Case Study

Fortune 500 corporation transitions enormous **Power Builder application** to standardized web platform and architecture.

How a Fortune 500 (The company) was able to migrate over 8 million lines of Power Builder code to a standard Java / Web architecture using Mobilize.Net state of the art Artificial Intelligence conversion tools while complying with corporate policies for quality assurance and information security.
Business Situation

The company is a market leading supplier of software to manage tax reporting and calculations for Global 2000 enterprises.

This particular application was created with PowerBuilder on top of Oracle stored procedures; because it performs complex tax calculations, it contained—as code—vast quantities of business rules and algorithms that had been developed and debugged over more than a decade. The correctness of those calculations represented much of the value of the application. However, a key IT goal is to standardize all development and deployment environments to web and cloud. The level of specialized knowledge that was embedded in the application made a rewrite highly risky and economically unviable: Mobilize.Net was therefore chosen to modernize the application.

Market pressures required the transition not only to a more modern platform but also a complete re-architecture to ensure the application could be deployed in a cloud environment and could run essentially in any available browser. Functional equivalence, performance, and security were key requirements for the generated solution.

Moreover, the company was losing the ability to react quickly to market changes and demands, since it lacked a platform that would enable them to develop new system functionality in a swift, safe manner.

Consequently, a migration to Java / HTML5 was presented as the most reliable solution, given its benefits in terms of time/costs savings and maintainability. The company then engaged in a thorough solution analysis that included candidates from several companies, and selected Mobilize.Net as the official supplier for this and many other modernization projects, because of its extensive background and level of expertise. Mobilize.Net’s assessment tools and methodologies also allowed for the creation of a highly precise work plan to optimize resources and meet the specified delivery dates.
Challenges

The project consisted of over 6000 screens and more than 8 million lines of Power Builder source code. The level of functional equivalence after migrating could not be less than 100%, under the strong condition of preserving all business logic incorporated into the systems through years of operation and maintenance.

Specific requirements included:
- Move the application to a modern platform to enable agile, rapid development
- Maintain a familiar look and feel so that long-time customers wouldn’t be disoriented
- Provide an architecture that expands options – for example, moving the application to a SaaS model in the future.

After choosing Mobilize as the modernization partner, it was necessary to design a secure collaborative environment for development and testing. The geographical barriers as well as the nature of the applications required the creation of a communication strategy that would minimize the dependency on services only available in the company’s environment (physical devices, databases, etc.).

Additionally, appropriate mechanisms were required for protection of sensitive data to be used for sending and receiving data, accurate control of migrated and tested versions, and so forth.

Project Approach

To begin with, Mobilize's assessment tools and methodology were used to gauge the project and ensure that customer expectations and provider goals were realistic and attainable. The resulting analysis identified several particularities in the company’s applications and systems that were important to understand prior to the project kickoff.

The collaborative environment was optimized with Mobilize’s specialized methodology for project planning and execution, supported by various mechanisms of communication and data transfer.

The implementation of Microsoft SharePoint Server and connection to the company’s VPN facilitated coordination efforts between project teams, comprised of approximately 20 full-time consultants from Mobilize.Net.
The solution followed a clear conversion pattern. The GUI of the migrated application is generated using HTML5, CSS and JavaScript, using several libraries including Angular.js and Bootstrap.js. Every window and data window in the original application has a corresponding view file.

The client side of the application contains ViewModel objects for each one of the windows of the application. These ViewModel objects are created using JavaScript (in JSON format) and these objects are bound to the HTML controls of the GUI using Angular.js.

On the server side, the converted application has a set of Spring MVC controllers that represents each one of the windows of the original application; these controllers have endpoints to handle each one of the different interactions of the windows, and receive and return data in JSON format. The communication between the client and server sides is performed by a series of JavaScript libraries developed by Mobilize.Net that use JQuery to process the events on the client side, determine the changes to the ViewModel objects and send those changes to the server via Ajax requests using JSON objects.

Besides the controllers, on the server side the application has model objects that represent the elements of each one of the windows and are equivalent to the ViewModel objects on the client side. There is a set of libraries written in Java 1.8 by Mobilize.Net that take care of the synchronization of the objects sent from the client side with the models in the server-side, as well as handling logic to create instances of these elements and retrieving the necessary objects for each user’s session. The source code for these libraries is part of the migrated application, removing any post-migration dependencies on Mobilize for support or maintenance.

Once a request from the client side is processed by a controller on the server-side, instances of the models for the current windows are created and control passes to the business logic layer, where the actual logic of the application is (this is the logic that comes from the event handlers and functions of the original application). This logic interacts with the model classes instead of the user interface. The models on the server side replicate the hierarchy defined in the original application, and there is a set of libraries that replicates the built-in PowerBuilder functionality (things such as the binding of controls to elements in the data source, the life cycle of events of a data window and tracking of changes in specific records).

Additionally, during the conversion of the application all the logic that accesses the database is moved to a new layer that uses JDBC to connect to the database and execute any existing queries. This layer exposes a method for each one of the different queries in the application and is the only set of classes with direct access to the database.
The following diagram shows a high-level view of the proposed architecture.

- **Client-Side (Modern Browser)**
  - Client-Side Helpers
    - View Model binding
    - Serialization
    - ViewModel Sync
    - Multiple-Windows
    - Management

- **Server-Side (WebServer, Tomcat)**
  - Controller
  - Model
  - Business Logic
    - Business Logic

The communication between client and server is performed using AJAX in JSON format.

- **HTML & CSS elements generated from PB visual elements (Window and DataWindow)**
- **The helper Classes help replicate functionality (PowerBuilder) and allow achieving 100% automation**

- **There will be a controller and a model for each Window or DataWindow in the source application**

- **The business logic from the Source Application is kept on the server-side.**

- **Oracle, SQL Server, ... Application Database**
- **Session Database**

- **Server-Side Helpers**
  - Serialization
  - Model Sync
  - Session
  - State
  - Data Access
  - FileSystem Access

- **Power Builder helper Classes**
  - DataWindow
  - PB Libraries
  - Reporting/Printing/Export
  - Dynamic Window
  - Data Binding
Results/Benefits

Compliance with security policy information
During the planning and execution phases of the project, all regulations set by the company’s internal politics for information security were rightfully met, including confidentiality contracts with the provider’s personnel.

External collaboration model
A secure collaborative environment for testing and development was effectively implemented, which enabled The company to work with an external supplier even in highly complex projects.

Business logic preservation
Mobilize’s modernization methodology guaranteed 100% functional equivalence, limiting potential disruption risks and keeping intact the business logic contained in the systems. At the same time, the fact that the migrated application behaves identically to the original kept impact on learning curves for end users at a minimum.

Productivity improvements
One of the benefits inherent in automated software migration is that the time needed to debug the code is much lower than with a rewrite. This project was no exception, and Mobilize delivered top levels of quality and order throughout the process of source code managing.

Increased response capacity
With the deployed solution it is now possible to develop new system functionality more efficiently, allowing the company to be more agile in the face of rapidly shifting market landscapes.

New Application lets The company be competitive
The new web version of the application gives The Company much more flexibility including:

• Ability to easily add customers without requiring a complicated and slow installation
• Opportunity to quickly add functionality and instantly deploy to all customers
• Option to move to a SaaS model which will enable infrastructure cost savings.
• Full standardization of the development and deployment environment

On-time and On-Budget
The project was delivered On-Time and On-Budget and at 1/3 of the cost of a manual rewrite. This was the most cost efficient and less risky way to execute the project.
About Mobilize.Net

Mobilize.Net accelerates and simplifies the transformation of software applications to .NET, web, mobile, and cloud platforms. Millions of developers over more than 25 years have used Mobilize.Net technology to successfully modernize billions of lines of code.

Mobilize.Net solutions enable customers to reduce risk, cost, and time while moving applications to the platforms businesses demand today.

Mobilize.Net migration technology is Microsoft’s chosen solution for Visual Studio and MSDN customers.

The privately-held Mobilize.Net is based in Bellevue, WA, and is led by former Microsoft Corporate Vice President, Tom Button.

More case studies can be found at:
www.mobilize.net/Case-Studies/case-studies