Lab 1 - Power Play Product Description

David Heeschen

CS411W

Janet Brunelle

February 5, 2016

Version 1
Outline

1. Introduction
2. Power Play Product Description
3. Identification of Case Study
4. Power Play Prototype Description
   4.1 Prototype Architecture
   4.2 Prototype Features and Capabilities
   4.3 Prototype Development Challenges
5. Glossary
**Introduction**

Any organization, of any size, cannot function adequately without competent decision makers correctly making necessary decisions. Even more importantly, it is not sufficient to wait idly for issues to arise. It is crucial for decisions to be made proactively versus reactively. To be able to curate potential opportunities, and mitigate potential threats. This is particularly important for companies and organizations that offer critical services to people, such as utility companies. Where an outage is not simply bad for business, but a potential public health hazard. Effectively making these proactive decisions requires foresight and domain specific knowledge cultivated through study and experience. Such knowledge and experience does not come without a price. Whether that is in the form of time spent furthering knowledge through education, both in a traditional learning environment or with hands-on experiential education. On-the-job training is also an option, but comes with more direct financial cost to the employer.

**Problem Definition**

This is the same issue that Dominion Power finds itself in during its annual planning activities. The actions taken by Dominion are costly, and have high impact to consumers. Poor decisions can lead to a host of issues, and having to hastily make important decisions. An electrical grid that supports millions of customers by a variety of energy sources does not manage itself. It takes a concerted effort by trained individuals to guarantee the success of the company. There is the significant task of maintaining the current infrastructure, but also maintaining and growing the business. To adapt to changing customer needs and energy requirements.
Currently, there is not an environment in which Dominion can make decisions in a type of sandbox where risks can be mitigated. So when decisions are made, they incur real world consequences to the business and the consumer. This presents serious issues for Dominion during the planning process as it significantly limits innovation. Actions that are not tried and true are quickly quashed by arguments that such risks are not worth it. The high cost of bad decisions produces a risk averse environment in an industry that thrives on innovation.

Furthermore, without the ability to accurately simulate potential situations, training the next generation of employees who will be in charge of maintaining a complex system is daunting. It is critical that employees acting in capacities that require difficult decisions to be made have the required knowledge. Successful companies and organizations have drastically failed as a result of incompetent management. Obviously, there will be some growing pains involved with any significant transition in corporate leadership, but for that reason it is imperative that proper training occurs.

For these reasons, Dominion has a high interest in expanding the skills of its current employees, but also in providing resources for prospective candidates interested in the field of energy management. These are the goals of Power Play. Power Play will serve as a virtual environment in which anyone can simulate managing an electrical grid in a game-like environment. This will allow the program to fulfill all of its intended goals for Dominion. Firstly, the training aspect of the simulation will lead to a more informed staff participating in the planning process. Secondly, the game aspect of the interface will potentially engage people previously uninterested in this specific domain. There is also the additional benefit in generating
favorable public relations with its customer. Dominion can leverage the existence of this program to generate good will between disenchanted customers.

**Project Description**

Thus, the purpose of Power Play is to provide an environment that will simulate the challenges of managing a complex and diverse electrical grid in a play like manner. This is a broad and general statement for a large complex problem, but at a high level this is the primary intention for the development of this software. Users can make decisions regarding the management of their simulated power grid while the simulation engine will provide feedback to the user. For instance, a user can expand the number of coal using power plants. The simulation engine could then potentially trigger an increase in coal prices. This will force the user to adapt to the updated circumstances, and additionally proving the importance of diversifying the types of energy fuelling their electrical grid. In this example, the training aspect of the simulation is strikingly apparent. The program also logs this information, and will eventually be able to provide troves of simulated historical data. An invaluable resource when the decision making moves from simulation to reality.

- **Key Product Features and Capabilities**

To successfully provide the simulation environment, the program will require a number of features. The program aims to provide a simulation environment for individual users. This necessitates that the program will be able to provide the standard suite of features for the programs users such as authentication, password reset, and assigning permissions to individual users. We have identified three groups of users for the program. The majority of users will fall under the general player category. As a player, users will only be allowed access to play the
game and their individual data. There will also be groups for game administrators and data analysts. Game administrators will be responsible for creating games and have a one-to-many relationship with players in their individual game instance. Game administrators will not have access to game play, but will be able to manipulate the simulation. For instance, in the previous example where users had invested highly in coal technology, the game administrator could raise the price of coal exorbitantly to force users to react. The final group of users, data analysts, is a blanket group that will cover any individual interested in accessing the data generated by the program, and will have no access to any of the gameplay features. The majority of individuals in this group will be internal users at Dominion.

The actual gameplay of Power Play will function in a similar manner to fantasy football. At each round of the game, users will be presented with a variety of scenarios. At this point the user will have the option to make their set of actions. Once the user confirms they wish to make those actions, that data is then sent to the simulation engine that will run against the provided inputs. The important thing to note that, interaction with the simulation engine is not simultaneous or directly interactive. The actions determined by the simulation engine are made by the cumulative effect of all decisions made. So, the results of those actions cannot be immediately computed. It will require time for the results of those actions to come into fruition.

Keeping with the similarity to fantasy football, there will be a ranking system to provide a competitive aspect to the gamification. We will be able to assign a point value standing for all users involved in a game. Users will then be able to see their standing compared to their peers. It will also serve an alternative purpose where users and game administrators will be able to gauge individuals progress as well as the overall progress of the group.
As previously mentioned, the program will also provide a degree of data analysis on user decisions, and the outcomes of those decisions. A key feature of the analysis is the ability for the employee to query the data set that we provide to them. An employee at Dominion could access and query this data. The data set will also be able to download the full set of data, or the result of some query to refine the results. This information could then in turn be loaded into a third-party application and repurposed.

Major Components

The major components will require hardware for the client and the server and a host of software needs. The server hardware simply requires a machine that will run Ubuntu Linux and Apache. At the time we cannot specify any more details since each potential customer will have different needs. Thus, any processor and memory requirements will be set on a case by case basis by the consumer based on desired performance, and anticipated traffic. On the client end, any traditional computer with a high speed internet connection and an up to date browser will suffice. A traditional computer is intended to specify any non-mobile device such as tablets or smartphones. There has been much progress over the years in porting web applications to native mobile applications, but a mobile application will not meet the intended goals at this time.

This program will be heavily dependent on third-party libraries and software. Since this program will rely heavily on JavaScript on the front-end it is an absolute requirement that the client have a modern and up to date browser. Without a capable browser, we can offer no guarantee that the program will run as expected. As the program is event-driven with major user interface features, we will be using a Javascript framework such as Angular or React along with
jQuery to reach our intended goal. It will not be possible to produce the intended behavior without client-side code to be executed.

The server will also require a significant amount of software. The system cannot run without server software installed. In this case we will be using Apache2 to communicate with the client. The program could also run on Nginx servers if so desired for increased performance. Additional software required on the server side is Ubuntu Linux, MongoDB, and Flask. Ubuntu will provide the operating system for the program to run in. The program could certainly run on any machine capable of supporting Python and Mongo, but Ubuntu’s package manager and ease of use will eliminate many of the issues associated with server administration. MongoDB was chosen over a traditional relational database management system for ease of use in communicating with the simulation engine. The simulation engine design was done independently of Power Play and during design they chose to use Mongo as their database. Flask is required to provide an API to the browser through which the game can push and pull data to the server.

Identification of Case Study

The potential end customer for this program will be Dominion Power, but the program and simulation engine could potentially be generalized for other use cases. Other customers could be other utilities companies whether specifically power companies or not. Power Play will initially be designed and developed to provide a front end to the simulation engine. The design and development of the simulation was made possible by a grant from Dominion Power to the College of Engineering and Technology at Old Dominion University. Guidance for work on Power Play will also be provided by students and faculty from ODU’s College of Engineering
and Technology. The principal advisor and mentor for the development will be provided by
Nathapon Siangchokyoo, a PhD candidate in engineering management with additional
contributions from Dr. Mamadou Seck. As domain experts, their guidance will be used to refine
the features provided by Power Play to maximize benefit to the end customer. Both will also
also act as liaisons between the group developing Power Play, and the engineering students that
will be working on the simulation engine since both systems are being developed independently
of each other.

**Power Play Product Prototype Description**

The prototype for Power Play will be to develop a front end for the simulation engine that
is currently being developed. It is important to note that the simulation engine is being
concurrently developed by another team, and the successful development of a user interface for
that simulation engine by no means indicates a completed simulation engine. Thus, the accurate
simulation of events in the game is fully omitted as a functional requirement. Since accurate
simulations will not be required in the prototype, data analysis will also be removed as a
requirement. The prototype will only demonstrate the minimal set of features on which a more
robust system could potentially be built given additional time and resources.

**4.1 Prototype Architecture**

The architecture of the prototype will directly mimic the architecture of the end product
since the elimination of certain features does not change the overall architecture. The prototype
is still a modern web application that will require numerous languages and software for its
execution. Additionally, as a web application there are two facets to the client server
architecture. The requirements for the program to be run on the client machine, and the necessary components for the program’s execution on the server.

For the client side, the program will rely heavily on modern front end development technologies. As such, the client must have a modern browser in which the client side javascript can execute. Without a capable browser, the application will most likely fail or produce unexpected behavior. Given that even the prototype is a web application of considerable size, the program will not be limited to using vanilla JavaScript for its implementation. The program will make extensive use of open source JavaScript libraries. Most notably, jQuery will provide an invaluable number of functions that will ease the pains of DOM manipulation, and AJAX calls.

The back end will be written entirely in Python 3 which will additionally use the Flask web framework. The sole purpose of the back end is to provide an API, application programming interface, to allow data to be pushed and pulled from the database to the client’s browser through jQuery AJAX calls. Since Power Play will be relying heavily on JavaScript, the argument to use JSON as a data store has some credence. For that reason, we will be using MongoDB that uses a form of JSON to store data. Mongo was also the database of choice for the simulation engine, and since both systems share a common data form communication to and from the simulation engine is eased considerably. Data does not need to be filtered or cleaned before being sent out.

4.2 Prototype Features and Capabilities

The table below provides an exhaustive list of all features that will be implemented by the prototype, as well as features that have been eliminated. As previously mentioned, this project is being developed independently of the simulation engine, and as a result some features have been eliminated due to that. Most notably, the prototype will not support any of the data
analysis features previously mentioned. The data generated by the program has little meaning to any user. The application will initially run off data populated by project members with no domain specific knowledge. For this reason creating game scenarios and accessing game results will only be partially implemented for the prototype.

<table>
<thead>
<tr>
<th>Features</th>
<th>RWP</th>
<th>Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Player Features</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Plant retirement</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>- Plant refurbishment</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>- Plant construction</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>- Viewable resources</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>- Viewable Environmental Impact</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>- Viewable Earnings</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td><strong>Administrative Features</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Create game scenarios</td>
<td>YES</td>
<td>Partial (hardcoded)</td>
</tr>
<tr>
<td>- Create and remove players</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>- Reset collected data</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>- Access player game results</td>
<td>YES</td>
<td>Partial (hardcoded)</td>
</tr>
<tr>
<td>- Player update notification</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><strong>Analysis Features</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Viewable decision data</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>
### 4.3 Prototype Development Challenges

The development of this prototype presents a number of challenges from a host of different issues. The first glaring challenge, is that the prototype and the simulation engine are being developed by two different teams with little contact with one another. Certainly, the amount of contact between the two teams will increase as progress is made, but it is a significant limitation. Furthermore, even when the prototype is completed, there is no guarantee that the simulation engine will be. Despite Power Play becoming fully functional, there is the very real possibility that the simulation engine will never fully come into fruition. Without a simulation engine for the prototype to integrate with and utilize the prototype is in effect rendered useless and further development of the project will be canceled.

Another similarly related development challenge is the integration of various 3rd party APIs. Most notably, the application will rely heavily on Google Maps for much of the interactive nature of the game. There is the possibility that the program may be pushing the limitations of what Google provides in the API. Even if it does, there is also the real risk that team members may not be able to find enough resources in the API documentation to actually implement some of the prototype features. This would also effectively render the project incompletable. Relying on Google Maps for much of the geographical representation in the game is a bit of a gamble,
but the best option at the moment. There are similar offerings from other companies, but significantly less robust than Google’s offerings.

Another potential technical hurdles are the limitations of Python and Flask. Python is a high level interpreted language. As such, performance is a potential issue, although unlikely, it is worth noting. Team members will need to take this into consideration as the project moves into implementation. In many cases, the naive solution will potentially not be an acceptable implementation for many of the features that the program will provide. As our data set grows larger over time, the prototype must be able to process that data and send it to the browser efficiently. If Python and Flask are incapable of processing that data, we will have serious scalability issues as the number of users and the size of the data increases.

Finally, with any web application we run into browser compatibility issues. What looks fine in one browser, may render differently in another. Browsers have different JavaScript engines that will affect performance on various platforms. This will present serious issues if not addressed during development. Compatibility issues must be kept in mind while we develop the user interface. Even beyond issues between browsers, we will also have to address the appearance of the user interface on various screen widths. The prototype will not be supported on mobile platforms so many of the issues associated with mobile devices is remove. However, even on medium to large screens there are large variations in size of the viewport in which the program will be running. At best, such issues become a minor user experience issue. At worst, they remove essential functionality from the prototype.

**Glossary**

**Flask:** Flask is a microframework for Python based on Werkzeug and Jinja 2.
**Python:** Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**Apache2:** Web Server

**MongoDB:** Schema-Less Database Management System

**jQuery:** JavaScript Library. Eases use of AJAX calls and DOM Manipulation

**DOM:** Document Object Model

**PHP:** PHP is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

**Javascript:** an object-oriented computer programming language commonly used to create interactive effects within web browsers.

**AJAJ:** AJAJ stands for Asynchronous JavaScript and JSON. AJAJ is a new technique for creating better, faster, and more interactive web applications with the help of JSON, HTML, CSS, and Java Script. Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.

**Spark Simulation Engine:** Spark Simulation Engine is a system being created to model and simulate different power distribution scenarios and events. This will provide the back end to the Power Play Project.

**Administrator:** Educational Instructors and Trainers at Dominion Power’s Training Facilities

**Analyst:** Data Analyst at Dominion Power

**Player:** Students and Trainees

References
