Astadia™ THE MAINFRAME TO CLOUD EXPERTS

Mainframe Modernization:

The balanced Path to Success

Organizations spend years developing mainframe programs that become critical to their success, but they need constant upgrades to keep up with the rest of the world. Mergers, joint ventures, and other conditions may also demand changes, but exclusive mainframe features can delay or halt progress. A balanced solution preserves existing value while improving it. Astadia provides unique services focused on Unisys® mainframe modernization and migration.

Through its collective expertise, experienced consultants, and proven tools, Astadia offers a low-cost, reliable way to move forward. Consult with the best, review the choices, and build the reight solution. Your mission depends on it.

Contents

Executive Overview

With all of the disruptive technologies introduced over the past several years, it is easy to forget that legacy systems still deliver most of the services we use every day including the mobile and online services so critical to your digital transformation. The truth is that you can't become a digital enterprise without the business processes and data contained on your mainframe. Unfortunately, legacy systems are the biggest roadblock to the progress you need to make.

This whitepaper is focused on the unique challenges of migrating a Unisys infrastructure to an open-systems environment. It is based on experience gained during dozens of conversions completed over more than 20 years in the industry. Our goal is to provide enough detail on migration alternatives, risks, and planning requirements for you to have informed discussions at the executive level to build consensus on next steps.

Highlights of this report include:

- While many Unisys platforms use COBOL, architectural differences between generations makes careful planning critical. Unisys COBOL may contain numerous extensions that make it proprietary to specific hardware. For this reason, Astadia developed the Rules-Based Transformation Engine which converts proprietary Unisys COBOL to industry-standard Micro Focus® COBOL.
- Unisys mainframes also use Algorithmic Language to handle low-level operating routines and the code is specific to the Unisys operating system. Open systems, on the other hand, use languages such as C, Java, and .Net. Currently, no reliable automated processes exist to convert this code to C, Java, or .Net used in open systems so the code has to be rewritten.
- The proprietary Unisys Data Management System (DMSII) is a hierarchical database that includes backup and recovery tools. Migrating and preserving this data to an open-systems standard requires validation, conversion, and even possible modifications to the targeted destination architecture. It is not easy and requires extensive planning.
- DMSII also supports occurring items that programs can access with a subscript. Most relational databases, however, do not support occurring items. There are several methods for handling these, each with its own advantages and disadvantages which must be weighed against the overarching strategy.

The report also includes details on several of the proprietary tools Astadia has developed over the years to overcome many of the migration challenges, including OpenMCS, Astadia's message control system for Linux, Unix and Windows, that provides the necessary transaction processing features of Unisys COMS to support migrated applications.

We hope you find this document useful. Migrating your legacy infrastructure may represent one of the most challenging accomplishments of your IT career and is a critical step in your digital transformation. Choosing the right partner to assist you will be one of the biggest decisions you will make. We hope to hear from you.

Competing in a Global Economy

The provision of unique, high-quality products and services is no guarantee of success in this increasingly global economy. Organizations vie with lean competitors for market shares that demand the same innovation and quality at a lower cost. Organizations must be flexible to exploit the rapid development of new products in response to constant technological change. They must stay ahead while managing risk and cost. They must balance modernization with the preservation of mission-critical resources.

Nature and history abound with examples of migration. In nature, species migrate to survive. Historically, migrations have occurred in response to adversity and opportunity. The common thread for all migrations is change. Similarly, organizations now face rapid technological changes that require movement from legacy mainframes to newer platforms.

The Dilemma

When it comes to mainframe systems, the decision to remain competitive is easier said than done. For example, the proprietary nature of Unisys mainframes is incompatible with current software applications that could increase productivity and profitability. This incompatibility leads to a dilemma: “Do we continue to write applications for the existing mainframe or move to an open-systems architecture?”

Any decision to change the mainframe can run into several roadblocks: 1) Staffing. The age and complexity of many mainframe systems often require time and expertise not available through current staffing. 2) Risk. Organizations cannot afford the loss of legacy data, the degradation of mission-critical applications, or the interruption of profitable services. 3) Economy. The cost of writing new applications or purchasing licenses often exceeds the immediate cost of preserving current system architecture.

Nevertheless, internal and external changes continue to drive the urgency for change. Business partners implement systems that are incompatible with the current architecture. Groups within the organization require business intelligence unavailable through the current database design. Customers switch to organizations that can provide enhanced services at lower cost. The time for change has arrived, but any solution must pass an acid test to meet the criteria for success and added value.

The Criteria

Any attempt to move from an old mainframe to a new system requires consensus building and buy in. Financing is not the only challenge. Members at all levels of the organization will either reap the benefits or live with the consequences. The successful solution meets the following criteria:

- **Stable**: The solution supports millions of queries and operations during peak business hours without stalling or crashing.
- **Inexpensive**: The solution’s value does not exceed its implementation and maintenance costs; it costs less than maintaining the current mainframe system.
- **Flexible**: The solution supports current applications and provides adaptable technology that anticipates future updates.
- **Powerful**: The solution works at lightning speed and increases the ability to exploit valuable data.
- **Efficient**: The solution provider engages and disengages quickly while keeping costs low and ensuring success.
- **Standardized**: The solution and its implementation is compliant with the industry’s best practices and compatible with internal and external clients.
- **Intuitive**: The solution provides the convenience of modernization without sacrificing well-known business models, interfaces, and rules.

Few organizations can meet these criteria in isolation. The intricacy of moving from Unisys to an open system requires a systematic approach that is well equipped and informed by experience.

The Solution

With its unique combination of expertise, experience, and technology, Astadia offers a reliable, low-cost solution with these benefits:

- Modernization of existing legacy applications without the cost of rewriting them
- Automated migration and renovation that reduces time to market
- Integration of legacy applications with open-systems technologies
- User-friendly interfaces that simplify training needs
- Cross-platform standardization and consolidation

Mainframes and Open System Platforms

On the surface, mainframe architecture seems relatively simple: a centrally located computer processes data through an input/output subsystem and stores its computations in memory (Figure 1). At the other end of the mainframe are printers and terminals comprised of a keyboard and a monochrome monitor that only displays alphanumeric characters. The terminals communicate with the mainframe through protocols recognized in a communication box that has a processor and memory of its own.

For all of its apparent simplicity, mainframe architecture can be extraordinarily complex. For example, mainframes process thousands of transactions per second and require an extensive infrastructure to house and maintain. Hundreds or perhaps thousands of terminals require a message routing scheme that can quickly prioritize and route transaction requests.

This mainframe architecture is not as prevalent today as it was in the late 20th Century; however, it remains a viable solution for current and future business requirements. Large corporations still use mainframes because they provide powerful business intelligence and support many high bandwidth applications such as banking and telecommunications. Mainframes are also more secure, reliable, and scalable than distributed networks.

Nevertheless, several issues have increased the cost of maintaining traditional mainframes. For example, the

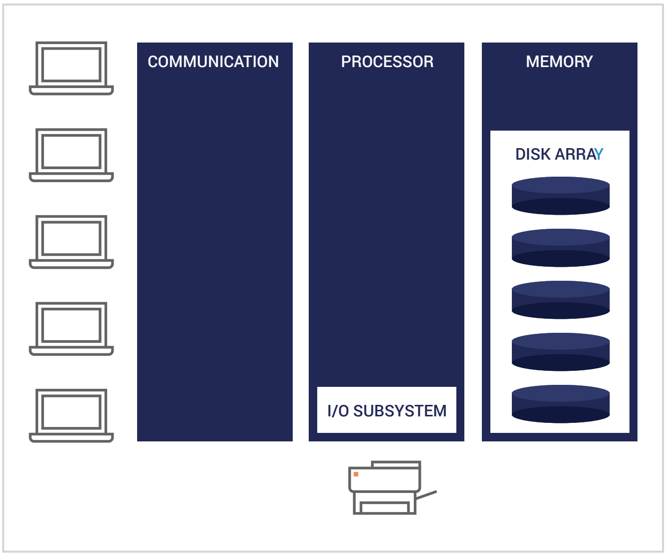

Figure 1: Typical mainframe architecture

lack of flexible licensing creates compatibility issues for organizations that require nonproprietary add-ons and updates. Workarounds often require the programming of expensive applications to accommodate new requirements.

Maintenance is also expensive, because the mainframe is often the nexus of a large-scale infrastructure that requires expert care. For example, older mainframes are often housed in cabinets or rooms that require precise environmental controls.

Hardware and software complexity also requires hiring expensive technicians and programmers throughout the life of the system, a problem compounded by the potential shortage of skilled personnel to implement the solutions and maintain these powerful systems. Many qualified COBOL programmers have reached or will soon reach retirement age. Perhaps offshoring can address some of these concerns, but the lack of adequate educational resources could still create a shortage of qualified personnel.

Unisys mainframes

Organizations cite the following reasons for moving away from Unisys mainframes:

- Internal and external partners have moved to other mainframes or open systems
- The total cost of licensing Unisys applications has increased beyond all expectations
- The proprietary nature of the Unisys mainframe inhibits third-party add-on technology
- Aging and retirement have created shortages of qualified programmers and technicians

Astadia offers proven solutions for migrating Unisys mainframes to other mainframes and open systems. These solutions are the result of combining Unisys expertise with the experience of many migrations and a suite of automated tools. For this reason, Astadia is the ideal partner for the modernization and migration of Unisys mainframes.

IBM mainframes

Large organizations often target their Unisys migrations to newer mainframes such as IBM® z/OS®. The desire to move forward is matched by the desire to retain the speed, power, and volume of the large mainframes. Buyouts can also cause organizations to target IBM mainframes. For example, a large corporation might acquire a smaller organization that runs on Unisys. As the newly acquired unit works its way into the new corporate culture, it will have to migrate its unique, legacy applications from Unisys to IBM.

Migrations from Unisys to IBM mainframes carry a unique set of requirements. For example, IBM mainframes impose a four character limit on trancodes not present in Unisys mainframes. Messages that work with T27 terminals in Unisys do not work with 3270 terminals in IBM. Consequently, every migration from Unisys to IBM requires specialized knowledge and tools. Organizations that work with Astadia gain the collective resources of its IBM expertise and a suite of tools specifically designed for Unisys-to-IBM migrations.

Open-systems platforms

The Software Engineering Institute (SEI) at Carnegie Mellon University provides the following definition for open system:

> A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered components to be utilized across a wide range of systems with minimal changes, to interoperate with other components on local and remote systems, and to interact with users in a style that facilitates portability.[1]

Open systems conform to well-documented standards and avoid the proprietary issues so prevalent with mainframes. Companies can evolve quickly through add-on technology, with the freedom to leave development time and costs to another party. In most cases, organizations need only purchase, install, and periodically upgrade the add-on. Apart from a few configuration tasks, little or no programming is required. All of these factors can lead to significant savings and an agile response to paradigm shifts.

However, migration to an open system is not without its pitfalls. While application licenses are more expensive for mainframe environments, they are often easier to administer. Organizations may also be reluctant to leave behind their extensive licensing investments. The purchase and installation of multiple thirdparty licenses for each member of the organization often seems daunting.

Open systems also come with a limited batching facility. Batch processing permits the completion of jobs without human intervention, often automatically monitoring the use of system applications and scheduling jobs according to the availability of resources. While batch processing is native to mainframe systems, open systems require the purchase of third-party software to replicate this necessary functionality.

1 Retrieved from http://www.sei.com.edu/opensystems/faq.html, on August 31, 2007

The move from Unisys mainframes to open-systems platforms requires careful planning. While the potential exists to reduce costs and build compatibility, its realization depends on a thorough analysis of the following alternatives:

- Replace the applications with commercial, off-the-shelf packages
- Reengineer the applications with alternative languages
- Rehost the applications or migrate them to a new platform
- Improve the applications with the infusion of new technology

The analysis should assess the cost, risk, and impact of each alternative until it identifies the right plan of action. Equipped with the latest tools, Astadia can quickly discover the value of every application and provide comprehensive system reports to aid the decision process. The way forward does not require the sacrifice of profitable technology investments.

Modernization Requirements

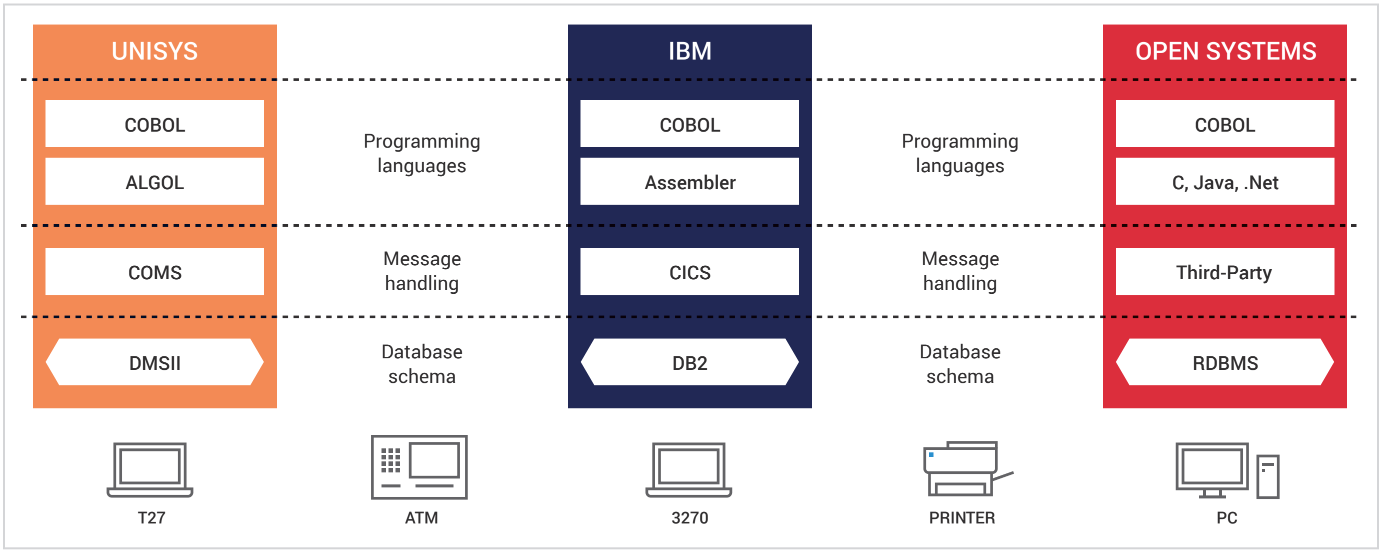

Figure 2: Mainframe and open-system software

Architectural differences between platforms often determine the appropriate migration strategy. As shown in Figure 2, each platform uses different programming languages, databases, message handling facilities, and peripherals. The best way to plan for any mainframe migration is to use Astadia's code transformation, rehosting, and database conversion services.

Code transformation

While many platforms use Common Business Oriented Language (COBOL), architectural differences between them require a careful migration strategy. Programmers typically use COBOL or a 4GL to write programs for mainframes. Mainframe applications may target a hierarchical, platform-specific database. COBOL usually contains numerous extensions that make it proprietary to specific hardware environments.

One of the main challenges with the migration of Unisys applications is to ensure open-systems compatibility without sacrificing years of architectural development. For this reason, Astadia developed the Rules-Based Transformation Engine, which converts proprietary Unisys

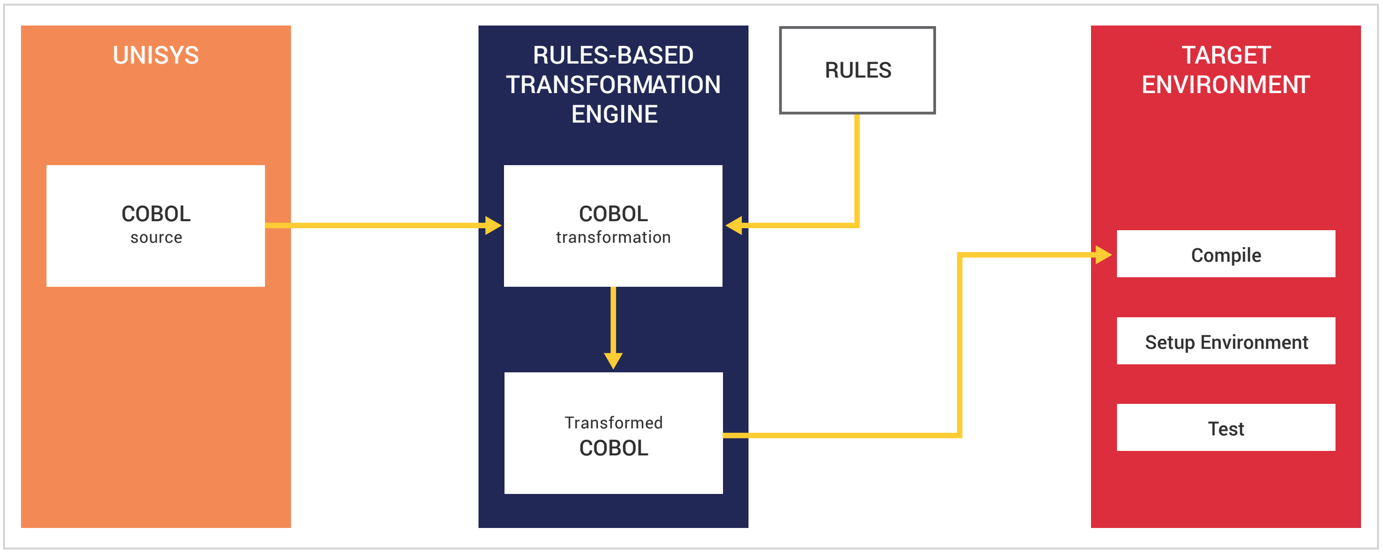

Figure 3: COBOL transformation

COBOL to industry-standard Micro Focus® COBOL (Figure 3). IBM COBOL is easier to migrate, because it is less embedded and more compliant with the American National Standards Institute (ANSI). Consequently, migration challenges with IBM COBOL are platform specific. Astadia employs the “Lift and Shift” strategy to ensure compatibility with the new environment.

Unisys mainframes also use Algorithmic Language (ALGOL). This language handles calls at a lower level than COBOL, but it works only with Unisys and its code is specific to the operating system. Assembler is IBM's machine specific code for handling low-level operating routines. Open systems use languages such as C, Java, and .NET. Currently, no automated process exists for converting Assembler or ALGOL applications. In most cases, Astadia develops COBOL or C/C++/C# to rewrite ALGOL and Assembler functionality.

Rehosting

Code transformation represents only the first challenge of migrating from Unisys mainframes to other platforms. The new platform also requires preparation. For example, the new system must emulate the legacy system's handling of transactions, processing items, and terminal messages. The name for this aspect of migration is rehosting.

Trancode routing

The accommodation of transaction codes is one of the most critical requirements in the rehosting of Unisys applications. Transaction codes (trancodes) are a sequence of characters within a message that direct a program to execute a certain portion of its transaction processing logic. For each window, valid trancodes can be defined and associated with programs; however, IBM mainframes impose restrictions not found in Unisys, and open systems require a third-party solution for trancode routing.

Preprocessing and postprocessing items

Another rehosting requirement is the ability to handle preprocessing and postprocessing items. The Unisys Transaction Server for ClearPath MCP Programming Guide provides the following information about processing items:

> A processing item is written in ALGOL and typically addresses a unique task that can be used by multiple applications and can be used many times within an application…Processing items can be used to preprocess messages before they reach a direct-window program and postprocess messages after they leave a direct-window program. For example, you can use preprocessing to format a menu of an application, while you might use postprocessing to format and print a bill.

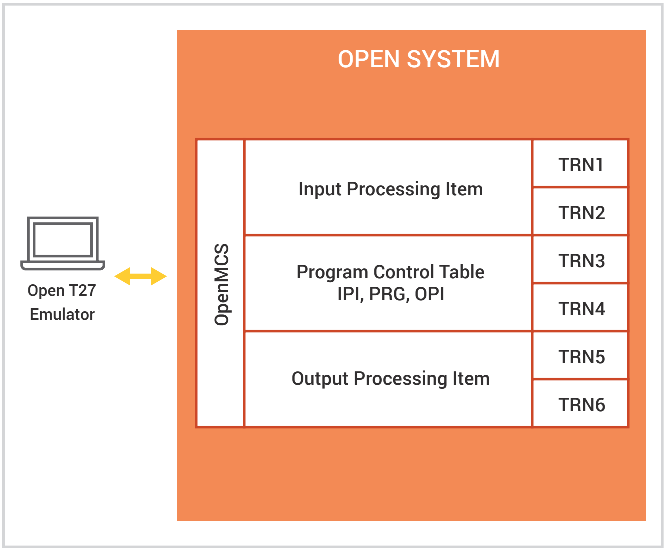

Figure 4: Input and output processing items in OpenMCS

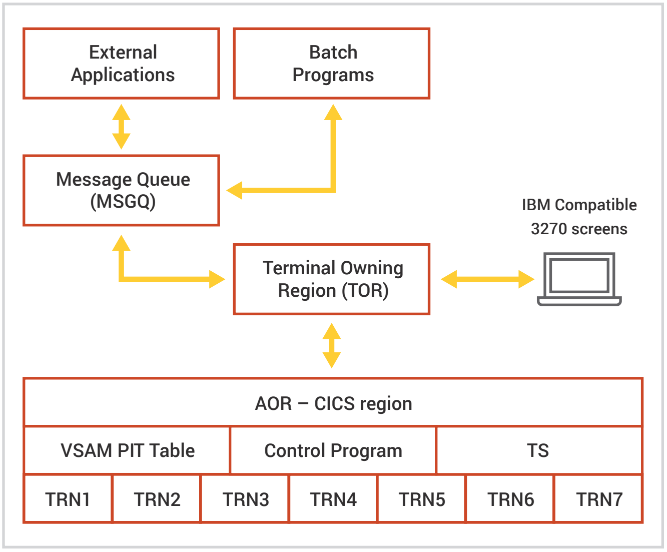

Figure 5: Z-COMS regions

On open systems, Astadia's OpenMCS simulates processing items with default input or output programs declared for each window (Figure 4). OpenMCS first routes each input message to the default input program. This program can either forward the message to the destination program or take another action such as returning an error message to the terminal. Likewise, a default output program forwards output messages, errors, or other actions from the destination program to the originating terminal.

Communications messaging system

Unisys mainframes handle program-to-program and program-to terminal messaging through the Communications Management System (COMS®). COMS provides trancode routing and control, system recovery, and database synchronization. The business logic of Unisys online applications often depends on COMS-specific features that are integrated with calls to COMS service functions. The result is that Unisys applications expect certain responses from COMS to function properly. Applications interfacing with terminals also expect stations to send and receive T27-style messages and to respond to T27 control codes.

IBM mainframes use the Customer Information Control System (CICS®) for message handling and application management. The challenge with CICS is its rigid processing item facility, its limitation of four characters per trancode, and its one-in-one-out structure. Normal practice for CICS online programs is to use Basic Mapping Support (BMS) maps for the user interface; therefore, Astadia enhanced its toolset to convert T-27 screens to BMS maps.

During Unisys-to-IBM migrations, Astadia uses Z-COMS to manage trancodes with more than four characters. Z-COMS is a tool that facilitates the migration of Unisys applications to IBM without the need to rewrite segments of code that communicate with the datacom.

This add-on architecture enables CICS regions to process variable-length trancodes (Figure 5). Routing information is stored in the Program Information Table (PIT), which resides in a Virtual Storage Access Method (VSAM) file. Z-COMS requires a single executable to route transactions with optional input processing items (IPI) and output processing items (OPI).

Z-COMS processes each element in sequence. Return of control to Z-COMS depends on the execution of the PIT table entry. All trancodes (TRN1 through TRN7) are customer configurable. Temporary storage (TS) consists of memory-resident temporary storage tables. An optional terminal owning region (TOR) controls the routing of messages into the application owning region (AOR).

For migrations to open systems, Astadia uses its message control system, OpenMCS, and OpenT27 Emulator (Figure 4). OpenMCS provides trancode routing, which is unavailable in open-systems architectures apart from a third-party solution. The OpenT27 emulator provides the expected T27-style messages for open systems, but industry-standard T27 emulators are supported as well. See Replacing COMS with OpenMCS, later in this document, for a discussion of issues typically encountered in migrating to an open-systems messaging scheme.

Database conversions

The Data Management System (DMSII®) is a proprietary, hierarchical database with a rich set of tools for backup and recovery. For this reason, the migration and preservation of legacy data is not a simple matter of moving the database structure or the data to a new standard. The data requires validation, conversion, and possible modifications targeted to the destination architecture.

In Figure 6, DBCHECK validates the syntax of the Data and Structure Definition Language (DASDL) from DMSII and saves it as an object file with a .DB extension. The Database Convert (DBCVT) program then converts the validated DASDL to an SQL Data Definition Language (DDL) schema. This DDL requires further analysis and possible modification before Astadia can use it to generate the final schema. For IBM migrations, the final destination is a DB2® database. For open-systems migrations, the final destination is a relational database management system (RDBMS) such as Microsoft SQL Server, Oracle, IBM DB2 LUW. In IBM and open-systems architectures, the resultant schema takes advantage of the unique features and performance of the target RDBMS. See Data and database conversions, later in this document, for a discussion of the issues and solutions that typically accompany DMSII-to-RDBMS conversions.

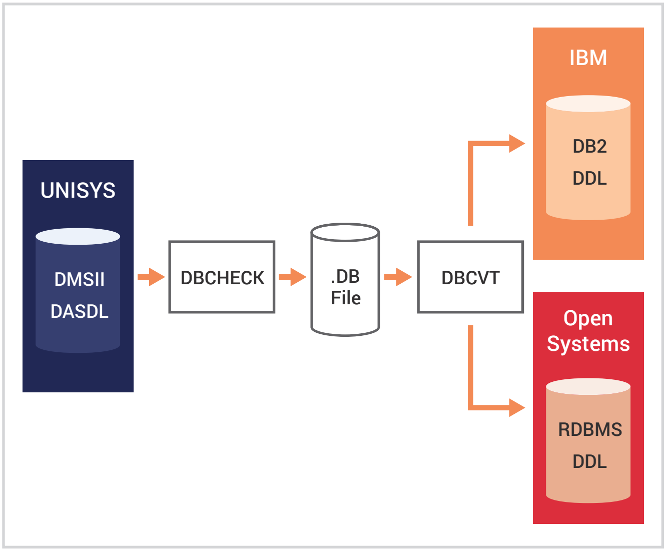

Figure 6: DASDL conversion process

Issues and Solutions

Over the years, Astadia has encountered and addressed the full range of issues that challenge modernization and migration projects, developing as a result, a full set of standardized tools and solutions for success.

Data and database conversions

One of the first challenges in any migration project is the database conversion. For example, items in DASDL are frequently incompatible with DDL. These incompatibilities often require database changes, application changes, or both. Though not exhaustive, the following list represents some of the issues that accompany database conversions:

- Groups
- Occurring items
- Logical databases and remaps
- Embedded datasets
- Pointers
- End transactions
- Unique keys

Groups

In DMSII, DASDL supports the definition of groups that refer to two or more contiguous database items. Consequently, programs can reference individual items or group items, and use groups as key items when retrieving records. Apart from a third-party solution, relational databases do not allow group items.

METHOD	ADVANTAGES	DISADVANTAGES
Single-column	One column is accessed a little faster, and requires less SQL to get to the occurring items.	You cannot read the individual items with tools. This method can also waste storage space.
Individual-column	Individual columns can be accessed with external tools such as report writers and SQL queries.	You have to use more SQL to access all of the occurring items. You can exceed the number of columns the database allows for a table. This method can also waste storage space.
Separate-table	A separate table is closer to a normalized database model, and it does not require access to unused occurring entries. In addition, this method uses storage space more efficiently.	This method requires changes in the program logic and extends the migration timeframe.

Table 1: Method for handling occurring items

Astadia has identified two solutions for overcoming the group restriction. The first solution is to change group name references into individual field references. If the group is a key item, the resulting index would reference the individual items.

To reduce the number of program changes, Astadia defines the group name in a working-storage buffer. The assignment of individual fields to the group allows the program to reference the group name even though it does not exist in the new database schema. Data moved to the group fills in elementary items that correspond to actual fields.

The second solution is to use triggers that simulate group keys. A trigger is a procedure declared within the relational database that automatically executes when a specified database event occurs. In this approach, the RDBMS table requires a new column, large enough to hold all of the concatenated group members. Every time the record changes, the trigger updates the new column. The best solution depends on application requirements, and Astadia's consultants can provide assistance in making the right selection.

Occurring Items

DMSII supports occurring items that programs can access with a subscript. Most relational databases do not support occurring items. During the design of the DDL, however, the database administrator can combine the occurring items into one large column or split them into individual columns. Since the DMSII structures are replicated in working storage, the logic of the program will not need to change with either choice. A third option would be to split each occurring entry into a separate table, but this method requires a change in program logic. Table 1 lists the advantages and disadvantages of these methods.

Logical databases and remaps

Relational databases do not support logical databases or remaps. The database administrator selects either the physical dataset names or the remaps to identify the relational tables, and the programmer modifies programs to match them. Relational databases contain synonyms and views that can simulate portions of logical database functionality and remaps as required. Synonyms allow the use of alternate names to reference the tables. Database administrators can define synonyms and views that create virtual tables, which include only a few of the columns from the actual table.

Embedded datasets

A DMSII dataset may contain another embedded dataset. Each record within the embedded dataset is linked by a many-to-one relationship to a record within the master dataset. To read or update records from the embedded dataset, the program must first read the corresponding record in the master dataset; however, RDBMS tables do not support embedded datasets.

The solution for this problem is to separate the embedded tables, rename them, and link them to the master table through a foreign key. The database administrator changes the database description so that each embedded structure forms a new, distinct RDBMS table. Foreign keys require the addition of new columns that link records in the new tables to records in the master table. The addition of foreign keys establishes a many-to-one relationship between the new tables and the master table.

Pointers versus cursors

DMSII uses record and index pointers to access database records. Pointers enable a program to use the DMSII FIND statement for navigation through the database in a specified direction. Relational databases use cursors to specify a collection of database rows matching the selection criteria. When the original program executes a FIND FIRST followed by a FIND NEXT, the converted program must use a cursor to maintain a pointer to the next record. If the set allows duplicates or specifies a partial key, the program must declare the cursor, open it, and fetch the records in sequence.

Most relational databases do not allow the specification of direction on the FETCH command. The fetching of records occurs in only one direction: from first to last. Reversing the direction of each key simulates FIND LAST followed by FIND PRIOR. For example, ascending becomes descending when building the cursor.

Another potential issue is the size of the cursor, which can grow so large that it includes every record in the table. Some relational databases have an option to limit the number of rows returned by a query. Judicious use of this option can dramatically improve performance.

End transaction statements

Mainframe batch programs can periodically issue an ENDTRANSACTION statement and then continue to read and update records. The relational database equivalent of ENDTRANSACTION is the COMMIT statement. Issuing a COMMIT usually closes all open cursors, and causes them to lose the position of the last record read.

One solution is to perform a single COMMIT at the end of the program. This solution keeps the cursors from closing prematurely, but it can result in a large transaction that locks many records for an unusually long time.

Another solution is the use of hold cursors. For DB2, a WITH HOLD clause in the cursor declaration prevents the COMMIT from closing the cursor; however, the associated cursors require some overhead. Cursors that specify FOR UPDATE OF, or are referenced by a WHERE CURRENT OF clause, cannot be held open across commits. Microsoft SQL server provides configuration settings that allow you to define how the database engine handles cursors across COMMITs, while Oracle maintains cursors across COMMITS by default unless running in Oracle ANSI mode.

Unique keys

A DMSII dataset is not required to declare a set that does not allow duplicates. RDBMS tables should usually contain a column or a combination of columns that does not allow duplicates. Migrated programs have a variety of uses for these unique keys. For example, unique keys can identify records for updates or deletions.

In some cases, only a few records require an update. If the program uses a cursor, it can execute the update by using the unique keys of the record last read by the cursor. This solution eliminates the need to use the WHERE CURRENT OF clause on the UPDATE or DELETE statement.

Programs can also use saved unique keys to re-read the current record, corresponding to a FIND without any specified direction. Cursors that must be closed and reopened require unique keys to place the cursor at the previous position. This procedure might be necessary if a cursor were closed by a COMMIT statement or if a batch program were interrupted.

Some relational databases offer a ROWID or its equivalent. This automatically generated value uniquely identifies each record. Using the ROWID in the WHERE clause is an alternative to using the unique keys.

Replacing COMS with OpenMCS

The migration of online applications from the Unisys mainframe to open systems presents a new set of problems beyond those of batch programs. In addition to database conversions and language transformation, the migration plan must address the need for transaction processing capabilities and screen handling.

Unisys online applications

Online applications running under Unisys MCP use the COMS transaction processing environment. COMS is highly integrated with the MCP operating system. It is stable and capable of handling high loads while maintaining application performance.

The business logic of online applications is often dependent upon COMS-specific features and may be intertwined with calls to COMS service functions. One transaction initiated from a terminal may cause several messages to be routed to different programs. Multiple processing items written in ALGOL may be invoked to handle security, statistics, or logging. Programs expect certain responses from COMS to function properly.

Screens may be coded within the programs or stored in a proprietary format, accessed by an application programming interface (API), and maintained by tools that are available only for MCP. Applications expect stations to send and receive T27 style messages and to respond to T27 control codes.

Open-systems message control

In most multi-user distributed computing environments, such as Linux or UNIX, each terminal requires a unique copy of each program it runs. Furthermore, each copy requires its own memory allocation. The program may also require exclusive use of files, and prevent other users from simultaneously accessing those files.

To move from one program to another, you end the first program and execute the second program. Most UNIX-type systems also use character-at-a-time "dumb" terminals, which require operating system intervention for each character as it arrives or leaves. The following factors can create a significant burden for the operating system:

- Character-at-a-time interrupt servicing
- Single character packet assembly
- Disassembly over a LAN
- Task switching
- Memory swapping to disk
- Managing of dozens of individual users

Issues in the Microsoft Windows environment are similar; however, Windows allows you to start multiple programs and leave them in local memory while you switch tasks. In this case, all character handling is a local function. Nevertheless, a few factors can complicate the management of an environment with multiuser file access and synchronization. For example, the legacy application might be complex and consist of many programs. A large number of users may also share the rehosted applications. The solution for this problem is a Message Control System (MCS) that permits many users to be more productive with minimum use of resources.

MCS requirements

Since Linux, Unix, and Windows do not offer built-in transaction processing support, they require a third-party MCS with the following attributes:

- Flexible configuration
- Transaction routing capabilities
- Demand-based program initiation and termination
- Security features
- Easy interface to migrated applications
- Performance equal to or better than mainframe

Introduction to OpenMCS

OpenMCS, Astadia's message control system for Linux, Unix, and Windows, provides the transaction processing features of COMS necessary to support migrated applications. Figure 7 shows the components of OpenMCS architecture.

The purpose of the OpenMCS Administration utility is to configure OpenMCS. This utility stores its configuration data in a database and exports a single runtime configuration file. OpenMCS reads the configuration file at startup, starts the required number of copies for each program, starts its terminal listener, and waits for incoming connections.

OpenMCS also supports Astadia's OpenT27 Emulator and any UNISYS-TD830 terminal emulator that can connect to the OpenMCS listener through Telnet. A user enters a transaction at one of these terminals and presses the transmit key. OpenMCS then uses the previously configured trancode table to route the transaction to the I/O queue of the appropriate program. The first available copy of the

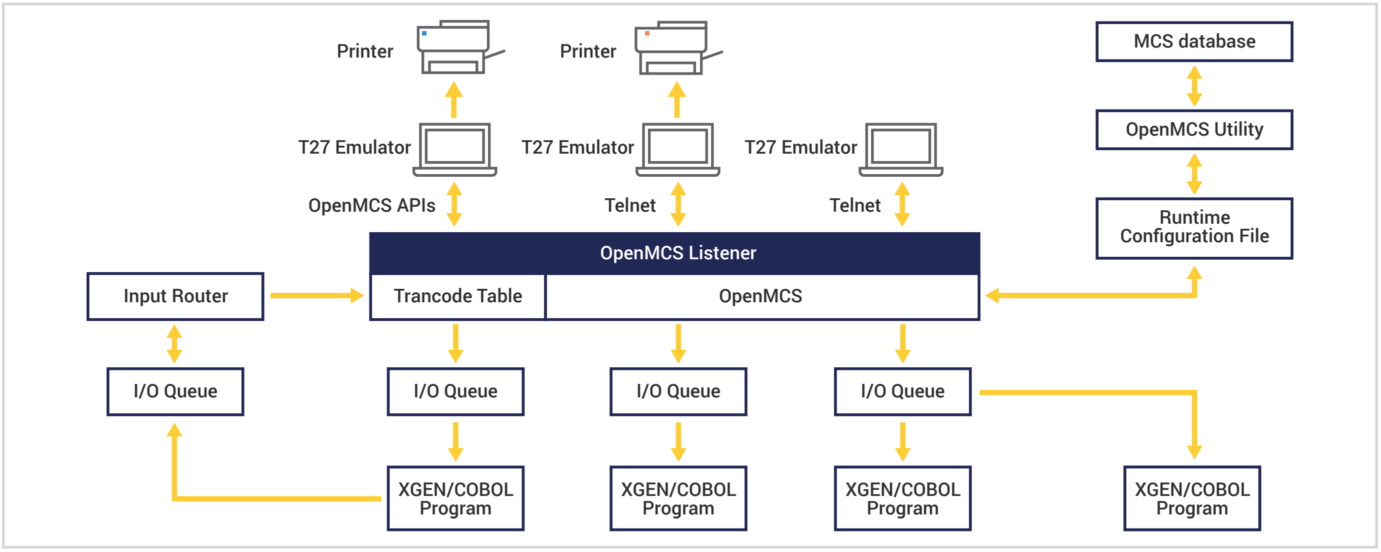

Figure 7: OpenMCS system diagram

program removes the message from the queue, processes it, and sends a response back to the terminal. The input router processes messages that are programmatically sent to it as if they were transactions entered at a terminal.

OpenMCS migration process

The first step of the OpenMCS migration is to run the COMS configuration file through a utility that creates a comparable OpenMCS configuration file. The conversion utility quickly duplicates window, program, transaction, and user declarations in the new environment. The OpenMCS Administration Utility also simplifies adjustment parameters such as the number of program copies and timeouts.

The second step is to run the legacy applications through Astadia's Rules-Based Transformation Engine. This tool converts Unisys COBOL to Micro Focus COBOL and makes the COMS interface compatible with OpenMCS. Programs written in Astadia's XGEN language require only a small change to generate COBOL and OpenMCS calls for the target platform. An OpenMCS API supports programs written in other languages.

OpenMCS interfaces to T27-style terminal emulators. For this reason, screens coded within COBOL programs do not require transformation. Astadia converts screens stored and accessed through proprietary tools to XGEN screen files. Astadia's drop-in libraries enable programs to use the new screen format without inserting lengthy portions of additional code.

Open MCS security schemes

Astadia consultants use the following methods to achieve the desired security scheme in OpenMCS:

- Carry the security information from transaction to transaction in the program's conversation area
- Use a default input program similar to a mainframe processing item to check security for each transaction
- Place each trancode into a security group that grants the use of the transaction only to users associated with the group
- Create a security window that requires user validation through a custom security program that grants access to other windows

Interoperability

Some projects call for an incremental migration approach that allows the mainframe to process selected transactions while the target platform processes others. During this transition, the OpenMCS router and bridge ensure that migrated applications can interact with applications remaining on the mainframe. OpenMCS sends transactions that require mainframe processing to the router, which forwards them through the bridge to Unisys COMS.

Mainframe responses then return through this path to the original terminal.

Trend analysis

Astadia consultants, clients, and partners can use transaction statistics accrued by OpenMCS to aid in runtime analyses. Statistics can be gathered automatically at fixed time intervals to show peak usage throughout the day. These statistics are a valuable tool for diagnosing bottlenecks and adjusting the number of concurrent copies of each program for testing and ongoing maintenance.

The smart choice

Astadia specifically designed OpenMCS as an open-systems replacement for Unisys COMS. This tool also provides benefits such as multiple levels of security, configurable parameters, T27-style screen handling, and load-balancing capabilities. For all these reasons, OpenMCS provides the features and reliability required for the successful migration of online applications. The combination of Astadia's Unisys migration experience and its powerful migration tools make OpenMCS a smart choice for any migration project.

Astadia Roadmap

Astadia approaches its projects with a systematic roadmap that ensures the implementation of the industry's best practices. Each project consists of four phases: Discovery, Design, Transformation, and Deployment (Figure 8). Each phase consists of several related activities that build on previous knowledge and prepare the way for subsequent phases. While no two migrations present the same challenges, this roadmap provides a valuable benchmark for the progress of the project.

Discovery

The Discovery phase sets the stage for successful delivery. During this phase, Astadia collaborates with the client to identify project goals, select the appropriate personnel, create the appropriate plans, and establish requirements. These activities take the form of scope setting, preparation, orientation, and requirements definition.

Scope

Setting scope establishes solution boundaries and prioritizes business functions or processes to be included as project activities and deliverables. Astadia begins the project by forming a partnership with the client to identify business objectives, business cycles, and technical constraints.

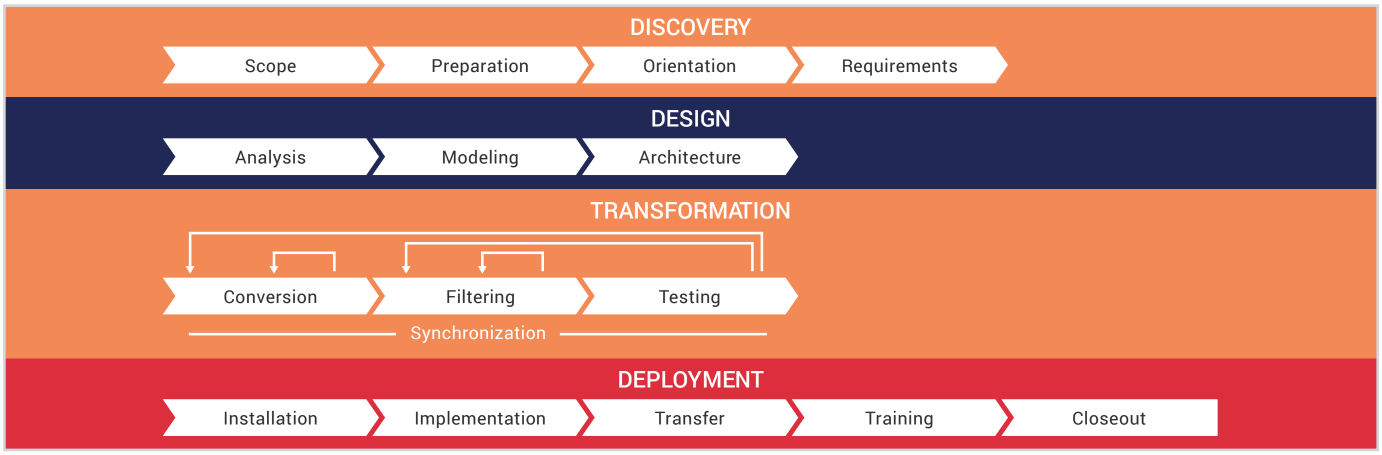

Figure 8: Phases and activities in Astadia's RoadMap

Preparation

Once the key objectives and boundaries are known, Astadia identifies and organizes team members into an informed, collaborative network of professionals. Preparation activities include educating team members about the client and the project, selecting the most effective requirement-gathering strategy, and setting an agenda for the project kickoff meeting.

Orientation

The project kickoff meeting is the primary orientation activity, during which Astadia and client team members discuss project approach, timelines, and roles and responsibilities. Other potential activities include the identification of additional team members and the development of the following artifacts:

- Project charter
- Change management plan
- Risk management plan
- Communication plan
- Project plan
- Detailed timeline
- Other project-specific management plans

Requirements definition

Astadia collaborates with the client to add more detail to the design, construction, testing, and delivery requirements. Detailed definitions typically result in the creation of detailed business, functional, and technical requirements. Other documentation might include usability plans, process flows, and data flows.

Design

During the Design phase, the Astadia project team analyzes the requirements gathered in the Discovery phase and designs the solution. Depending on the project goals and recommended solution, the team may gather or create detailed designs of the client's system architecture, application programs, or other legacy components. As with all deliverables, the team then reviews these designs with the client for quality and completeness, and incorporates any changes made to the requirements during the Discovery phase.

Analysis

Analysis activities lead to the correct migration and modernization strategy for the targeted platform. For this reason, Astadia uses a powerful combination of automated tools to discover and document the client's applications. Team members then cycle through additional requirements gathering, analysis, and design until all analysis questions have been answered in detail. The result of the refined analysis is the creation of a statement of work.

Modeling

Models provide a graphical representation of the required business processes, data schemas, and target platforms. Deliverables might include a proposed workflow, a security model, or other design documents. As the team reviews the solution models, it may identify requirement gaps and initiate further analysis and mitigation activities. Astadia conducts a proof-of-concept conversion to test the model, to prove the viability of the design, and discover areas that require improvement.

Architecture

Architecture transforms the functional and technical requirements into an architectural design that supports the migration project. The design includes details such as transaction load, batch requirements, integration with external systems, and future requirements.

Transformation

The purpose of the Transformation phase is to build and test the solution that evolved during the Design phase. Transformation is an iterative process that includes the conversion of databases and code into the target format and migrating them to the new platform. Astadia normally conducts this work in coordination with the client's business and technical staff to ensure consensus on the project deliverables. The focus of the Transformation phase is the production of the desired solution within the time, cost, and system parameters defined in the Discovery phase.

Conversion

The combination of automation, experience, and collaboration also leads to the early identification of issues. During the data conversion process, the team identifies potential issues, implements modifications, and uses Astadia's DMSII Database Conversion Filter to extract, validate, and convert databases and data from DMSII to the new schema. Astadia repairs any issues identified during this activity and runs the conversion process again. This iterative process continues until the team eliminates every issue. For a list of typical database conversion issues, see Data and database conversions earlier in this document.

Filtering

Filtering is a hybrid process that applies the automated conversion of proprietary code with human intervention. The tool behind the automation is Astadia's Rules-Based Transformation Engine. This tool preserves the business logic and rules of the client's legacy applications while transforming proprietary code. Its smart filters ensure the preservation of mission-critical applications and back-end components such as trancodes, security policies, and message routing.

Though the Rules-Based Transformation Engine is a proven technology, Astadia backs it up with experience and collaboration. Astadia's consultants are professionals with many years of experience in Unisys migrations to other mainframe and open-systems platforms. These consultants are also experienced in collaborating with clients to ensure the fulfillment of their expectations. The combination of automation and human intervention ensures that legacy applications will work in the target environment without sacrificing their original functionality.

Testing

Testing ensures that the solution meets the client's business, functional, and technical requirements. Team members run unit tests throughout the Transformation phase, and support the client's testing at key milestones. Astadia also provides an opportunity for the client to run system tests against business requirements. Applications that do not meet requirements return to the appropriate transformation activity until they perform as expected.

Astadia also provides testing documents, which may include a high-level test strategy, a detailed test plan, test cases, or other documents. The purpose of these test documents is to select the appropriate tests, organize testing efforts, consistently record results, and provide engineers with the data they need for any necessary corrections. At the conclusion of its testing activities, Astadia delivers the final results to the client.

Synchronization

Astadia can also provide an agreement for synchronizing the source code of the migrated applications throughout the project. As the project comes to a close, Astadia takes a final snapshot of the system and synchronizes the migration one more time.

Deployment

The Deployment phase begins with the final installation of the solution and ends with the transition of ownership to the client. At this stage, Astadia and the client have agreed on the acceptance criteria for the solution delivery. Therefore, this phase mainly addresses the mechanics of deploying the solution and ensuring that the client is prepared to take ownership.

Installation

The project team develops the solution at Astadia's lab, but installation must take place in a production environment on the client's site or chosen facility. This activity generally includes the client team members, thus incorporating some knowledge transfer to better enable the client to support the solution.

Implementation

Though rehosted applications are often transparent to end users, Astadia remains available to assist support personnel with the implementation of the solution. Implementation may include the following activities:

- Preparation and delivery of installation guides and scripts
- Support for deployment of the solution into the production environment
- Development and delivery of back-up and recovery processes

Transfer

Transfer activities ensure that client staff understand the technical architecture and design of the migrated system. The client cannot assume ownership of the migrated platform until it gains the expertise to support daily processes and ongoing maintenance. Astadia provides hands-on knowledge transfer and documentation to facilitate the client's needs. Depending on the client's availability, knowledge transfer may occur during the project engagement in the form of mentoring, working with project team members, or both. Transfer can also occur through formal knowledge transfer sessions at the end of the project.

Training

Astadia's team members provide formal or informal training for end users and administrators of its solutions. Astadia will typically recommend a train-the-trainer approach to enable the client to conduct effective training for the initial deployment and ongoing training as needed. Additionally, based on client need, Astadia may provide full training solutions ranging from complete classroom workbooks to online help and how-to guides, producing the most effective user training for the client audience.

Closeout

Towards the end of the project, Astadia works with the client to complete the project deliverables and resolve any outstanding issues. The client signs off on the final deliverables and, therefore, closes out the project. The client's response feeds back into Astadia's process and practice methods for future engagements.

Economy Through Automation

One way to keep project costs low is to use the right tools for the job. Astadia's strategy has been to use the best existing technology where possible and to develop tools for industry gaps. The result of this effort is a powerful suite of automated tools that can eliminate the errors caused by manual changes, dramatically cut project costs, and point the way to the best modernization strategies. Through its continued partnerships and engineering, Astadia has advanced the field of mainframe modernization through the following approaches and techniques:

- Application portfolio management
- Data and database conversion
- Code transformation
- Message handling
- Terminal emulation
- Interface modernization

By using Astadia's services, organizations gain the benefit of its tools without the direct cost of a purchase or the long-term costs of development.

Application portfolio management

Before moving forward with any migration plan, organizations must understand the true cost and value of their legacy applications. Adding to the challenge is the need for a comprehensive analysis that is quick, reliable, and well documented. This kind of upfront analysis is called application portfolio management (APM), and its purpose is to facilitate an agile response to business changes. To incorporate APM into its modernization solutions, Astadia formed a partnership with Micro Focus. Through this partnership, Astadia has gained the powerful technological advantage of Micro Focus' Enterprise Analyzer. This exciting tool combines analytical data with qualitative data to generate graphical and tabular representations of IT portfolio health. These representations enable in-depth analysis of application portfolio inventory relations and interfaces. With this tool, Astadia can discover, evaluate, and document applications, their business value, and their interdependencies across the entire organization. Astadia and the client team evaluate this valuable system intelligence to design a solution that meets the organization's goals.

Data and database conversion

Unisys mainframe environments do not support relational databases such as SQL Server, Oracle, or DB2 LUW. One of the primary reasons to move to relational database technology is to give users better access to their data. The DMSII Database Conversion Filter recreates the hierarchical DMSII database as an equivalent relational schema. Astadia transport facilities convert the application data to this new format and create the production database. Additionally, Astadia can apply years of database tuning knowledge and expertise to maximize the RDBMS performance.

Code transformation

Unisys mainframes use proprietary forms of COBOL that require transformation to a compatible format during the migration of legacy applications to the new platform. The problem is that the modernization of Unisys mainframe applications can involve millions of lines of code. To ensure a consistent transformation process, Astadia developed several tools including the Rules-Based Transformation Engine.

Rules-Based Transformation Engine

For organizations that want to migrate from Unisys platforms while retaining their COBOL applications, Astadia has developed a Rules-Based Transformation Engine. This tool automatically converts proprietary Unisys COBOL with its extensions to open-systems equivalents and replaces the hierarchical database interface with standard embedded SQL. By coupling the transformation engine with OpenMCS, Astadia can redeploy COBOL applications on a variety of commodity hardware platforms.

Code generation

XGEN is Astadia's multiplatform tool that offers Windows, LINUX/UNIX, and mainframe users a powerful, open-systems development environment. Astadia developed XGEN with the benefit of the latest technologies and years of user experience. As a result, XGEN is a database and datacom-independent tool for the generation of COBOL for the target platform. While not required for Unisys mainframe migrations, XGEN is used in many Unisys shops to improve productivity and quality.

Programmers familiar with COBOL enjoy a rapid learning curve with XGEN, because its language is based on COBOL. The language constructs are so intuitive, however, that non-XGEN programmers can read and understand XGEN specifications and programs. In addition, programmers can immediately start to write XGEN applications that can coexist with the organization's legacy applications. This compatibility eliminates the need to rewrite entire legacy applications for the implementation of an XGEN development strategy.

Message handling

Unisys mainframes use a variety of program-to-program messaging systems such as the Communication Messaging System (COMS). Calls to these messaging systems are woven into all legacy software applications developed for Unisys mainframes. This is a primary reason Unisys mainframe applications are not portable to other platforms.

Open-systems message control

Astadia developed its own message control system (OpenMCS) to emulate these messaging systems and facilitate migration of Unisys applications. In addition, the Rules-Based Transformation Engine transforms this code to communicate with OpenMCS. These tools simplify the conversion process and allow Astadia to complete conversions in a fraction of the time it might take others.

Customer information control system

COMS uses variable-length trancodes to route messages for preprocessed, executable, and output items. In IBM mainframes, the Customer Information Control System (CICS) provides a similar function, but imposes a four-character limit. To overcome this limit, Astadia developed Z-COMS. This solution facilitates the migration of Unisys applications without the need to rewrite code segments that communicate with Unisys MCS. Z-COMS also simplifies the conversion process and facilitates the rapid completion of the conversion process to the IBM mainframe.

Terminal emulation

The OpenT27 Emulator is Astadia's Unisys T27 terminal emulator that runs on a variety of Windows, Linux, and Unix platforms. This tool provides terminal connections to Unisys MCP mainframes through Telnet protocol. Its terminal emulation window acts like a standard Unisys ET1100 or T27 forms-mode terminal. The OpenT27 Emulator can also communicate with local LINUX/UNIX applications, Microsoft Windows applications, or applications running under the OpenMCS environment. Finally, the OpenT27 Emulator supports a robust capture/playback/compare facility to aid migration efforts.

Interface modernization

Astadia's ClientView Builder offers the fastest method for modernizing outdated terminal screens or implementing reliable Web-enabled applications. Organizations seldom have a compelling reason to redevelop or replatform legacy applications, but the desire to interoperate with modern technologies and modernize user interfaces often moves them in that direction. Instead of replacing mission-critical systems that took years to develop, Astadia can modernize them in weeks or months.

Services

Astadia's mission is to provide resources and services, which allow organizations to modernize their legacy applications while preserving their mission-critical business rules and processing, as described in Table 2.

SERVICE	DESCRIPTION
Assessment	With over 25 years of specialized Unisys migration experience and a powerful set of automated tools, Astadia is uniquely qualified to assess migration projects.
Planning and management	Astadia provides detailed project plans for the rehosting of the Unisys mainframe applications. These plans include a technical design that outlines specific solutions to unique environmental issues. Astadia is uniquely equipped to coordinate Unisys migration projects. Typical services include project resource management, migration, issue identification and management, project status reporting, and project plan maintenance.
Code transformation	Astadia processes and translates Unisys MCP-based COBOL into Micro Focus COBOL through the proven technology of its Rules-Based Transformation Engine. If the client mainframe uses Unisys LINC, Astadia translates it to XGEN language, which builds on the power of Micro Focus.
Quality assurance	Astadia builds quality into every step of its migration solutions. Business, functional, and technical requirements form a foundation that ensures high quality and satisfied clients.
Implementation assistance	Occasionally, organizations already know their business requirements and have developed a migration plan; however, they have never conducted a migration project and would like assistance to avoid the pitfalls. Astadia provides the experience, expertise, and tools required to carry out the organization's plan.
Staff augmentation	Even well-managed organizations may find it hard to stay with the times in the wake of retiring COBOL programmers. Astadia can supplement staff either in expanding the organization's existing capabilities or staffing a complete mainframe replacement.

Table 2: Astadia Services

Case Studies

Michigan Lottery

The Michigan Bureau of State Lottery has generated revenue and provided quality entertainment for the State of Michigan, consistent with the public good, for over thirty years. In fiscal year 2003, the Lottery reported annual sales of over $1.6 billion. These sales resulted in the distribution of over $586 million for the State School Aid Fund. Retailers received annual commissions of over $121 million while Michigan Lottery players collected prizes over $919 million.

Problem

Nevertheless, several issues forced the Lottery to consider an alternative platform for its mission-critical applications. The existing Unisys mainframe provided few choices for the Lottery's hardware and software operations. The mainframe was significantly outdated, its maintenance costs had skyrocketed, and the software licenses and maintenance contracts were about to expire. Preservation of the existing architecture required the cost-prohibitive purchase of a new mainframe that would not have supported other systems. The Lottery required a cost-effective, open solution that was compatible with its GTECH online gaming system.

Solution

After researching the problem and reviewing similar case studies, the Lottery determined that the best solution was to rehost its applications in an open-systems platform. The key criteria for the decision were cost, performance, reliability, uptime, software compatibility, and online gaming compatibility. The Michigan Lottery chose Astadia for the migration because of its extensive experience in rehosting organizations from proprietary environments to open systems.

To understand the scope and complexity of the problem, Astadia analyzed the programs and databases that the Lottery identified for migration. The result was a statement of work that outlined all the responsibilities, deliverables, and costs. Astadia then worked in partnership with the Lottery to create a comprehensive project plan. This plan provided a detailed list of deliverables and timeline for completion. During the project, Astadia provided training and mentoring in the use of its conversion and development tools. Astadia also provided other services such as database conversion, program conversion, testing assistance, and project management expertise. Astadia successfully completed a smooth migration in partnership with the Michigan Lottery. The Lottery staff can now select the best products for the new platform and experience a more seamless and cost-effective operation. By eliminating the purchase of a new mainframe, Astadia and the Lottery saved Michigan's citizens over $1.5 million.

ASCE

The American Society of Civil Engineer's (ASCE) 150th Anniversary Web site states that it was founded 1852 with a mission to advance professional knowledge and improve the practice of civil engineering.[4] Since then, ASCE has provided career-advancement opportunities through education, recognition, and networking. In 2002, the ranks of this international organization included over 123,000 members throughout the United States and over 200 nations.[5]

Problem

ASCE's previous computing environment was based on a Unisys A-12 mainframe with applications written in LINC and COBOL. The proprietary features of this architecture kept operating costs high and complicated the Society's efforts to serve its membership. By 1993, however, three challenges forced a change. First, a newly adopted strategy called for lower maintenance costs and increased service to its membership. Second, the ASCE faced a 2.3 million-dollar upgrade for the ASeries mainframe. Third, the Unisys mainframe would have kept costs high and denied access to the latest software and application development tools.

These challenges led ASCE to seek an open-systems platform. One of the most significant questions ASCE faced after this decision was how to migrate mission-critical applications such as PROMAPS. This large, LINC-based membership and publication system was written for Unisys A-Series and used mainframe conventions such as DMSII and COMS. ASCE knew what it wanted to do, but was not sure how to achieve it without rewriting applications.

Solution

When ASCE turned to Astadia, it found a professional group of consultants that could provide an automated solution for its migration requirements. For example, the solution included tools for converting A-Series LINC and COBOL to open-systems standards such as Micro Focus COBOL or XGEN. The Astadia solution also let ASCE pay for the migration and buy all new hardware at a lower cost than upgrading the mainframe.

During the seven-month project that followed, Astadia formed a partnership with ASCE to deploy two Unix Digital Alpha 8200 servers. The plan that evolved from this partnership included four phases. In phase 1, the team converted the DMSII database to an Oracle design and created a test database. In phase 2, the team ran the LINC code through an automated conversion filter that converted it to XGEN. This change enabled the generation of code that was compatible with the Unix platform and able to interface with the Oracle database and OpenMCS.

In phase 3, the team tested the code to ensure the faithful reproduction of legacy user interface. In phase 4, the team used ClientView Builder to create the PROMAPS block mode screens as graphic user interface (GUI) screens in the new platform. This strategy provided a modern Windows appearance without the need to relearn the applications or the sacrifice of functionality.

Glossary

CICS	Customer Information Control System: (1) A family of application servers and connectors that provides industrial-strength, online transaction management and connectivity for mission-critical applications (IBM). (2) An IBM-licensed program that provides online transaction-processing services and management for business applications
DASDL	Data And Structure Definition Language: a hierarchical database language used in the DMSII schema for Unisys mainframes
DDL	Data Definition Language: a database language used in the RDBMS schema in open-systems platforms
DML	Data Manipulation Language: A collection of commands used to manipulate data stored within a relational database management system (RDBMS)
DMSII	Data Management System: The proprietary database schema used by Unisys mainframes
Group	A single database item that contains two or more contiguous database items
RDBMS	Relational Database Management System: a structure organized by tables of data values with a set of operations performed on the tables where the goal is to store data only once
XFIB	Extended File Information Block: a tool that provides file attribute information not available to the IBM mainframe environment
XGEN	A programming language that generates COBOL for a variety of platforms and databases
XGEN online Administrator	A tool that lets system administrators update, review, and print configuration settings, and security information.
XIDE	XGEN Integrated Development Environment: a tool used to create, maintain, and visually organize XGEN programs and dictionaries
Z-COMS	A tool that facilitates the migration of Unisys applications to IBM mainframes without the need to rewrite segments of code that communicate with Unisys MCS.