

Implementing Low Code Solutions for Financial Surveillance and Reporting

Ryan Nicholos, Connor Maloney, Sebastian Rodriguez Traconis

Data Science (DSC05)



Abstract:

The adoption of low-code platforms has revolutionized financial surveillance by enhancing the integration and analysis of data across enterprise systems. Traditional methods for connecting disparate financial data sources often require time, technical expertise, and manual intervention, leading to inefficiencies and operational risks. Low-code technology addresses these challenges by enabling faster development of custom applications and workflows with minimal hand-coding. This allows institutions to automate data collection and reporting, seamlessly bridging multiple enterprise data sources such as ERP systems and siloed databases. The intuitive interfaces and pre-built modules empower stakeholders to contribute to processing automation, reducing the reliance on IT teams and enabling rapid iteration to address client needs. In support of the USN, our team set out to implement low-code techniques to streamline data validation, anomaly detection, and regulatory compliance by creating dashboards between two enterprise data sources.



Introduction

Often the issue with data engineering and analytics is the lack of authoritative data or data with the proper hygiene to support making informed choices. But what happens when there is an abundance of data available that is siloed away in desperate locations? How do analysts tap into data lakes to unleash the stored knowledge from these enterprise data resources and harness it to provide insights that ignite critical decision making? Enterprise Resource Planning (ERP) sites have long reigned supreme as the data solution in the commercial, private, and government sectors. ERP systems are incredibly powerful tools providing amazing features and custom solutions. However, an issue that project teams typically come across when utilizing multiple ERP sites comes in the form of connecting data between two or more sources to achieve a wholistic picture of the data available and uncovering the story that data is telling.

Prior to low-code applications, project teams would require high level programmers with immense technical skills and background to develop bespoke integration code to access, connect, and manipulate data pulled from ERP systems. The Herren Team presents the typical business cases and key considerations for implementing low code solutions, the technical framework to low code cross walk structures, lessons learned and best practices from applications deployed within the United States Navy (USN).

Business and Technical Justification for Low-Code Adoption

While traditional software development is the foundation to the technological advancements that currently enable the low code revolution, the development process still faces cumbersome procedural challenges. The traditional development process typically kicks off with a review of key business requirements and working objectives that the project software aims to achieve. The design phase of the project outlines the technical framework for the software team to begin developing the application which includes the overall architecture, data structures, and supporting algorithms that power the application's business logic. Project developers determine their preferred programming language and write code to build out the application. Testing is conducted regularly throughout the processes or once the planned application build is completed depending on the preferred development method. The approach most practiced today is an iterative Agile approach vice the Waterfall Method. Testing ensures the application's code is working properly before integrating external systems with custom Application Programming Interfaces (API) and Graphical User Interface (GUI) to connect the project and interface with external data sources. Finally, the software is deployed to a chosen infrastructure with dedicated network configurations, databases and servers before ultimately going live for the end user.

Even with the advent of Agile methodology enhancing the speed at which the process can deliver, traditional software development still faces challenges with resourcing. Traditional coding usually takes a large team of skilled developers and technical resources to build applications from scratch to achieve the business goals of the client, an extended return on investment since the teams for bottom-up coding projects require a higher level of expertise, as well as longer deployment times in comparison to low-code applications. Additionally, finding the requisite level of expertise is becoming more difficult than



ever with the US Labor Department estimating that the global shortage of software engineers may reach 85.2 million by 2030.

Low-Code applications allow development teams to further increase deployment cycles with platform interfaces that offer increased mobility, allowing improved responsiveness in user interfaces that implement prebuilt components which enable faster iteration.

Low-code applications accelerate deployment with platform interfaces that enhance agility, improve user interface responsiveness, and enable faster iteration using prebuilt components. Built-in security features protect client data with advanced encryption and role-based permissions to control and limit who can access what parts of the application which is especially important when connecting enterprise data resources together. A core aspect of low-code development is automation, utilizing workflows to transform business processes reduces manual effort typically found in enterprise software. Low-code platforms integrate seamlessly with DevOps, enabling continuous integration and continuous delivery (CI/CD) pipelines for efficient development, testing, and deployment. They also offer case management, centralizing business processes, and workflows to support product and client maintenance. While all of these are characteristic of low-code help to prove its worth, the data integration aspect is the primary attribute that promotes low-code from a business and technical justification aspect. The most common scenario for our USN clients is viewing financial data and performing surveillance on financial metrics that use multiple legacy authoritative ERPs. Low-code platforms simplify this with connectors, APIs, and data mapping tools, enabling seamless access to databases, third-party services, and legacy systems.

Key Considerations for Implementing Low-Code Solutions

Determining how best to improve the financial surveillance in client spaces comes down to problem solving: what is the problem statement? What tools, technologies, and skillsets are at our disposal to address the problem? Solving these problems requires two essential things: a deep understanding of our client's domain and the challenges they face, as well as technical expertise with the tools and technologies available.

When starting work in a new client space, evaluating and selecting a low-code platform may be required before any development work can begin. There are numerous technologies available in this space, including industry stalwarts such as Microsoft Power Platform, Appian, and ServiceNow, in addition to many other platforms and companies, both large and small. Many factors will influence which software is best suited for each client's scenario, but most platforms include the basic tools required: the ability to create custom business applications, store business data in a relational database, automate workflows, and visualize data.

The decision-making process typically comes down to the following:

Cost: Low-code platforms are generally procured through a platform subscription model. Usually, a company will pay a monthly fee per user of the platform, with tiers of features and capabilities ranging from basic to advanced. Sometimes 'per-app' or 'per-workflow' licensing is also an option, where a subscription grants access to a single application or workflow for a set of users.



Integration: The ability to seamlessly integrate with existing client software, such as Microsoft Power Platform's native integration with Outlook and Teams, provide an easier path to user adoption, as clients are already comfortable using those existing tools. It's also important to select a platform with pre-built and/or the ability to create APIs for integration with existing databases. A platform that can easily retrieve data from a company's existing authoritative data source will greatly simplify implementation and maintenance of solutions.

Developer Experience: If developers already have experience with one platform, that will inherently reduce speed to implementation. However, commonalities between the different platforms provide a generally shorter learning curve to learn a new platform if a developer has worked with a different one.

When a platform is selected, the first step in implementing a low code solution is to work with the client to develop a problem statement and break it down into focused business requirements. These requirements will drive the implementation framework. Herren development teams use root cause analysis to pinpoint problems and leverage technical expertise to define realistic solution requirements.

Another key point to consider when selecting a platform and planning a solution is how the solution will integrate with existing databases, tools, and technology. Many of these platforms have built-in APIs that can retrieve data from existing databases, a very useful feature for maintaining authoritative data sources. There is also the capability to develop custom APIs, if one does not exist. The solution should capture and store as little data as possible; maximizing the use of existing authoritative data sources will greatly simplify future development and maintenance.

When planning and developing a solution, an important consideration to keep in mind is data governance and security. The developer should be mindful of whether the data is sensitive in any way (proprietary, PII, etc.) and take the necessary measures to ensure strict access to data for those necessary. Low-code platforms have built-in cybersecurity by nature of being Software as a Service (SaaS) technology, but within the platforms there are additional tools and features to control permissions to data. Developers should work closely with clients to make sure users only have access to the necessary data to conduct their work tasks with nothing sensitive and potentially harmful.

Low-Code Implementation Framework

Team Herren implements low-code solutions with an emphasis on customer priority and feedback. Our project team employs an Application Lifecycle Management (ALM) framework for developing and maintaining the low-code solutions deployed. This framework includes agile principles that help deliver tested capability incrementally, while establishing a feedback mechanism with customers to ensure requirements and priorities are met.



Figure 1. Low-code Software Development Lifecycle Framework

1. Plan

The project team works directly with process owners and organizational leadership to identify where a process or system can be improved using a low-code solution. Utilizing our financial management domain expertise in conjunction with a deep understanding of the pain points described by the customer, the project team begins to define requirements for a solution to improve and/or automate a given process or system.

First, the team looks at the business data involved in the solution, where it comes from, and how it's stored. In scenarios where the data is not coming out of an existing database, our team will develop a data model for the process, to ensure a sustainable data storage solution is created in the platform's relational database. Next, we begin selecting specific tools we plan to use, such as business applications, automated workflows, and/or dashboards, and then (for complex solutions) developing a mock-up, or 'wireframe', of the product for presenting to the customer. The planned design can be iterated as many times as necessary to ensure the customer receives the best product possible.

Our project developers input the required development work into an ALM tool we developed internally to track all features through deployment. This tool allows us to plan agile 'Sprints' and prioritize features and products according to customer needs.



2. Develop

Once a design is confirmed with the customer, our team begins the development work to bring the plan to reality. Our teams develop solutions in a separate 'development' environment than where the solution is hosted for end users, so developers are free to make changes without fear of breaking the 'live' product. We are constantly keeping up to date with the latest features and capabilities available on our low-code platform, which allows us to provide our customers with the most efficient and capable product. With a proper plan and design framework from the previous step, development can proceed without hesitation. Our teams regularly meet with customers throughout this phase to ensure development is on the right track and to test features as they are developed.

Typically, one or two junior developers will do most of the development work, with a senior developer guiding development and available to help as needed. Working sessions and brainstorming the best way to meet the requirements regularly produce the most innovative and creative solutions.

3. Test

The Herren developer team has implemented a 'peer review' process where members discuss and walk through the logic developed in the solution internally prior to the customer receiving the product. A senior developer or group of developers will test the features developed, simulating end users, and record any missing features or bugs. Regression testing conducted on existing features will also take place at this time if new features introduce changes that could affect existing logic. This helps to identify bugs and technical issues while also providing an opportunity for more experienced developers to teach junior developers best practices.

Once the quality assurance "QA" review is completed, the team requests the end users of the solution to demo the solution as one final chance for testing features prior to release. This includes publishing the solution in an environment that users can test the solution themselves, in a 'User Acceptance Test'. This is also an iterative process, where the customer can request as many updates as necessary to ensure the end-product meets the needs of the users. We also track requests received here in our ALM tracking system, assigning them to developers and prioritizing them against other existing requests. Any small changes will go through the development test deploy steps before release; larger changes, which are rare at this stage, may need to be deployed in a future release.

4. Deploy

At this point, the solution is published to the 'production' environment where the customers can use it. We track the release number, associated with all the features developed, in our ALM tracking tool for configuration management purposes.

We also work with customers at this stage to determine the easiest path to adoption among users. Depending on the customer, this can include developing documentation like SOPs (started in the develop or test phases), recording a video demonstrating how to use the solution, or doing a live demo to the users and holding a Q&A session.



5. Maintain

The 'Maintain' phase of ALM includes working with the customer to identify additional requirements and bugs that require fixing. We typically build feedback features into our solutions so users of the solutions can easily submit an idea for improvement or when they've encountered an issue. These requests are then brought up to the process owner, and these additional requirements and fixes flow through the ALM framework just like the initial development of the solution.

Best Practices for Successful Low-Code Implementation

Implementing Low-Code solutions in the financial data and cost world can bring many benefits if done correctly. As one of the leading low-code domain experts servicing the USN, our team provides insights into our successful practices in delivering impactful solutions.

Team Herren development teams practice early and regular (weekly) engagement with stakeholders to align business objectives and end user needs. For low-code solutions specifically, the importance of stakeholder communication sets the stage to identify any disparate data sources, legacy systems, and integration requirements that the proposed solution will need to be compatible with. This ensures application platforms seamlessly connect with existing infrastructure and comply with business requirements.

Herren project teams regularly participate in collaborative internal communities of practice to explore, discuss, and understand the best application of available low-code features. The leading industry low-code platforms (Power Platform, Appian, ServiceNow) all push regular updates to their technology. Staying sharp on the "art of the possible" enhances project teams' creativity and ensures timely application with the most impactful solution.

Herren project teams practice a systems engineering perspective and approach to implementing low-code solutions. By partnering the techniques and methodologies with low-code technologies, our solutions excel in financial management by offering data analytics, compelling visuals, streamlined data input, and process automation. Success depends on knowing which solution to apply and when to provide meaningful data insights.

Case Studies and Real-World Examples

Team Herren primarily supports USN and many of the directorates within the Program Executive Offices of Integrated Warfare systems (PEO IWS) as a trusted partner to identify process improvement initiatives and digital transformation candidates to implement low-code solutions and workflow automations. The following case study highlights one of the many real-world examples that our project teams have conducted and delivered for the Ship Self Defense System program IWS 80.



The Problem

Naval Sea Systems Command (NAVSEA) is composed of seven PEOs, all of which are responsible for keeping track of their budget and spending. Specifically, our USN client within PEO IWS 80 had the primary goal to provide improved surveillance on program fiscal year obligations (funds planned) and expenditures (funds expended). The number actuals for the program's budget, obligations, and expenditures are maintained and reported through Navy Enterprise Resource Planning (ERP).

To ensure that each PEO funds are sufficiently allocated, and resources are expended properly, a plan for obligations and expenditures is created within the legacy system known as NAVSEA Enterprise Planning System (NEPS). Within NEPS, obligations and expenditures can be planned per month for the current fiscal year.

Apart from the plan created in NEPS, each executive office is given benchmarks from NAVSEA and The Office of Secretary of Defense (OSD) that are meant to ensure all funds are expended on time and these are applied to Navy ERP. Ultimately the two authoritative enterprise data sources are siloed and do not talk to each other.

Our client's desired outcome was for our team to find an efficient way to pull data reports from both legacy data sources. Provide improved surveillance between the funding planned in NEPS and the actual funding figures in Navy ERP. Lastly, enable our client to more effectively reference data from both sources simultaneously with an automated data refresh capability.

The Solution

Plan

Applying system engineering approaches to understanding the client's business requirements laid the foundation for implementing low-code technology. Through weekly working groups the Herren project team established the operational requirements that the low-code solution is set to address. In collaboration with the client our team identified the pertinent stakeholders along with the functional requirements and integration needs. This low-code solution was chosen over creating excel VBA coding to provide scalability and a more seamless data transfer between legacy data sources. Our team decided to utilize the native software suite of the USN, Microsoft Power Platform.

Develop

The first step in creating a report is to connect to the necessary data sources. Low-code programs give one the ability to choose from various data sources such as pdfs, SharePoint, SQL servers, websites, etc. In this example, the data sources of interest are the ERP ZRQs report which shows actual obligations and expenditure amounts formatted into rows using Budget Structures, the NEPS CAR report which shows the monthly plan for Obligations and Expenditures by Task (TPS), and lastly the NAVSEA/OSD



benchmarks. In this case, all three data sources are Excel based documents located in a shared environment using Microsoft SharePoint. Low-Code programming allows for direct connection to a SharePoint environment and its contents. This means that instead of having to connect to these three data sources separately, we can create one single connection and have access to all.

After establishing a connection to the data sources, the next step is data transformation. Low-code platforms provide built-in tools for efficient data preparation and processing, enhancing data validation, reducing noise, and improving overall report quality. In Power BI, this functionality is facilitated by Power Query, which enables operations such as removing columns, filtering rows, and replacing values. Additionally, Power Query allows for merging and appending multiple datasets, a critical feature when working with the ERP ZRQs report, which contains up-to-date obligations and expenditures. By stacking multiple ZRQs pulls and incorporating an import date column, users can track both current and historical data.

For the NEPS CAR report, which outlines the monthly plan for obligations and expenditures, consolidation is unnecessary as only the current plan is relevant. However, further transformations are required due to the report's extensive number of columns. To streamline the data, only key columns—such as the TPS number and monthly planned obligations/expenditures (October through September)—are retained. The “month” columns are then unpivoted into a single column with corresponding values for obligations and expenditures. Finally, the values are pivoted to create distinct columns for obligations and expenditures.

Month	1 ² ₃ Obligations Total	1 ² ₃ Expenditures Total	1 ² ₃ Obligations DC	1 ² ₃ Obligations RE	1 ² ₃ Expenditures DC	1 ² ₃ Expenditures RE
January	0	0	0	0	0	0
July	0	0	0	0	0	0
June	0	0	0	0	0	0
March	0	0	0	0	0	0
May	0	0	0	0	0	0
November	0	0	0	0	0	0
October	0	0	0	0	0	0
April	0	0	0	0	0	0
December	0	0	0	0	0	0
February	0	0	0	0	0	0
January	0	0	0	0	0	0
June	0	0	0	0	0	0
March	0	0	0	0	0	0
May	0	0	0	0	0	0
November	0	0	0	0	0	0
October	0	0	0	0	0	0
April	0	0	0	0	0	0
December	0	0	0	0	0	0
February	0	0	0	0	0	0
January	0	0	0	0	0	0
March	0	0	0	0	0	0
May	0	0	0	0	0	0
November	0	0	0	0	0	0
October	0	0	0	0	0	0
August	0	0	0	0	0	0
December	0	0	0	0	0	0
February	0	0	0	0	0	0

Figure 2. Pivot View of NEPS Data (Client Data Scrubbed)



A key advantage of low-code platforms is the automation of these transformation steps. Once configured, these steps are reapplied automatically whenever the Plan (NEPS) or Actuals (ERP) data is updated, eliminating the need for manual intervention and ensuring efficient report maintenance.

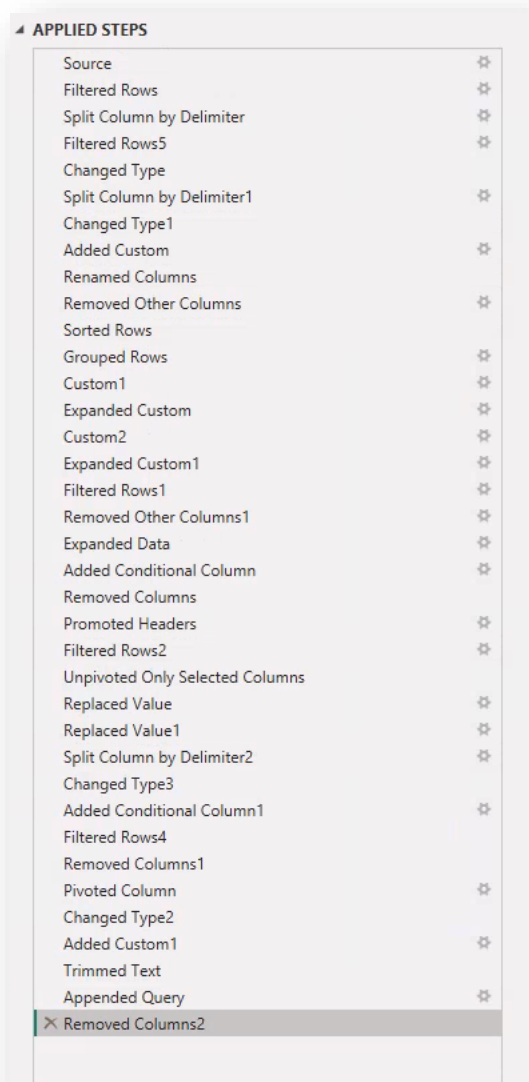


Figure 3. Applied Data Transformation Steps (NEPS)

Once data is transformed and loaded, the next step is to model the report, which involves establishing relationships and creating calculated columns and measures. Low-code programming facilitates the development of complex models by enabling seamless integration of multiple data sources. Establishing



relationships allows independent datasets to be analyzed collectively, which is essential for comparing actuals against planned values.

To integrate ERP and NEPS data, a crosswalk provided by the client maps each ERP Budget Structure to its corresponding NEPS Task Planning System (TPS) number. Relationships are established by linking the crosswalk to ERP through Budget Structure and to NEPS through TPS Number, ensuring a many-to-one relationship. This structure allows multiple instances of the same Budget Structure in ERP to align with a single entry in the crosswalk, and the same principle applies to TPS numbers.

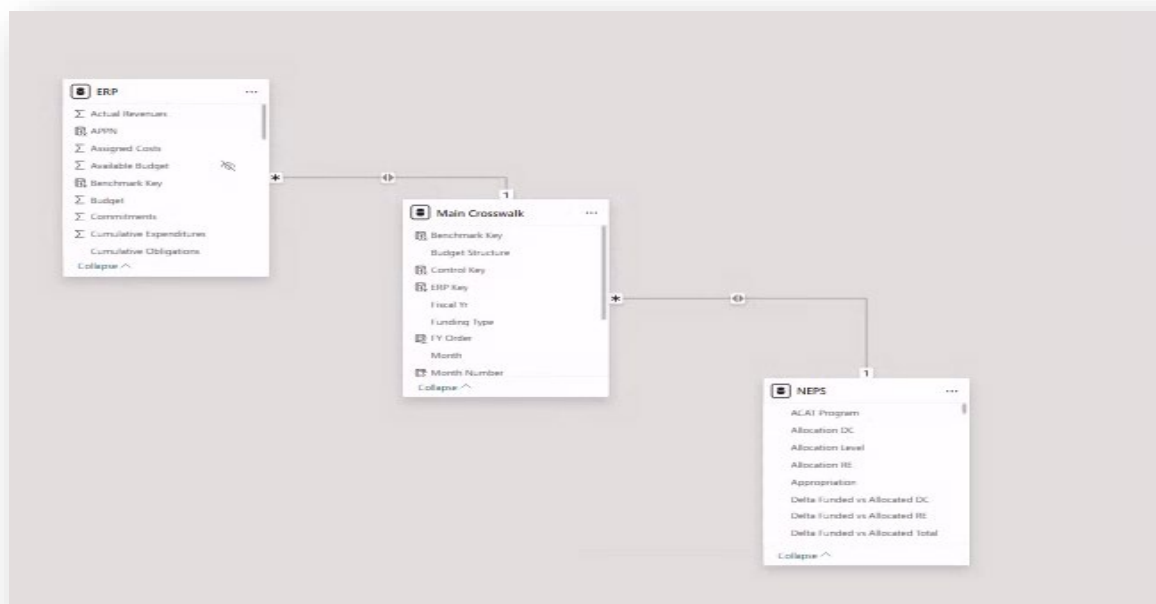


Figure 4. Crosswalk (NEPS)

Once relationships are in place, calculated columns and measures can be created. Low-code platforms simplify this process, functioning similarly to Excel formulas. The consolidated ERP dataset includes multiple ZRQs reports across various dates, supporting historical trend analysis. However, to ensure up-to-date reporting, a calculated column is implemented within the ERP dataset to flag the most recent ZRQs pull, returning "Yes" for the latest data. Reports can then filter current obligations and expenditures accordingly.

To facilitate performance assessment, a calculated measure divides actuals by the planned value, yielding a percentage of completion. This metric provides stakeholders with a clear comparison of actual expenditures against projections, enabling data-driven decision-making.

The final step in report development is visualizing the data to effectively convey insights. Low-code platforms like Power BI simplify this process through drag-and-drop functionality, enabling rapid creation of visualizations such as pie charts, scatter plots, and tree maps. While visualization is often considered



the easiest step, selecting appropriate visual representations is critical for accurately depicting data trends and relationships.

Additionally, low-code platforms support conditional formatting, allowing for customized visual enhancements. In this example, a bar chart was used to compare actual values against the plan, with a marker indicating the planned target. Conditional formatting was applied to color-code bars—green, yellow, or red—based on their proximity to the plan, enhancing interpretability and highlighting performance variations.

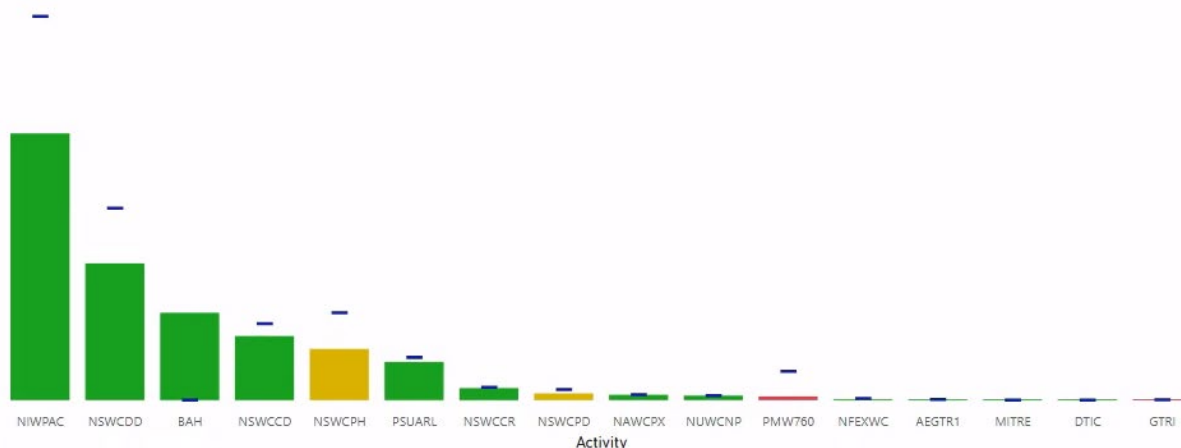


Figure 5. Navy ERP vs NEPS Data with Conditional Formatting Applied

With the ability to create a consolidated ERP dataset, a calculated measure was created to calculate the average burn rate for obligations and expenditures. Using this measure, it was possible to create visuals that attempt to forecast future spending patterns and align them to the current plan. This visual seeks to provide further insight to see if a program is on pace to meet their plans. With conditional formatting, the bars that are forecasted are colored gray, while the bars that show actual up to date values are blue. A line in the chart, corresponds to the current plan. This conditional formatting improves the overall quality of the report and ensures the data can be understood.



Figure 6. Monthly Navy ERP Data (Actuals) vs NEPS (Planned)

Testing

As visualizations are being fleshed out, the data can be simultaneously verified. This step requires an independent thorough cross check and meeting with clients to confirm that the data is reported as expected. To validate the ERP, specific Budget Structures are looked up in the ZRQs report to see if the values in the excel file match the value reported in Power BI. The same procedure is done for NEPS using the TPS Numbers instead. Meeting with Budget Financial Managers, the report was further validated by ensuring the data reported behaved as expected and nothing seemed out of place.

Deployment

Once data validation has been complete and the values reported from the dataset can be trusted, the report is now ready to be made into a deliverable product. The low-code platform Power BI includes Power BI Service, a cloud-based platform that allows users to publish, share, and collaborate on interactive reports and dashboards. It is here where security measures can be applied as reports can be shared to specific people/groups. Permissions can be set as well to limit who is able to edit or view any published reports. On top of that, Power BI allows users to incorporate Row Level Security (RLS). RLS allows users to restrict data access for specific users, ensuring they only see the data relevant to their role or permissions.

Maintain

Data is ever evolving, growing, and constantly needing to be updated. As clients continue to upload new data into the report, it is important to ensure that data continues to be correctly reported. This task is made easier by setting up and utilizing a Power BI feature known as Alerts. With Alerts, when a report is updated with new data, Alerts can be set to ensure. The project team set out to automate the process of pulling refreshed ERP and NEPRS data and ingesting the data into the client's local operational data lake. Through the implemented cross walk framework this current information is fed to the data visuals displayed in the modeled report. Herren support maintains the deployed application and iterates functionality with regular updates and services any ad-hoc design requests to meet updated project requirements.



The Takeaway

The key takeaway that ultimately achieved client satisfaction was the ability to provide automated reporting with the most current data from their authoritative data sources. The process time to gather updated data was improved from 6 hours report to 30 minutes enabling a higher frequency data pulls from once every month to being conducted as frequently as the source data is refreshed through the Alerts feature. The main benefit is that we can save time on this process, create more compelling visuals, and spend less time looking through data and more time analyzing and making informed decisions driven with the data.

The use of low code platforms to perform the analysis of NEPS vs ERP has demonstrated the ability to be very effective. This implemented low-code application performed complex transformations and relationships with relative ease. Traditionally, one would have to have both data sources at hand (which can be quite large X# of lines) and cross reference which rows in NEPS correspond to the rows in ERP and then capture the data in local excel spreadsheets to further analyze. From there, using lookups in excel, the numbers between NEPS and ERP would be compared. This task alone was tedious (6 hours) especially as data was updated (every other day), these steps would be repeated.

Transforming and modeling data are two crucial steps in creating reports that ultimately determine the speed, accuracy, and effectiveness of the report. Low-code platforms allow users to create transformations on their data in applied steps.

These steps are automated to be re-applied to the data even when it is updates, saving a huge amount of time as transformations only must be configured once. This is essential, especially when dealing with large data sets, it ensures the report will run efficiently and that the data being reported is correct. Creating calculated columns and measures are also as simple as creating formulas in excel. With the ability to select from various built-in visuals, building a report can be done in just a matter of minutes. On top of that, customization of visuals can be applied such as conditional formatting, that will further enhance the report. Low-code programming allows businesses to spend more time making decisions and less time sorting through data.

Future Trends and Innovations in Low-Code Development

The no-code, low-code conversation rests on the premise that technology should speed up innovation by empowering non-tech users to build custom applications with simplicity and ease. Industry leaders predict that developers outside formal IT departments will be comprised by 80% of users of low-code and no-code development tools by 2026, up from 60% in 2021. Leveraging pre-trained models and AI-driven tools for tasks such as code completion, bug detection, and optimization, streamlining the development process is another trend that our project team believes the industry will head towards.

A trend that should be followed intently is one where administrators are also able to track changes made to applications such as database logins, edits, and deletions. Near term low-code trends are likely to include an increase in features such as audit logs that provide more transparent visibility into what



collaborators have been working on. This would allow administrators to monitor changes and investigate any suspicious activity and to ensure that government financial data and data at large are secure and accounted for.

Specifically for government agencies this technology will unleash the data from within legacy systems and support hybrid cloud models to integrate and enhance automated services with prebuilt API connectors and triggered workflows based on real-time data driving operational efficiency. This capability combined with the cybersecure, and compliant nature offers enterprise legacy data a renewed strategic value.

Conclusion

Data is everywhere, and the ability to understand it has become ever more essential in a data driven world. In this digital age, the volume, variety, and velocity of data has increased; making the need for data literacy a necessity for personal and professional success. Low-Code platforms grant the ability to digest a variety of data sources, transform the information, and craft informing insights with it. If applied and implemented with systems engineering techniques, low code platforms can help businesses and organizations spend less time gathering and looking through data, and more time making informed and impactful business decisions.

Standalone systems can lead to siloed data, meaning employees can struggle to work together in larger businesses as they cannot share data across departments. Meanwhile, a unified solution built on one common – yet open – platform allows data to flow seamlessly through multiple applications, making it easier to establish workflows that benefit businesses. Failing to acknowledge and consider how the power of low-code/no-code applications can help is a missed opportunity to enhance efficiency, accelerate development, and drive innovation within the organization.



Authors:

Ryan Nicholas

Associate

Herren Associates

Maritime Plaza II, 1220 12th Street SE, Suite 310, Washington, DC 20003

Email: ryan.nicholos@jlha.com

Mobile: 757-374-2272

Connor Maloney

Associate

Herren Associates

Maritime Plaza II, 1220 12th Street SE, Suite 310, Washington, DC 20003

Email: connor.maloney@jlha.com

Mobile: 716-698-2816

Sebastian Traconis Rodriguez

Consultant

Herren Associates

Maritime Plaza II, 1220 12th Street SE, Suite 310, Washington, DC 20003

Email: sebastian.rodriquez-traconis@jlha.com

Mobile: 610-655-7039