Chapter No.1
"Introduction to Legacy Modernization"
In this package, you will find:

A Biography of the authors of the book
A preview chapter from the book, Chapter NO.1 "Introduction to Legacy Modernization"
A synopsis of the book’s content
Information on where to buy this book

About the Authors

Tom Laszewski has over 20 years experience in databases, middleware, software development, management, and building strong technical partnerships. He is currently the Technical Director of the Oracle Modernization Solutions team. He established the initial business and technical relationships with Oracle's modernization Systems Integrators and tools partners (the Oracle Modernization Ecosystem). His main responsibility is successful completion of all modernization projects initiated through the partner ecosystem. Tom works on a daily basis with EDS and HP alliances, technical architectures and account managers to ensure the success of joint modernization projects. He is also responsible for Oracle Modernization customer assessments and workshops, modernization reference architectures and modernization best practices. Before Oracle, Tom held technical and project management positions at Sybase and EDS. He has provided strategic and technical advice to several starts up companies in the Database, Blade, XML, and storage areas. Tom holds a Master of Science in Computer Information Systems from Boston University.

I would like to thank my wife for being my voice of reason and helping me to be a better person. For my two boys (Slade and Logan), who make life worth living. Thanks to Marc Connolly and Bob Mackowiak of Oracle, and Mike Ballantyne of EDS for their technical advice and ideas to make the book better. To my current and best manager I have ever worked for, Lance Knowlton, for always being supportive of my career aspirations. To my previous manager, John Gawkowski, for giving me the initial opportunity to work in the exciting world of legacy modernization. Lastly, to Jay Peretz and Prakash Balebail for giving "my first big break" at Oracle.
As Product Manager of Modernization Solutions at Oracle, Mr. Williamson is part of the team responsible for developing and implementing Oracle's Modernization strategy, cultivating a partner ecosystem for implementing solutions that modernize to Oracle products. Mr. Williamson works with product management within Oracle and its partners to drive integration, innovation and adoption for modernization to open-systems.

Mr. Williamson has over 14 years experience in the software industry and has extensive knowledge in legacy modernization techniques and commercial software development. Prior to joining Oracle, Mr. Williamson was Global Product Management for BluePhoenix Solutions, where he was responsible for providing technical leadership and new product development within the legacy modernization space. Mr. Williamson also worked for Relativity Technologies assisting the modernization efforts of companies around the world. He has also successfully founded and launched a commercial software company, leveraging emerging technologies as well as creating and managing strategic partnerships key to the company's success. In addition his work within the technology sector Mr. Williamson also served in the United State Marine Corps. He lives with his wife Susan and four small children.

Mr. Williamson has a BSc MIS from the Virginia Commonwealth University.

I would like to thank my wife for her love and support in all of my efforts. God could not have given me a better partner! Thanks very much to all of the Oracle folks who helped me out and our partners. Thanks to the best boss I've ever had, Lance Knowlton, for encouraging me to do great things.
Oracle Modernization Solutions

Much has been written about legacy modernization in the past few years. Most of the books, analyst reports and white papers focus on why you should modernize and theorize at a high level regarding the different approaches and possible outcomes. This book intends to get to the heart of the matter very quickly and into the details of two very well known modernization approaches—SOA Integration and Re-architecturing.

The book is agnostic in terms of hardware and operating systems as most of these have proven to be able to handle the reliability, scalability, and performance of a mainframe system.

What This Book Covers

Chapter 1 gives an overview of the legacy modernization space. Here, we look at the different options for modernization and some of the drivers of modernization.

Chapter 2 Here, we will present an overview of modernization through SOA Integration. We will explore three strategies for SOA Enablement and when these options make sense.

Chapter 3 The focus of this chapter is to place SOA in the context of Modernization. SOA Legacy Modernization will bring you legacy IT infrastructure into the world of World Wide Web, Web 2.0, and all the other latest Internet-based IT architectures.

Within days, a legacy system can be accessed via a web browser. Your time-to-market using the Legacy SOA Integration approach is weeks, instead of months or years for some other modernization options.

Chapter 4 This chapter is an SOA Integration hands-on example using web enablement of mainframe COBOL/VSAM. We will use Java Server Pages (JSP), Java Database Connectivity (JDBC), the Oracle Legacy Adapter, Oracle Application Server, Java EE Connector API, and XA transaction processing to show a two-phase commit across an Oracle database and VSAM on the mainframe.

Chapter 5 takes a deep dive into what re-architecture is, how to approach re-architecture and strategies for execution. This will enable you to gain an understanding of this modernization option and prepare you for hands-on examples in the next chapters.

Chapter 6 Re-architecture is a multidimensional set of technologies and business drivers, similar to a Rubik's cube. In this chapter, we will simplify the target architecture variable, by making a move to Oracle. Even as we do this, we will find a wide selection of Oracle products and technologies, which we will narrow down for you. We end the chapter with a mapping of re-architecture business drivers to Oracle products, and finally a bit about modernization re-architecture and IBM.
Chapter 7 There are probably some legacy systems out there without batch, but it is most likely that batch processing is a key component of your mainframe legacy system. Business information processing in today’s world is workload automation centered, event focused, and business-centric. However, we learn in this chapter that there are many reasons—both technical and business—why nightly batch jobs will be with us for some time. New technologies such as BPEL could potentially replace classic mainframe job scheduling software in the distant future.

Chapter 8 The online technical architecture has the potential to be much more complex than the batch open system architecture because the possible combination of products and technologies are endless. The combinations of target products and technologies include everything from cell phone access, to ERP application integration, to business intelligence, and transactions across multiple databases from different vendors. The same can be said of the source legacy system: multitudes of end user devices, languages, transactions processors, and databases. What we will immediately do is simplify the target architecture to three layers: presentation, transaction, business logic processing and database.

Chapter 9 The purpose of the scenario is to reinforce that re-architecture is not "rip and replace", but leveraging existing mainframe artifacts and forward engineering these artifacts into an Oracle/Java EE architecture using Relativity, and Oracle tools and products. The focus in this example will be on Relativity tools because the most complex and important aspects of re-architecture are discovery, recovery, capture of legacy artifacts, and creating a source model. The source application is a classic mainframe COBOL/CICS/VSAM/DB2 order entry system.

Chapter 10 Re-host based modernization approach is focused on migrating the application off the mainframe to a compatible software stack on an open-systems platform, preserving the language and middleware services on which the application has been built. It preserves legacy investment by relying on a mainframe-compatible software stack to minimize changes in the core application, and preserve the application's business logic intact, while running it on an open-system OS using more flexible and less expensive system infrastructure. It keeps open the customer's options for SOA enablement and re-architecture, by using an SOA-ready middleware stack to support Web services and ESB interfaces for re-hosted components.

Chapter 11 will let us examine trends in computing, where the modern data center is going and how this affects modernization choices today. We will see where the industry is trending with respect to tools, software and infrastructure. We'll examine concepts such as cloud computing, model driven modernization and green computing.

Appendix A gives an overview of the technological terms and legacy acronyms used throughout the book.
Introduction to Legacy Modernization

A lot has been written on legacy modernization in the past few years. Most of the books, analyst reports, and white papers discuss at a high level why one should modernize and theorize, and the different approaches to, and possible outcomes of modernization. Instead of going into modernization theory, we will quickly dive into the details of two very well known modernization approaches: SOA Enablement and Re-architect. There will be a specific focus on modernization to Open Systems taking advantage of the Oracle technology stack, which can provide mainframe quality of service while delivering the agility of a modern architecture simultaneously. We will uncover a specific set of tools and show the process from end-to-end.

We will take an agnostic perspective of hardware and operating systems as most of these have proven to be capable of handling the reliability, scalability, and performance of a mainframe system. In fact, at the time of this writing, the current records for transactions per second have been delivered with Oracle on Intel-based servers.

For most organizations, the ideal solution would be to re-architect everything since re-architecting yields the most modernized environment—the environment that makes the best use of modern technology, is the most agile when it comes to change, and relies no longer on legacy skill sets.
Although such a big bang scenario is technically feasible, in reality, it is difficult and risky for any organization to accomplish this in a single re-architecting step—no matter how desirable the outcome. Most organizations would view such a big bang approach as putting their entire organization at risk. As a result, they take several intermediate steps. The following chapters show several options that could be considered in order to break down the modernization problem into byte-sized chunks—all the while delivering the final goal of achieving a process-driven SOA architecture based on J2EE. Additionally, these intermediate steps of SOA enablement will yield measurable ROI and benefit.

What We Won't Cover
Before we begin our path to modernization, let's take some time to talk about the things that we will not cover in this book. The main focus of this book is a practical application of how to modernize a legacy application using two specific techniques. We won't cover topics such as marketplace, methodologies, and estimation techniques.

Methodology and Estimation
Countless books have been written on application development methodology. Every system integrator/programming shop within a large company or technology group has a general development methodology, be it waterfall, agile, or eXtreme programming. The techniques in this book can fit any given protocol.

Estimation is a bit different and varies from system to system and with the choice of the modernization option. It can depend upon factors such as target language, tools, and the level of automation you are employing. If someone tries to sell you a solution based on the line of code or function point counts and complexity, you can pretty much throw that out of the window. Function point analysis is a great tool for understanding the complexity of the source code and can drive estimation, but there is certainly no general formula for how long a modernization will take, or how much it will cost. Another book can be written on this subject.

The Modernization Marketplace and Why Modernize
If you are reading this book, then we will assume that application modernization is a necessity for you. You are looking at "how to modernize" rather than "why modernize". Further, much market research has been done on this subject. Countless presentations, white papers, and events are actively being conducted on this subject.
The largest and best of breed systems integrators of the world have practices built solely around the modernization market. There are several reasons that drive a legacy modernization project. High costs, lack of agility, an aging technology workforce are just some of the reasons for modernization. Sometimes the motivation to modernize is driven from the business, at other times it is pure technology play. The reasons are many, and the final decision to embark on this effort depends on each organization. Again, much material is being developed on this subject and is not the topic of this book.

The **Oracle Modernization Alliance (OMA)** is an effort by Oracle to bring together the best of breed partners and products to enable modernization to open systems. This is truly an emerging field both for companies considering modernization, and for the companies working to provide those technologies. The OMA is a resource to help customers identify the best path to modernization. The following is a table of some key resources that you can have access to from Oracle around modernization. In addition, we will list some key alliances that Oracle has in the modernization space. Here, you will find abundant market research, white papers, and links to key contacts for getting engaged on a modernization initiative.

Oracle Modernization Alliance resources are as follows:

- Oracle Modernization Alliance (http://www.oracle.com/goto/oma)
- Oracle Modernization Blog (http://blogs.oracle.com/jblog)
- Oracle Migration Technology Center (http://www.oracle.com/technology/tech/migration)
- Oracle Migration Knowledgebase (http://www.oracle.com/technology/tech/migration/kb)

Oracle works with many global systems integrators who focus on legacy modernization. The following is a list of the current system integrators that are apart of the Oracle Modernization Alliance.
Deep Dive on Approaches

There are five primary options for modernization, and all are worthy of deep exploration. In the next section, we will review each of these options at a high level. However, this book is a deep technical dive on two approaches for Legacy Modernization, namely SOA enablement and re-architecture. These two options are selected for two reasons. First, it gives a modernization option for staying on the mainframe (SOA enablement) and moving off the mainframe (re-architecture). Second, many organizations around the world are engaged on one of these two paths, or both in many cases. Although either modernization option can be chosen independently, together they provide a smooth and measured path to a modern environment without the risk of a big bang approach. We also cover a rehosting-based approach to modernization, which minimizes the upfront risk and supports SOA enablement and selective re-architecture during or following the automated platform migration. We will cover more of this later.
Overview of the Modernization Options

There are five primary approaches to legacy modernization:

- Re-architecting to a new environment
- SOA integration and enablement
- Replatforming through re-hosting and automated migration
- Replacement with COTS solutions
- Data Modernization

Other organizations may have different nomenclature for what they call each type of modernization, but any of these options can generally fit into one of these five categories. Each of the options can be carried out in concert with the others, or as a standalone effort. They are not mutually exclusive endeavors. Further, in a large modernization project, multiple approaches are often used for parts of the larger modernization initiative. The right mix of approaches is determined by the business needs driving the modernization, organization's risk tolerance and time constraints, the nature of the source environment and legacy applications. Where the applications no longer meet business needs and require significant changes, re-architecture might be the best way forward. On the other hand, for very large applications that mostly meet the business needs, SOA enablement or re-platforming might be lower risk options.

You will notice that the first thing we talk about in this section—the Legacy Understanding phase—isn't listed as one of the modernization options. It is mentioned at this stage because it is a critical step that is done as a precursor to any option your organization chooses.

Legacy Understanding

Once we have identified our business drivers and the first steps in this process, we must understand what we have before we go ahead and modernize it. Legacy environments are very complex and quite often have little or no current documentation. This introduces a concept of analysis and discovery that is valuable for any modernization technique.

Application Portfolio Analysis (APA)

In order to make use of any modernization approach, the first step an organization must take is to carry out an APA of the current applications and their environment. This process has many names. You may hear terms such as Legacy Understanding, Application Re-learn, or Portfolio Understanding. All these activities provide a
clear view of the current state of the computing environment. This process equips the organization with the information that it needs to identify the best areas for modernization. For example, this process can reveal process flows, data flows, how screens interact with transactions and programs, program complexity and maintainability metrics and can even generate pseudocode to re-document candidate business rules. Additionally, the physical repositories that are created as a result of the analysis can be used in the next stages of modernization, be it in SOA enablement, re-architecture, or re-platforming. Efforts are currently underway by the Object Management Group (OMG) to create a standard method to exchange this data between applications. The following screenshot shows the Legacy Portfolio Analysis:

**APA Macroanalysis**

The first form of APA analysis is a very high-level abstract view of the application environment. This level of analytics looks at the application in the context of the overall IT organization. Systems information is collected at a very high level. The key here is to understand which applications exist, how they interact, and what the identified value of the desired function is. With this type of analysis, organizations can manage overall modernization strategies and identify key applications that are good candidates for SOA integration, re-architecture, or re-platforming versus a replacement with Commercial Off-the-Shelf (COTS) applications. Data structures, program code, and technical characteristics are not analyzed here.

The following macro-level process flow diagram was automatically generated from Relativity Technologies Modernization Workbench tool. Using this, the user can automatically get a view of the screen flows within a COBOL application. This is used to help identify candidate areas for modernization, areas of complexity, transfer of knowledge, or legacy system documentation. The key thing about these types of reports is that they are dynamic and automatically generated.
The previous flow diagram illustrates some interesting points about the system that can be understood quickly by the analyst. Remember, this type of diagram is generated automatically, and can provide instant insight into the system with no prior knowledge. For example, we now have some basic information such as:

- **MENSET1.MENMAP1** is the main driver and is most likely a menu program.
- There are four called programs.
- Two programs have database interfaces.

This is a simplistic view, but if you can imagine hundreds of programs in a visual perspective, we can quickly identify clusters of complexity, define potential subsystems, and do much more, all from an automated tool with visual navigation and powerful cross-referencing capabilities. This type of tool can also help to re-document existing legacy assets.

**APA Microanalysis**

The second type of portfolio analysis is APA microanalysis. This examines applications at the program level. This level of analysis can be used to understand things like program logic or candidate business rules for enablement, or business rule transformation. This process will also reveal things such as code complexity, data exchange schemas, and specific interaction within a screen flow. These are all critical when considering SOA integration, re-architecture, or a re-platforming project.
The following are more models generated from the Relativity Modernization Technologies Workbench tool. The first is a COBOL transaction taken from a COBOL process. We are able to take a low-level view of a business rule slice taken from a COBOL program, and understand how this process flows. The particulars of this flow map diagram are not important; rather, this model can be automatically generated and is dynamic based on the current state of the code.

The second model shows how a COBOL program interacts with a screen conversation. In this example, we are able to look at specific paragraphs within a particular program. We can identify specific CICS transaction and understand which paragraphs (or subroutines) are interacting with the database. The models can be used to further refine our drive for a more re-architected system, help us identify business rules and help us populate a rules engine, which we will see in the later chapters.
This example is just another example of a COBOL program that interacts with screens—shown in gray, and the paragraphs that execute CICS transactions—shown in white. So with these color coded boxes, we can quickly identify paragraphs, screens, databases, and CICS transactions.

**Application Portfolio Management (APM)**

APA is only a part of IT approach known as Application Portfolio Management. While APA analysis is critical for any modernization project, APM provides guideposts on how to combine the APA results, business assessment of the applications' strategic value and future needs, and IT infrastructure directions to come up with a long term application portfolio strategy and related technology targets to support it. It is often said that you cannot modernize that which you do not know. With APM, you can effectively manage change within an organization, understand the impact of change, and also manage its compliance.
APM is a constant process, be it part of a modernization project or an organization’s portfolio management and change control strategy. All applications are in a constant state of change. During any modernization, things are always in a state of flux. In a modernization project, legacy code is changed, new development is done (often in parallel), and data schemas are changed. When looking into APM tool offerings, consider products that can provide facilities to capture these kinds of changes in information and provide an active repository, rather than a static view. Ideally, these tools must adhere to emerging technical standards, like those being pioneered by the OMG.

Re-Architecting

Re-architecting is based on the concept that all legacy applications contain invaluable business logic and data relevant to the business, and these assets should be leveraged in the new system, rather than throwing it all out to rebuild from scratch. Since the new modern IT environment elevates a lot of this logic above the code using declarative models supported by BPM tools, ESBs, Business Rules engines, Data integration and access solutions, some of the original technical code can be replaced by these middleware tools to achieve greater agility. The following screenshot shows an example of a system after re-architecture.
The previous example shows what a system would look like, from a higher level, after re-architecture. We see that this isn’t a simple transformation of one code base to another in a one-to-one format. It is also much more than remediation and refactoring of the legacy code to standard Java code. It is a system that fully leverages technologies suited for the required task, for example, leveraging Identity Management for security, business rules for core business, and BPEL for process flow.

Thus, re-architecting focuses on recovering and reassembling the process relevant to business from a legacy application, while eliminating the technology-specific code. Here, we want to capture the value of the business process that is independent of the legacy code base, and move it into a different paradigm. Re-architecting is typically used to handle modernizations that involve changes in architecture, such as the introduction of object orientation and process-driven services.

The advantage that re-architecting has over greenfield development is that re-architecting recognizes that there is information in the application code and surrounding artifacts (example, DDLs, COPYBOOKS, user training manuals) that is useful as a source for the re-architecting process, such as application process interaction, data models, and workflow. Re-architecting will usually go outside the source code of the legacy application to incorporate concepts like workflow and new functionality that were never part of the legacy application. However, it also recognized that this legacy application contains key business rules and processes that need to be harvested and brought forward.
Some of the important considerations for maximizing re-use by extracting business rules from legacy applications as part of a re-architecture project include:

- Eliminate dead code, environmental specifics, resolve mutually exclusive logic.
- Identify key input/output data (parameters, screen input, DB and file records, and so on).
- Keep in mind many rules outside of code (for example, screen flow described in a training manual).
- Populate a data dictionary specific to application/industry context.
- Identify and tag rules based on transaction types and key data, policy parameters, key results (output data).
- Isolate rules into tracking repository.
- Combine automation and human review to track relationships, eliminate redundancies, classify and consolidate, add annotation.

A parallel method of extracting knowledge from legacy applications uses modeling techniques, often based on UML. This method attempts to mine UML artifacts from the application code and related materials, and then create full-fledged models representing the complete application. Key considerations for mining models include:

- Convenient code representation helps to quickly filter out technical details.
- Allow user-selected artifacts to be quickly represented in UML entities.
- Allow user to add relationships and annotate the objects to assemble more complete UML model.
- Use external information if possible to refine use cases (screen flows) and activity diagrams—remember that some actors, flows, and so on may not appear in the code.
- Export to XML-based standard notation to facilitate refinement and forward-re-engineering through UML-based tools.
Modernization with this method leverages the years of investment in the legacy code base, it is much less costly and less risky than starting a new application from ground zero. However, since it does involve change, it does have its risks. As a result, a number of other modernization options have been developed that involve less risk. The next set of modernization option provide a different set of benefits with respect to a fully re-architected SOA environment. The important thing is that these other techniques allow an organization to break the process of reaching the optimal modernization target into a series of phases that lower the overall risk of modernization for an organization.

In the following figure, we can see that re-architecture takes a monolithic legacy system and applies technology and process to deliver a highly adaptable modern architecture.

**SOA Integration**

Since SOA integration is the least invasive approach to legacy application modernization, this technique allows legacy components to be used as part of an SOA infrastructure very quickly and with little risk. Further, it is often the first step in the larger modernization process. In this method, the source code remains mostly unchanged (we will talk more about that later) and the application is wrapped
using SOA components, thus creating services that can be exposed and registered to an SOA management facility on a new platform, but are implemented via the exiting legacy code. The exposed services can then be re-used and combined with the results of other more invasive modernization techniques such as re-architecting. Using SOA integration, an organization can begin to make use of SOA concepts, including the orchestration of services into business processes, leaving the legacy application intact.

Of course, the appropriate interfaces into the legacy application must exist and the code behind these interfaces must perform useful functions in a manner that can be packaged as services. SOA readiness assessment involves analysis of service granularity, exception handling, transaction integrity and reliability requirements, considerations of response time, message sizes, and scalability, issues of end-to-end messaging security, and requirements for services orchestration and SLA management. Following an assessment, any issues discovered need to be rectified before exposing components as services, and appropriate run-time and lifecycle governance policies created and implemented.

It is important to note that there are three tiers where integration can be done: Data, Screen, and Code. So, each of the tiers, based upon the state and structure of the code, can be extended with this technique. As mentioned before, this is often the first step in modernization.
In this example, we can see that the legacy systems still stay on the legacy platform. Here, we isolate and expose this information as a business service using legacy adapters.

The table below lists important considerations in SOA integration and enablement projects.

<table>
<thead>
<tr>
<th>Criteria for identifying well defined services</th>
<th>Services integration and orchestration</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Represent a core enterprise function re-usable by many client applications</td>
<td>• Wrapping and proxying via middle-tier gate-way vs. mainframe-based services</td>
</tr>
<tr>
<td>• Present a coarse-grained interface</td>
<td>• Who's responsible for input validation?</td>
</tr>
<tr>
<td>• Single interaction vs. multi-screen flows</td>
<td>• Orchestrating &quot;composite&quot; MF services</td>
</tr>
<tr>
<td>• UI, business logic, data access layers</td>
<td>• Supporting bidirectional integration</td>
</tr>
<tr>
<td>• Exception handling—returning results without tranching to another screen</td>
<td></td>
</tr>
</tbody>
</table>

**Discovering "Services" beyond screen flows**

- Conversational vs. sync/async calls
- COMMAREA transactions (re-factored to use reasonable message size)

**Security policies and their enforcement**

- RACF vs. LDAP-based or SSO mechanism
- End-to-end messaging security and Authentication, Authorization, Audition

**Platform Migration**

This area encompasses a few different approaches. They all share a common theme of low risk, predictable migration to an open system platform with a high level of automation to manage this process. With platform migrations, the focus is moving from one technology base to another as fast as possible and with as little change as possible. In Chapter 10, Introduction to Re-hosting Based Modernization using Oracle Tuxedo, we will focus on moving from mainframe platforms to open systems through a combination of re-hosting applications to a compatible environment maintaining the original application language (usually COBOL), and automated migration of applications to a different language when necessary. Each uses a high level of automation and a relative low level of human interaction as compared to other forms of modernization. The best re-platforming tools in the market are...
rules-based, and can also support automated changes to business logic or data access code when required to address specific business needs through specifically configured rule sets.

**Automated Migration**

Automated migration is a technique in which software tools are used to translate one language or database technology to another. It is typically used to protect the investment in business logic and data in cases where the source environment is not readily available or supportable (example skills are rare) on the target platform. Such migrations are only considered automated if the scope of conversion handled by the tools is at least 80 percent. Automated migration is very fast and provides a one-to-one functionally equivalent application. However, the degree of the quality of target code is heavily dependent upon what the source is.

There are two primary factors which determine how good the target application is. The first factor being, what is the source paradigm? If you are coming from a procedure-based programming model such as COBOL, then the resulting Java will not be a well-structured object-oriented code. Many vendors will claim pure OO, or 100 percent compliant Java. But in reality, OO languages programs can still be used in a procedural fashion. When the source is a step-by-step COBOL application, then that is what you will end up with after your migration to Java. This solution works quite well when the paradigm shift is not large. For example, going from PL/I to C/C++ is much more attainable with this strategy than converting COBOL to Java. This strategy is often used to migrate from 4GLs, such as Natural or CA Gen (formerly COOL:Gen) to COBOL or Java. Of the two target environments, migration to Java is more complex and typically requires additional manual re-factoring to produce proper OO POJO components or J2EE EJBs that can be easily maintained in the future.

The second factor one needs to consider is the quality of the source. Some re-factoring can be done on the source language, or the meta-language often generated in the transformation. But these usually only address things such as dead code or GOTO statements, not years of spaghetti code.

If your goal is to quickly move from one technology to another, with functional equivalence, then this is a great solution. If the goal is to make major changes to the architecture and take full advantage of the target language, then this type of method usually does not work.
Re-Hosting

Re-hosting involves moving an application to another hardware platform using a compatible software stack (example COBOL containers and compatible OLTP functionality provided by Oracle Tuxedo) so as to leave the source application untouched. This is most commonly used approach to migrate mainframe COBOL CICS to an open systems platform and has been used in hundreds of projects, some as large as 12,000 MIPS.

The fundamental strength of rehosting is that the code base does not change and thus there are no changes to the core application. There are some adaptations involved for certain interfaces, batch jobs, and non-COBOL artifacts that are not inherently native to the target environment. These are usually handled through automated migration. The beauty of this solution is that the target environment using open systems platform, typically UNIX or Linux, has a significantly lower TCO than the original mainframe environment, allowing customers to save 50 to 80 percent compared to their mainframe operations. The budget savings gained from this move can fund more long term, yet beneficial re-architecture effort.
Re-Hosting Based Modernization

Evolving from the core re-hosting approach and leveraging flexible, rules-driven automated conversion tools, this approach goes beyond re-hosting to a functionally-equivalent application. Instead of a pure shift of COBOL code to a target system without any changes to the original code, some of the automated tooling used by Oracle's migration partners to re-host applications and data also enables automated re-engineering and SOA integration during or following migration. For example, Metaware Refine workbench has been used to:

- Automatically migrate COBOL CICS applications to COBOL Tuxedo applications.
- Convert PL/I applications running under IMS TM to C/C++ applications under Tuxedo.
- Identify and remove code duplication and dead code, re-documenting flows and dependencies derived from actual code analysis.
- Migrate VSAM data and COBOL copybooks describing the data schema to Oracle DB DDLs and automatically change related data access code in the application.
- Migrate DB2 to Oracle DB, making appropriate adjustments for data type differences, changing exception handling based on differences in return codes, and converting stored procedures from DB2 to Oracle.
- Perform data cleansing, field extensions, column merges and other data schema changes automatically synchronized across data and data access code.
- Migrate non-relational data to Oracle DB to provide broader access from applications on distributed systems.
- Convert 3270/BMS interface to Web UI using JSP/HTML, enabling modifications and flow optimization in original legacy UI.
- Adapt batch to transactional environment to shorten batch windows.

APA tools for automated business rule discovery can also be used to help identify well defined business services and use Oracle Tuxedo's SOA framework to expose these COBOL services as first class citizens of an enterprise SOA. This approach can also be applied to PL/I applications automatically migrated to C/C++ and hosted in Tuxedo containers. The bulk of the re-hosted code remains unchanged, but certain key service elements that represent valuable re-use opportunities are exposed as Web Services or ESB business services. This approach protects investment in the business logic of the legacy applications by enabling COBOL components to be extended to SOA using native Web Services gateway, ESB integration, MQ integration, and so on of the Oracle Tuxedo—a modern TP/Application Server platform for COBOL, C, and C++.
Thus, we gain a huge advantage by having a well structured, SOA-enabled architecture on a new platform that was delivered with a high degree of automation. Using a proven application platform with built-in SOA capabilities, including native Web Services support, ESB transport, transparent J2EE integration, and integration with meta-data repository for full services lifecycle governance, makes this a low-risk approach. It also helps to address some of the key considerations in SOA integration table above. With this approach we have the ability to extend and integrate the legacy environment easier than a pure re-host, while benefitting from the automation that ensures high speed of delivery and low risk that is comparable to a black-box re-hosting.

The other aspect of this process is identifying components that will benefit from re-architecture—usually code with low maintainability index or code requiring significant changes to meet new business needs—and using re-architecture techniques to re-cast it as a new components, such as business process, declarative rules in a business engine, or re-coded J2EE components. The key is to ensure that the re-architected components remain transparently integrated with the bulk of the re-hosted code, so that the COBOL or C/C++ code out-side of the selected components doesn't have to be changed. With Oracle Tuxedo this is done via transparent bi-directional support for Web Services (using Oracle SALT) and J2EE integration (using WebLogic-Tuxedo Connector). The key guidelines listed for business rules-extraction and model mining apply to the components selected for re-architecture.

Re-hosting based modernization is sometimes referred to as Re-host++. This term highlights its roots in re-hosting applications to a compatible technology stack together with the broad range of re-engineering, SOA integration, and re-architecting options it enables. This unique methodology is supported by a combination of an extensible COBOL, C, and C++ application platform — Oracle Tuxedo, with flexible, rules-driven automated conversion tools from Oracle's modernization partners.
Data Modernization

Here we look at strategies to modernization—a set of data stores that are stored across disparate and heterogeneous sources. We often have problems with accessing and managing legacy data. There is an increase in cost to run batch jobs, which generate reports 24 to 48 hours after they are needed. Further, this legacy data often needs to be integrated with other database systems that are located on different platforms. So, from a business perspective, there is a real problem in getting actionable data in a reasonable amount of time, and at a low cost.

With Data Modernization solutions, we can look at leaving legacy data on the mainframe, pulling it out in near real time, lowering MIPS costs by processing reports outside of the batch winding, and integrating this with heterogeneous data sources. This is leveraged through employing several technologies in concert.
Legacy Adapters

With a collection of legacy adapters provided by Oracle and our partners, we are able to enable the organization to access almost any data store with any environment. Further, many of these technologies can employ bidirectional change data capture so that we can publish data changes to a data warehouse in near real time.

The following is the current list of legacy adapter and change data capture partners that are a part of the Oracle Modernization Alliance. The following is the list of legacy adapter/data migration partners along with their respective URLs:

- Attachmate (http://www.attachmate.com/)
- Attunity (http://www.attunity.com/)
- DateDirect Technologies (Progress Software) (http://www.datadirect.com/)
- GT Software, Inc (http://www.gtsoftware.com/)
- Hostbridge (http://www.hostbridge.com/)
- Micro Focus (http://www.microfocus.com/)
- OpenConnect Systems (http://www.oc.com/)
- SeaGull Software Systems (http://www.seagullsoftware.com/)
- Treehouse Software (http://treehousesoftware.com/)

With legacy data adapters, we can access relational and nonrelational data stores. So, once we have access to this information, we then need to rationalize that to a common data set. This is where we can employ Oracle's ETL Tools.

Using Oracle Data Integrator (ODI), or Oracle Warehouse Builder, we can then connect to these data stores to do the data mapping and transformation. With ODI, we can integrate with high volume and event-driven processes. Once this has been transformed to the target database, we then employ Oracle Business Intelligence.

With Business Activity Monitoring, we can extend the notion of Business Intelligence and gain a real-time view into the business processes of the organization. Managers can now transform their core business from disparate data, expensive, and stale reports to a vision into the discrete business processing that drives the organization. This enables organizations to leverage their legacy assets to deliver not only key business intelligence, but also correlate these processes to key performance indicators (KPIs).
Finally, another important feature of this solution is that it is noninvasive. Many mainframe shops do not go for the large scale modernization efforts due to the perception of high risk involved. With data modernization, an organization can retain the entire legacy infrastructure without changing it. We would rather employ SOA technologies to extend the mainframe, while simultaneously lowering MIPS costs.

From a visual perspective, we can go from this type of latent and expensive static reporting:

![New Order Data Modernization Diagram]
To this, leveraging Business Activity Monitoring/Business Intelligent in a fully integrated manner.

**Replacement**

Commercial Off-the-Shelf (COTS) products are frequently considered when a modernization project is undertaken. If the target package exists, COTS Replacement can be a highly cost-effective strategy with a significant reduction in risk. This works very well with common applications such as billing, HR, payroll, and other more commonly used applications.

The implementation of a COTS solution assumes that the organization is willing to adapt to the new system paradigm. Therefore, core business processes must be altered and adapted. Another aspect of a COTS Replacement built on a SOA framework is that one can utilize this new architecture for other component orchestration. Oracle Applications are built upon **Fusion Middleware**, which can enable an entire organization to integrate heterogeneous applications and data faster.
Business Value—What Really Matters?

It is very important to take a holistic view on what the benefits are for legacy modernization. There are, of course, cost savings in hard dollars that are associated with a large-scale modernization. After all, TCO/ROI is often what drives most organizations. However, we have seen an increasing amount of proof points around the soft dollar costs. Those are the benefits that have an indirect impact on costs, but a direct impact on the quality of service, agility, and the ability to service the customer. Remember, not all organizations are driven by profits, for example, the Public sector. So, C-level executives do not just look at the hard dollars, but also consider the soft dollar impact to modernization. The following is a detailed list of some of these hard and soft dollar benefits of legacy modernization. These are all real world examples, though some of the company names have been omitted for the purpose of nondisclosure.

These examples are mapped to specific Oracle products and partner's solutions delivered in a modernization effort.
### Soft-Dollar Benefits

<table>
<thead>
<tr>
<th><strong>Soft-Dollar Benefits</strong></th>
<th><strong>Solution Details</strong></th>
<th><strong>Hard Dollar Benefits</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved access to data</td>
<td>Simplified access to relationally based data, open to self service reporting</td>
<td>Using Oracle BI (Discoverer) a United States federal research agency reduced Compliance with National Nuclear Security Agency regulations and management reporting from 5 weeks to less than an hour.</td>
</tr>
<tr>
<td>Ability to enable Business Intelligence</td>
<td>Data harvesting and business analysis enables business agility</td>
<td>A hardware storage vendor increased Sales Productivity 100 percent, sales metrics updated 4 times a day using Oracle BI</td>
</tr>
<tr>
<td>Grid enabled infrastructure</td>
<td>Oracle Database and Middleware Grid technology lowers costs, increases uptime, increases performance and allows for capacity ‘on demand’.</td>
<td>Average results when moving from SMP to Oracle GRID: 1. ROI over 5 years ($1.7M investment) 150 percent 2. Hardware Savings First Year 55 percent 3. Projected HW Savings in 5 years 73 percent 4. Reduced Labor Requirement 3.75 FTE 5. Improved Computing Performance 10 percent</td>
</tr>
<tr>
<td>Reduced time to market for enhancements</td>
<td>Simplified and highly productive development environments enhance developer productivity</td>
<td>1. A Sports Equipment Mfg e implemented a customized B2B portal built and launched in 3 months. 2. U.S. based Insurance saw a 25 percent savings on process changes to the existing system using Oracle BPEL.</td>
</tr>
<tr>
<td>Faster response to regulatory/legislative changes in health insurance</td>
<td>Migration from mainframe to Tuxedo enabled tighter integration with J2EE components at reduced cost and enabled development organization to immediately start extending existing COBOL application with Java components, transparently to existing COBOL programs</td>
<td>Modules that used to take 6-12 month to develop and deploy, are now completed in 2-3 month Customer reduced overall TCO of $65M a year for a 12,000 MIPS mainframe environment down to $10M a year for an open systems infrastructure.</td>
</tr>
<tr>
<td>Open java based solution with vendor choice</td>
<td>Java based solutions foster greater competition among vendors and avoid vendor lock in</td>
<td></td>
</tr>
<tr>
<td>Ability to use third party open tools</td>
<td>Very rich third party toolset eco-system using open systems</td>
<td></td>
</tr>
<tr>
<td>Ability to create robust internet applications</td>
<td>Open access to end users, remote staffing, and partner eco-systems</td>
<td></td>
</tr>
</tbody>
</table>

---

*For More Information:*
## Introduction to Legacy Modernization

<table>
<thead>
<tr>
<th>Soft-Dollar Benefits</th>
<th>Solution Details</th>
<th>Hard Dollar Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved customer perception</td>
<td>Ability to develop dynamic, new generation applications enabled for emerging markets</td>
<td>The study (integrated platform running on a single instance) estimated Thermos' overall investment generating $6.2 million in benefits and an ROI of 222 percent over the life of the project. In addition to capital cost avoidance, savings came from higher staff productivity across several departments, lower inventory carrying costs, and reduced IT management expenses. The investment also yielded significant strategic benefits by lowering Thermos' risk of business disruptions, ensuring compliance with industry standards, and supporting new marketing and other business initiatives.</td>
</tr>
<tr>
<td>Improved time to market for enhancements</td>
<td>Ability to avoid IT backlog</td>
<td>US Navy project using a model-driven service framework provided up to 80 percent code generation, dramatically minimizing coding time. The Jacada Integrator product provided about a 90 percent codeless transactional integration solution for its Windows-based PowerBuilder application and its mainframe based financial information system.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enabled for Business Process Optimization</th>
<th>Ability to optimize systems to capture new markets</th>
<th>CargoSmart on Oracle Fusion Middleware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real time processing vs. batch orientation</td>
<td>Ability to respond immediately to changing business conditions</td>
<td>&quot;Our fast-growing system handles the following:</td>
</tr>
<tr>
<td>Reduced print requirements</td>
<td>Ability to eliminate traditional costs of printing via online reporting</td>
<td>- Adds 215,000 new shipments a month</td>
</tr>
<tr>
<td>Reduced report writing requirements</td>
<td>User reporting eliminates the need for centralization of reporting and associated costs</td>
<td>- Sends 4,000 to 9,000 shipment milestone notification e-mails a day</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Generates 15,000 reports a month (reports are customizable with over 70 possible information fields)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Server generates business critical documents such as bills of lading and invoices for customers&quot;</td>
</tr>
</tbody>
</table>

For More Information:  
### Soft-Dollar Benefits

<table>
<thead>
<tr>
<th>Business rules implementation</th>
<th>Ability to compartmentalize business logic for quicker response, lower maintenance costs and higher quality</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Common execution model</th>
<th>Ability to standardize on one model for database, application server and business rules and process environments</th>
</tr>
</thead>
</table>

### Hard Dollar Benefits

1. The study (integrated platform running on a single instance) estimated Thermos' overall investment generating $6.2 million in benefits and an ROI of 222 percent over the life of the project. In addition to capital cost avoidance, savings came from higher staff productivity across several departments, lower inventory carrying costs, and reduced IT management expenses. The investment also yielded significant strategic benefits by lowering Thermos' risk of business disruptions, ensuring compliance with industry standards, and supporting new marketing and other business initiatives.

2. The study (platform consolidation) estimates NYCHA will realize cumulative benefits of more than $105 million over the life of the B.E.S.T. project, translating into a net present value of $60.2 million. NYCHA will achieve payback on its investment in about 2.5 years, the study projects, and earn an overall return on investment of about 137 percent.

3. Vanderbilt projects that the implementation will yield a savings of 185 percent during the next five years, taking into account the performance and availability of the new systems, and the efficiency gained in managing a single vendor's system.

<table>
<thead>
<tr>
<th>Centralized systems management</th>
<th>Grid management technology reduces systems management staff</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Oracle Database 10g Grid Management:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 46 percent less DBA time than DB2 UDB 8.2</td>
</tr>
<tr>
<td>2. 30 percent less DBA time than Microsoft SQL Server</td>
</tr>
<tr>
<td>3. Saves up to $50,000 per year per DBA</td>
</tr>
</tbody>
</table>

---

For More Information:
## Introduction to Legacy Modernization

<table>
<thead>
<tr>
<th>Soft-Dollar Benefits</th>
<th>Solution Details</th>
<th>Hard Dollar Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboard capabilities</td>
<td>Ability to enable dashboard capabilities for key decision makers</td>
<td>Vanderbilt: &quot;Our legacy systems were unable to transform and organize data into the business intelligence the university needed. For example, Vanderbilt's management could not effectively analyze and manage the costs of services provided by the university's hospital, which accounted for 70 percent of the medical center's expenses. Without the means to make data available through Web browsers, Vanderbilt's users also lacked easy access to vital information.&quot; Vanderbilt projects that the implementation will yield a savings of 185 percent during the next five years</td>
</tr>
<tr>
<td>Low cost development production environments</td>
<td>Jdeveloper represents outstanding development environment</td>
<td>&quot;The built-in framework of Oracle JDeveloper makes our Java developers more productive. We can deliver J2EE applications based on the MVC design pattern in less than a month, compared to the many months it would have taken us without such a framework.&quot;—Josie Lang, E-Service Manager, California Public Employees Retirement System (CalPERS)</td>
</tr>
<tr>
<td>Native global language support</td>
<td>Cost avoidance supporting new markets</td>
<td></td>
</tr>
</tbody>
</table>
| Client training cost is reduced | Easy to use tools make training quickier and less expensive | 1. TietoEnator tracks 1.5B EUR of projects for 15000 employees in 20 countries without end-user training  
2. Honeywell deployed Oracle Analytics for sales-oriented contact center to 800 users in under 90 days  
3. Verizon Wireless deployed Oracle Sales Analytics to 3000+ users in 3 months  
4. Pfizer 4 months from decision to live for 1200 users for Oracle Pharma Analytics  
5. Microsoft live in 100 days, 6500 users using Oracle Sales and Marketing Analytics |
<p>| Client self-service changes, enhancements | Business Analysts, power users able to change rules, do data maintenance, report writing and so on. |
| Stream lined business processes | Reduced number of screens to complete a transaction, less manual workflow |</p>
<table>
<thead>
<tr>
<th>Soft-Dollar Benefits</th>
<th>Solution Details</th>
<th>Hard Dollar Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cybermation Cost Savings (TCO/In Production)</td>
<td>IDC identified an averaged payback period from deploying the Cybermation ESP enterprise job scheduling software of 6.4 months, yielding an average return on investment of 342 percent.</td>
<td>Time savings on defining job schedules after deploying the software averaged 42.7 percent. On average, the companies also spent 33.6 percent less time on monitoring batch job operations. Additionally, there were average timesavings of 14.3 percent in managing and supporting desktops and other clients.</td>
</tr>
<tr>
<td>IT Productivity (38 percent of overall savings here)</td>
<td>To determine the increase in IT staff productivity from deploying the job scheduling software, IDC asked questions about staff time needed for various activities related to IT administrative, operational and support functions, before and after the software's implementation</td>
<td></td>
</tr>
<tr>
<td>User Productivity (20 percent of overall savings here)</td>
<td>IDC asked about the increase in productive time and percentage of users affected after deploying the software. IDC also inquired about the number of downtime incidents and amount of downtime before and after the implementation, as well as the percentage of users affected and their average loaded salary</td>
<td>Based on an average loaded salary for the first year of $32.30 per hour, and annual 5 percent increases, the savings in user productivity averaged close to $1.4 million annually over three years, or $9,023 per 100 users</td>
</tr>
<tr>
<td>CPU Cost Savings (29 percent of overall savings here)</td>
<td>The added CPU benefit came from an average 18 percent reduction in the length of batch windows. Over the three-year period, the CPU savings averaged more than $2 million annually, or $13,352 per 100 users.</td>
<td></td>
</tr>
<tr>
<td>Other Cost Savings (10 percent of overall savings here)</td>
<td>Additional savings came from reductions in IT travel, missed SLAs, and training costs, and from hardware and software savings.</td>
<td>These cost reductions averaged $718,942 a year. Taken together, the savings from improved IT management efficiency and cost reductions yielded an average total savings of $944,487 a year, or $6,196 per 100 users.</td>
</tr>
</tbody>
</table>
### Soft-Dollar Benefits

<table>
<thead>
<tr>
<th>IT Efficiency (3 percent of overall savings here)</th>
<th>Solution Details</th>
<th>Hard Dollar Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>To determine increases in IT efficiency, IDC asked questions about the average number of users and jobs supported by each staff member, before and after deploying the job scheduling software. IDC also asked about IT staff salaries.</td>
<td>For the companies surveyed, the average number of users supported by each FTE rose from 552 before deploying the software to 788 afterwards, an increase of 43 percent. The number of jobs supported by each FTE rose by an average of 204 percent, from 424 to 1,287 per day. With an average first-year loaded salary of $98,127, and annual increases of 5 percent, the payroll savings from increased management efficiency averaged $225,545 a year over the three years.</td>
<td></td>
</tr>
</tbody>
</table>

### Relativity Technologies' Cost Savings References

This information is based upon comparing the cost to doing manual or through processes without Relativity tool set.

### Application Portfolio Management Areas:

- High level inventory statistics 99.80 percent
- Dependencies 99.80 percent
- Boundaries 99.96 percent

99 percent Reduction in cost

### Application Documentation

75 percent Reduction in cost

### Application Analysis Querying

30 percent Reduction in cost

### Business Rule Mining

1. Inventorizing rules
2. High-level \ rule mining
3. Documentation of business rules

74 percent Reduction in cost
Summary

IT organizations are under increasing demand to increase the ability of the business to innovate while controlling and often reducing costs. Legacy modernization is a real opportunity for these goals to be achieved. To attain these goals, the organization needs to take full advantage of emerging advances in platform and software innovations, while leveraging the investment that has been made in the business processes within the legacy environment.

To make good choices for a specific roadmap to modernization, the decision makers should work to have a good understanding of what these modernization options are, and how to get there. We have outlined five methods of modernization: SOA Integration, Re-architecture, Platform Migration, Replacement, and Data Modernization as well as some key factors to be considered for each option. We also explored the imperative that all modernization projects start with a legacy understanding phase.

Finally, this chapter exposed you to the advantages and considerations that need to be made when endeavoring to create a roadmap for modernization. In the following chapters, we will examine two of these methods in detail: SOA enablement and re-architecture, and how you can leverage the Oracle technology stack to achieve these goals. Now that we have built a foundation, let's get into the details.
Where to buy this book
You can buy Oracle Modernization Solutions from the Packt Publishing website: http://www.packtpub.com/oracle-modernization-solutions/book. Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.