The standard AO Modernization Roadmap, customized to individual installation requirements for maximum ROI

1. Modernization Education
    a. Need to understand why modernization is necessary, what benefits it brings and how to do it in order to succeed.
2. Plan Modernization Priorities
    a. Identify highest impact tables (files)
        i. 20% of application function represents 80% of all transactions.
        ii. Examine maintenance records to identify functions with highest churn.
        iii. Identify seldom-used functions (archiving, annual year-end processes) as extreme low-priority "not worth the effort" to avoid wasted resource.
    b. Adopt a "fix when touched" philosophy
        i. Incorporate modernization time into other standard maintenance/enhancement efforts.
        ii. Modernize tables and application code as the normal change process calls for updates.
3. Modernization cycle for each table
    a. Sanitize Data Dictionary elements
        i. For each field in the table, create or select a Master element (if not already done)
        ii. Review the details of the Master element and update as needed
        iii. Associate all other dictionary elements that refer to the same data item with the Master element
    b. Normalize the table
        i. Refactor the table as appropriate into discrete, smaller tables to isolate individual data items into logical containers, each with a unique key.
            1. Additions, deletions, and modifications of a data item occur in just one table and then propagate through the rest of the database using a foreign key into the single table.
            2. This eliminates redundant replication of data items that can otherwise potentially become unsynchronized and helps identify rogue data and orphaned records.
        ii. Where necessary, normalization may temporarily require:
            1. Working on more than one table to properly normalize the data
            2. Using an after-trigger to temporarily maintain the "old" version of the table in parallel for programs that cannot be modified in the current timeframe due to scope and schedule (preferred); or
            3. Creating a temporary joined view representing the "old" version of the table (only feasible if the unmodified programs only read the data).
    c. Extract relationships between tables from application code
        i. These become constraints.
    d. Extract validations from application code
        i. These become triggers.

   ii. Validation code likely exists in multiple programs. Conflicts in processing between programs will need to be resolved within the trigger code and validation code removed from the application.

e. Create I/O Servers to separate application code from database operations.
   i. Provides uniform I/O operation and abstracts the data implementation from the application. Application code should not contain direct I/O, but use calls to I/O servers.
   ii. I/O Servers for each table implement Create/Read/Update/Delete (CRUD) operations
   iii. Enterprise Servers and enterprise services can provide more complex function against multiple files where processing requires API-like operations.