tr>
<td>ForumTest001</td>
<td>1</td>
</tr>
<tr>
<td>ForumTest002</td>
<td>4</td>
</tr>
<tr>
<td>ForumTest003</td>
<td>4</td>
</tr>
</tbody>
</table>
As a whole, we were able to implement almost all of the original requirements that were stated at the beginning of the project, as well as some additional features that were requested during the beginning of the project. There are a few features that are missing in the final version of the project that were deemed to be not as important as the other features. This was based on discussions with the sponsor as well as a user survey that we had users fill out early on in the project.

The only planned features that were left out were private messaging and having a user return to a previously viewed page from their previous session. The reason they were left out were due to time constraints mostly and the amount of features that were originally requested.
Along with this, there were only a few features that were added to the project throughout the entire project. The features that were added were a "want-ad system" and different home pages based on user statuses. These features were implemented in the final version of the project with minimal effort.

The final project has been turned over to the sponsor as of the writing of this document. The documentation has been finished in terms of test case creation and how-to documents for the administration of the site. The site has been migrated and is working on the sponsors web host with all functionality as stated above. Acceptance testing on the sponsors side was not completed, however, because it was agreed upon that they would conduct the tests themselves once the site had been migrated to their server. The reason this was not completed was because of the issues we faced in migrating to the new server, which stemmed from issues with the web host themselves.

**Project Reflection**

Our team was a mixed group of experience, where we each came to the table with a slightly different understanding of good process. But there is one thing we all can agree upon, SCRUM facilitated our worked well for our needs. It allowed us to keep progress continuous throughout the project, making sure everyone always had something they could add to the project. It also allowed the project sponsor to not get overwhelmed with the requirements gathering process. Another thing that worked amazingly well, was the fact that we set ourselves to two week iterative releases. This allowed for the ability to show the sponsors who were not in person to see tangible proof that the site was making gains. Another thing that worked well was communication within the team. Our team had individuals that all had the same, or very similar understandings about software development. We also beleive that the choices for the technical aspects of the project were well informed and chosen for the correct reasons. There was also the HP Virtual rooms that made it so we could not only communicate with our sponsors but the fact that we were also sharing live what we were doing. This allowed for things like discussion about bugs on the site to be walked through, and the ability to share documents whilst being able to edit them in real time. The virtual rooms also worked really well with the SCRUM methodology, being that agile processes work best when there is a customer in the room.

On the whole the project was a success, but there were failings. Our first failing was to make sure drupal easily worked with MS SQL. This didn't seem like it would cause an issue when we first looked into it, cause we researched and the Drupal site said that it was compatible with MS SQL. But this was not quite as true as they made us believe. It would have required quite a bit of configuration and we were already having issues getting Drupal working without the extra complexities of trying to make it work with tools that it didn't support fully. this issue was resolved by speaking with the FARA team and realizing that MS SQL was chosen originally due to FARA's and HP's current use of Microsoft products. There was also the issue of communication between our team and the sponsors. The main complication here was the fact that most of the sponsors were unable to be there in person. between this and the expectations for communication between our team and the sponsors was unclear. There was also issue with the fact that some of the projects tasks were quite a bit larger then origionally expected. It was
known that template development would be crucial to the success of the site, but it was unknown that is where most of the development occurred.

The project went well, but there are some things that we would do differently. First off, we would need to set clear expectations of what communication should occur between the development team and the sponsors. Keeping the sponsors more abreast to what we are doing would have kept them more aware of what we wanted from them in terms of ideas and requirement change. Then we would probably come up with a better system for time estimation. what would have worked well is spending more time on the requirements by possibly using estimation cards, and the technical knowhow of sponsors to facilitate development. Lastly we would want to do sponsor testing of functionality as it was released.

Our team recommends that FARA continues to work hard to attempt to cure this debilitating disease and we all feel like we have helped make a difference

References
1. www.curefa.org
2. www.drupal.org
3. www.mediawiki.org
4. http://www.palantir.net/files/general/images/matw-1_0.png