IBM Redbook Solution Guide Insert

Database Modernization – The key to long-term value
Adsero Optima™ (AO) Roadmap

What if?

- ✓ You only had to maintain 10-15% of your lines of code yet provide the same functionality to your business users? And in parallel, deliver improved data quality, data integrity and performance?
- ✓ Your code became more manageable, maintainable and modern as a natural consequence of this process?
- ✓ You could achieve all of this for a fraction of the cost of replacing your applications or performing the effort manually?

Background

Understanding the AO approach requires acknowledgement of three fundamental observations taken from many modernization projects:

1. The enormous amount of time, effort, money and intellectual property invested in your IBM i, i5/OS and OS/400 based applications and the fact that the functionality rendered by these applications provides a significant competitive advantage to the business.
2. The inherent value encapsulated in the heritage application, as well as your database structures and the underlying rules and relationships that have evolved over the decades.
3. The validation rules and database relationships that are critical for application extensibility and development of new applications, modules and functions.

Because of the above three fundamental observations, AO focuses on providing a realistic, low-risk, non-disruptive, gradual process to unlock the inherent value and massive investment made in your accumulated customer and business data. This includes the investment made in developing the business rules and exposing this knowledge and the rules for re-use by the business.

Based on the above assertions, it then becomes imperative that we analyze why business users consider their LOB (line of business) applications as “legacy”. It is important to analyze this more acutely than usual, which superficially suggests that the User Interface/User Experience (UI/UX) is our biggest inhibitor and cause for user anxiety.

Having said this, we recognize that the UI/UX is an important contributor to legacy perception. However, it is not the main reason why business users have become disillusioned with legacy (heritage) applications.

Close analysis of the primary reason why installations have moved away from the platform show the common thread is a “lack of agility” which surfaces very quickly. A lack of agility usually presents itself with business users, who are frustrated with the slow response to IT changes and the ability to exploit new business opportunities. On careful analysis, it quickly transpires that IT is working

exceedingly hard; doing their utmost, but they are fighting a losing battle, due to immense maintenance backlogs.

A look at maintenance operations rapidly highlights the factors causing this excessive maintenance burden and lack of agility. It is caused by the way applications were necessarily developed in the 1980's and early to middle 1990's. Co-incidentally, this also happens to be the timeframe when the bulk of IBM i based ISV's and applications first came into being.

Contributing to this, IBM has been very successful in protecting and insulating the IBM i installed base from underlying changes in the IBM i system hardware and operating system. The platform is so reliable and forgiving, that applications developed in the 1980's happily keep on running, despite massive advances in the hardware and software constructs. This competitive advantage became its "Achilles heel".

Other suppliers would have demanded a rewrite of the applications.

Added to this was the maxim "If it isn't broke, don't fix it" that some programmers and IT Managers believe in, not recognizing that most of these applications have been "broken" from an architectural perspective ever since the introduction of the ILE programming model. The platform insulated us from this harsh reality.

Most of our heritage IBM i applications are characterized by huge structured monolithic programs which were developed over the years. This eventually created complex system environments, massive maintenance backlogs and frustrated users. Within these monoliths, 80% of the lines of code dealt with database relationships, also known as entity relationships, and database validations.

The monolithic programs allowed little or no separation of function, which made maintenance increasingly problematic. Every program that manipulated the database had (or at least were supposed to have, in order to ensure integrity) the same 80% of code repeated repeatedly. This led to an enormous amount of duplication, which was the only option in the 1980's and early 1990's.

Changing database relationships, adding fields or changing validation rules meant finding every single instance of that "rule" being used throughout the entire system and updating it – a very complex, time consuming and potentially error prone process; the inhibitor for agility.

In our experience, heritage applications encapsulate the competitive advantage of business, but critical questions need to be answered in order to make informed business decisions about the future:

Is my application worth modernizing?

If you are considering modernization, you need to know if your legacy applications are of value to the business. Do they (your applications) provide significant value considering:

a) The business success globally of companies using the platform;
b) The competitive edge encapsulated in the legacy application
c) The total cost of ownership that is being delivered to clients, suppliers and partners and;
d) The cost and risk of the viable alternatives?

If not, it does not make sense to modernize them, and it becomes a simple choice of retire or redevelop.

Based on experience and analysis of many installations that have moved off the platform, operational constraints and a lack of agility are largely responsible and are consistently being highlighted as the most significant factors. However, many organizations often regret replacing the IBM i applications due to a critical loss of required functionality. In addition, companies soon realize a dramatic decrease in availability and reliability were compromised because of the platform change.

If there is significant value in your legacy applications and you have determined that this needs to be retained, you need to modernize those applications to a modern database environment such as DB2 SQL. To begin, a careful analysis of the critical operational and strategic constraints needs to be identified:

1. Does the application provide a competitive advantage (e.g. a unique order entry process, special stock allocation and stock management algorithms)? List these in your environment.
2. Can your current business processes be improved?
3. Will the system allow this improvement?
4. What is inhibiting service delivery models for clients, suppliers and partners?
5. Document the cost structure to deliver the applications.
6. What is the functional fit of your current application? As a rule of thumb, off the shelf applications will deliver between 60 – 75% functionality.
7. Are the constraints you have experienced due to a lack of functionality or service delivery?

How much should we modernize?

Be very careful of the trap: “Analysis Paralysis”. You do not have to modernize your entire application, as many people suggest! Your objective should be to deliver maximum value to the business at all times. There are two important considerations that will help you determine how to extract maximum value:

1. Identify the 20% of application function that generates 80% of all transactions.
2. Perform a careful analysis of your maintenance or change requests as this will highlight application functions and programs that demand higher levels of maintenance that cause most frustration.

Start with the above two elements and you will soon identify where the greatest ROI can be achieved. In parallel, you will gain confidence with the methodology and approach.

In our experience, about 50 – 60% of your application will normally be modernized. The balance simply does not provide you with sufficient value (ROI) to even invest the effort. Additionally, we would recommend you implement a standard policy and criteria, which guides development staff to consider modernizing the code involved of most maintenance requests, as a standard. This will allow you to gradually modernize your system while performing routine maintenance. A coherent modernization strategy and plan should form part of the consultative process.

What options exist to best achieve the desired results?

In our experience, the only lasting modernization approach has to start at the database definition level.

Any other adjustments or maneuvers are tactical at best, providing only a brief respite. In the long term, these do not remove the barriers to a permanent solution. This is key for unlocking and reclaiming your heritage.

It is imperative to get your database definitions from the old DDS definitions into DDL. This is the first and foundational requirement to start long lasting modernization. AO will then facilitate moving the bulk of your relationships and validations gradually out of your application logic into the database engine. You want DB2 to do all the "heavy lifting" work for you, allowing you to focus on delivering innovative business solutions and logic.

This will open up a considerable amount of value and benefit, which can be achieved with:

- minimum disruption to your users,
- the lowest risk,
- greatest of ease,
- as fast as possible,
- yet deliver a significant return on investment

What business benefits will we achieve by modernization?

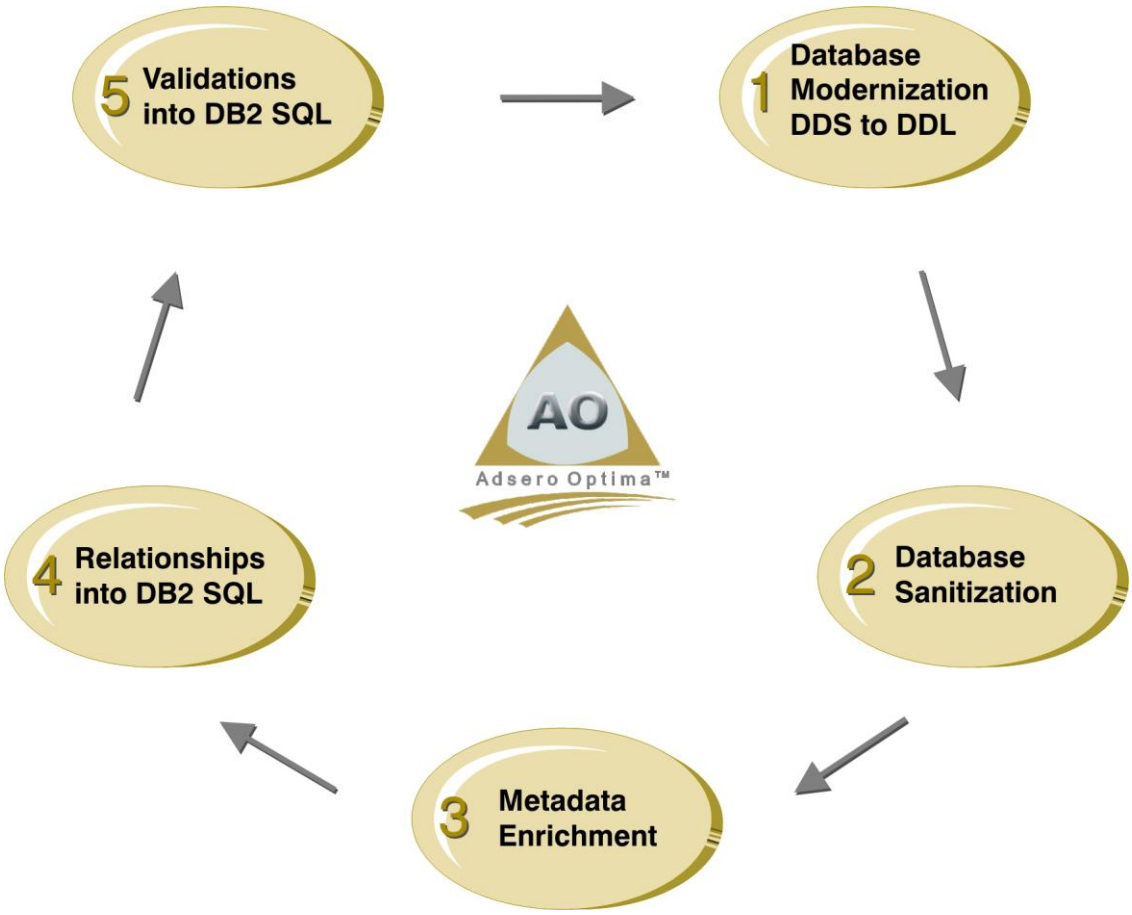
Cape Gate decided to modernize rather than replace their IBM i applications resulting in costs that equated to just 18% of the total hardware, software replacement and services costs of a replacement solution. That in itself was compelling to them. Momentum re-architected a single function as a trial to test the AO approach in their commercial application and reduced the run time on a long running job from more than 20 hours every month down to 20 minutes.

In addition, embracing change, implementing DB modernization provides the following additional benefits:

- ✓ Leverage latest DB2 SQL advancements
- ✓ Achieve exceptional cost savings:
  - 15% to 18% of expected replacement costs;
  - 10% to 20% to perform a manual modernization if at is at all possible. Larger installations simply are not able to implement a manual modernization, with some installations requiring on average a project timeline of almost 19 man-years.
- ✓ Gain access to a large young technical resource pool for future systems maintenance
- ✓ Benefit from re-invigorated developers and end user resources.

- ✓ Significant potential workload savings.
- ✓ A modern new database structure and methodology for on-going application development

Your database structures are the KEY to your future applications and modernizing your heritage



The first step in a long-term modernization process is to convert the database schema from DDS to DDL, unlocking a host of additional functionality in the SQE interface. This process can be achieved transparently to your heritage applications without the requirement of recompiling any programs (a few caveats exist – see AO Inspector output).

This process can be achieved with or without the need for surrogate logical files (AO supports both native as well as surrogating as viable migration methods). Surrogate logical files masquerade the change to the underlying database, allowing legacy systems to access the new database files without the need for recompilation. Surrogates are beneficial when new applications are being implemented, and legacy applications remain unchanged. Should you, however, aim to leverage the competitive advantage and value of your heritage applications, approximately 80 per cent of the lines of code (all lines of code implementing validations and enforcing data relationships) currently in your legacy applications will eventually end up in the database engine. By eliminating surrogate local files, you end up with a far more efficient approach to enabling long-term modernization.

In the majority of cases, the use of surrogates is unnecessary and arguably detrimental to future modernization and development efforts. AO uses unique technology to affect the DDS to DDL conversion.

The aim of this exercise is to build DDL from the cloned DDS structural metadata without changing level IDs in any way (which is a significant requirement when migrating and upgrading DDS to DDL in phase 1, which facilitates an easy, non-disruptive, low-risk process that is entirely transparent to legacy applications). The following animation will provide a more concise description of what AO does: http://www.adsero-optima.com/nutshell-msp

Additionally the following FREE tool called AO Inspector is available to assist you in analyzing your database, highlighting all the considerations during your modernization project:

http://www.adsero-optima.com/content/adsero-optima-inspector

AO Inspector

The objective of AO Inspector (AOI) is to provide factual information about the size and complexity of your (selected) schemas and databases, which will assist during your planning process and to highlight specific information from your database(s). This information is used to define tactical solutions, and provides the basis for your modernization strategy.

AOI will also highlight the state of your database, providing rich statistics and insight into your underlying database structures in a single place. Of particular importance is statistics that will allow you to gradually convert database structures to structures that conform to a true relational database model.

AOI Output for scoping effort and to highlight areas that will require attention includes the following, based on your selection criteria:

Total count of DDS defined physical files and DDL defined tables, total count of DDS defined logical files and DDL defined views/indexes and TOTAL count of all database constructs. As a rule of thumb, an experienced database scientist can manually generate DDL constructs from DDS using standard tools (iNavigator or the API's) at an approximately average rate of 2 hours per one physical file and all its associated logical files, excluding the migration of the data.

Additionally, AOI will extract and present all the following information:

- ✓ Unsupported DDS functions in DDL (PF):
  - ***Keyed, Not Unique***
  - Program Described File
  - Multiple Members
  - Alternate Collating Sequence
  - Zero Members (usually FRF)
  - Keywords
- ✓ Unsupported in DDL (LF):
  - Derived Fields
- ✓ Physicals/Tables (Issues) – individual counts and lists for both DDS and DDL
  - ***Arrival Sequence***

    - Level Check = *NO
    - *PUBLIC not *EXCLUDE
    - Re-Use Deleted Records
    - Not Journalled
    - *NULL Key Fields
- ✓ Constraint Definitions – individual counts and lists for both DDS and DDL
    - Total Constraints
    - Primary Keys
    - Unique Constraints
    - Check Constraints
    - Referential Constraints
- ✓ *BEFORE Triggers Definitions – individual counts and lists for both DDS and DDL
    - Total *BEFORE
    - *BEFORE/*INSERT
    - *BEFORE/*DELETE
    - *BEFORE/*UPDATE
- ✓ *AFTER Triggers Definitions – individual counts and lists for both DDS and DDL
    - Total *AFTER
    - *AFTER/*INSERT
    - *AFTER/*DELETE
    - *AFTER/*UPDATE
    - *AFTER/*READ
- ✓ DDS Logicals – total counts as well as Select/Omit at LF level
    - Views
    - Surrogate Views
    - Joins
    - Multi-Formats
- ✓ DDL Logicals – total counts as well as Select/Omit at LF level
    - EV Indexes
    - BR Indexes
    - Views

Additionally, AO also retrieves program observability statistics, to highlight potential considerations (part of the modernization effort is to move to the latest release of the operating system, with IBM i V6R1M0 as the bare minimum when complete).

Of particular interest are two statistics that are retrieved (please see preceding bold, underlined, italics items). Both express and highlight design and coding practices of 1980's and early 1990's. This was prior to the major DB2 database enhancements made on the platform. (Kindly refer to the solution guide insert discussing Adsero Optima™ Foundation for more in-depth detail of how AO facilitates implementing a true relational database from heritage database constructs.)

It should be recognized that although we have had access to a relational database engine on IBM i since the announcement of the platform, most installations and ISV's did not use it as such. The reason for this is application age and the elapsed time during which enhancements for DB2 for IBM i were made (25+ years). Added to this is the maturity and acceptance gained by relational database technology globally for which the IBM i and its predecessors, gained global acclaim.

As a result, the database engine made little or no enforcement of database validation rules. However, it is implemented in application logic at every instance whenever that database file is used (that is the theory of what is supposed to happen). The same applies to database relationships. This

means that our relational database is achieved by both DB2 and by the application logic. In our experience, this is the biggest cause for our agility constraints.

Should you exploit and leverage the latest capabilities of DB2 on IBM i, you can achieve three things relatively quickly:

1. Reduce the code base of your application by as much as 85% over time.
2. Improve your agility, allowing you to compete head-on with your competitors and provide a significant platform for true application innovation.
3. Leverage your competitive advantage and unlock the enormous value hidden in your heritage application's intellectual capital.

Conclusion

Based on extensive experience, it is clear that massive value remains in our heritage applications and that compelling business justification exists to modernize heritage application assets. This modernization process has to start at the database layer, as approximately 80% of our heritage functions belong in the database engine in the modern commercial application development paradigm.

Adsero Optima™ achieves this by:

1. Facilitating the (mostly automated) migration from DDS defined database structures, to native DDL defined database structures as an immediate initial step. This becomes the absolute foundation for ANY and ALL subsequent modernization.
2. Consultation to determine a modernization strategy and plan, and to highlight future development potential.
3. Consolidating and sanitizing metadata and structural metadata, as an immense amount of duplication and inconsistencies have been integrated within your applications and the underlying database due to decades of neglect.
4. Introducing referential constraints into the database, with fundamental knowledge recovered from your applications. This immediately starts to improve data integrity by orders of magnitude.
5. Removing user interface constraints (strategic partnerships), to facilitate multi-channel delivery of application functionality to any current or new UI delivery channel.
6. Recovering validation rules from the code base, facilitating the introduction of these validation rules as triggers directly into the database. This provides a process that will facilitate consolidation of all validation rules, improve data integrity and agility and achieve single instances (as opposed to multiple instances) of the validation rules. Additionally these rules will be implemented and consumed consistently throughout the system, regardless from where this is accessed.
7. Providing a gradual recovery and modernization of true business unique logic, this will represent the underlying competitive advantage encapsulated by your heritage application.

Supported platforms

This solution is supported by platforms with the following features:

- The AO Roadmap can be applied on ANY release of IBM i, or earlier releases of OS/400 and i5/OS, although some manual work will have to be performed.
- AO Inspector requires IBM i V5R4M0 as minimum.
- AO Foundation requires IBM i V6R1M0 as minimum.