

Migrating a Large Scale Legacy IT System to a System with Current Technology

Bill Wood, Mike Gagliardi, Phil Bianco

Copyright 2014 Carnegie Mellon University and IEEE

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM-0001020

Background

Agency wanted to Migrate two paired and tightly coupled systems

- Both are 24/7/365 and have disaster recovery at multiple sites
- Both have many classes of users
- 1. **Service Based System** - 15% functionality (going to 40%)
 - Java, Relational DB, COTS tools, Service-based, J2EE, SAP, Informatica
- 2. **Legacy System** - 85% of functionality (going to 60%)
 - COBOL, Hierarchical DB, mainframe

Migrate to a well defined target reference architecture (TRA) as a basis for a common platform infrastructure (CPI) – developmental, operational, test

- **Briefing is focused on the Legacy Migration**

Its Complicated

Understanding the Legacy System Architecture

- Infrastructure Tools
- Application Component Relationships (Data and Code)
- Business Process Threads

Understanding the Target System

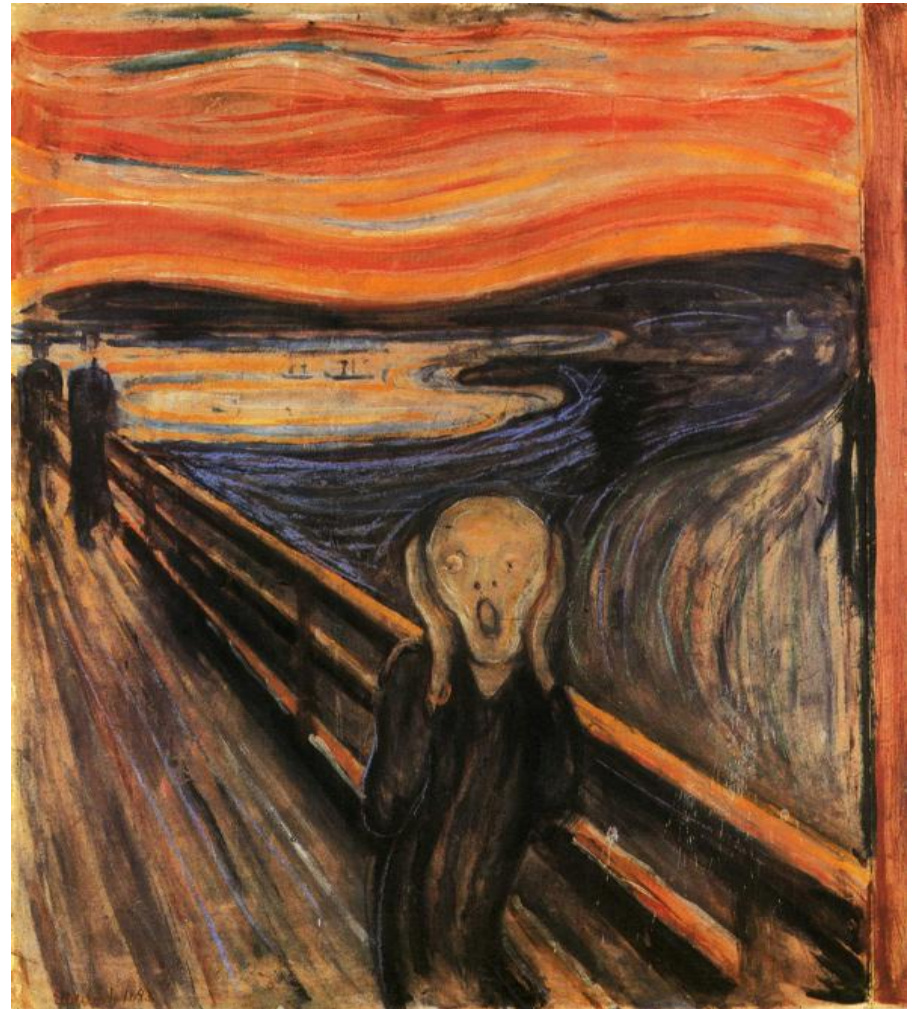
- Target Reference Architecture (TRA)
 - SOA; Layered Infrastructure
- Designing Services on top of TRA
- Business Process threads

Mapping between Legacy and Target in Phases

- Architecture mismatches (development, operational, certification, sustainment, COTS)
- Operating with dual authoritative data systems, cutover, synchronization
- Relationships between Application Components (Legacy versus TRA, COTS)

It's Worrisome

- **Is there too much code/ data coupling and spaghetti code to partition for migration?**
- Are the current business processes and screens appropriate?
- Is the CPI (and TRA) stable?
- Are COBOL to Java transformation tools up to the job?
- Can we operate with 2 systems overlapping authoritative data?
- Do we have sufficient technology expertise in: legacy system, target reference architecture, discovery and analysis tools, transformation tools?
- **Is the business logic only understandable in the legacy code?**
- How can we overcome the lack of architectural documentation?
- Will the entire testing and certification process change?
- **Will I be creating a maintenance nightmare?**
- It's a 24/7/365 operation- no downtime!



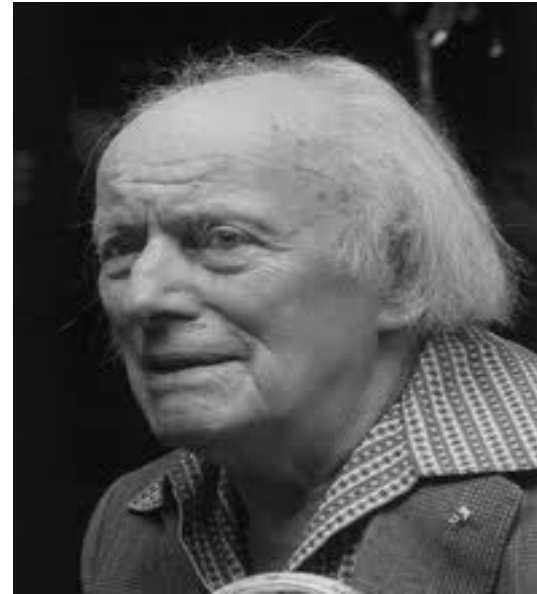
Keep the Right Vision



- **The TRA is a good start, but an application architecture with data modeling is needed**
- **Synchronize with phased TRA and CPI infrastructure deployment**
- Operating with multiple sources of authoritative data
- **Using new technology (e.g. customized security to infrastructure security)**
- Capability of transformative toolsets

Don't Live in a Dreamworld

- Big-Bang changes usually fail
- **Conducting transactions across networks and keeping response times satisfied!**
- Transforming spaghetti code automatically
 - Separating presentation from business processing from data access!
 - Moving from green screens to windows
- **Making multiple types of changes simultaneously!**
- **Edict no changes to the legacy system!**



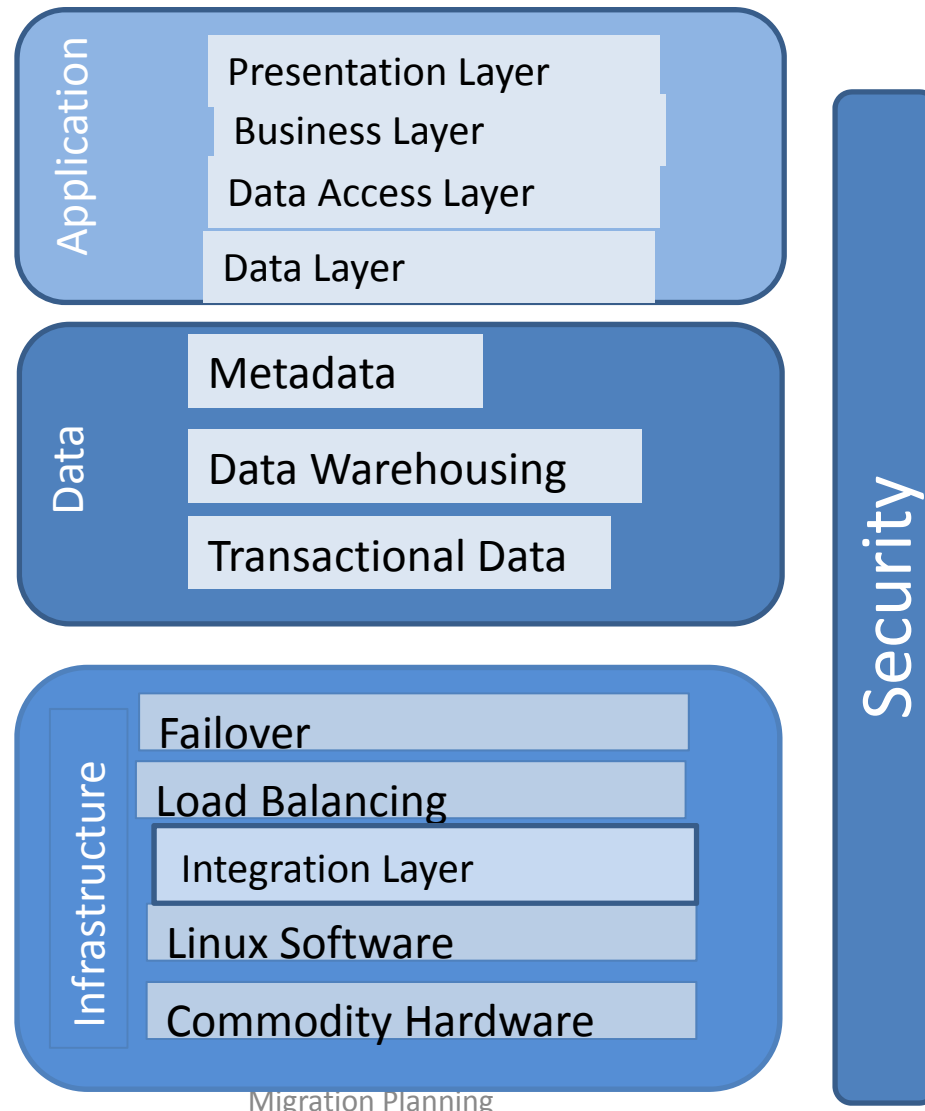
Paul Delvaux

Target Reference Architecture

This is a good start
But it is not enough

Need a system and
application software
architecture

- End-state
- Each Release



Migration Approaches

An RFI was conducted and multiple approaches suggested

- Lift and Shift (L&S)
 - Reduce Operations and Maintenance
 - Fast retirement of legacy infrastructure
 - Move to the CPI
 - Don't come close to satisfying the
 - Make minimal changes- keep COBC
- Modernize
 - Phased transformation to TRA
 - Keep legacy architecture
 - Change technology- Java, S
 - Use Conversion tools?
- Re-engineer
 - Re-Develop the system in Java or
 - Developed system and application architecture
 - Satisfy TRA

Options Considered

1. Do nothing (baseline)
2. L&S (baseline)
3. L&S and modernize
4. L&S and re-engineer
5. Re-engineer
6. Hybrid- L&S, modernize, re-engineer

Migration Process

Determine and Score Options

Explore implementation alternatives for options

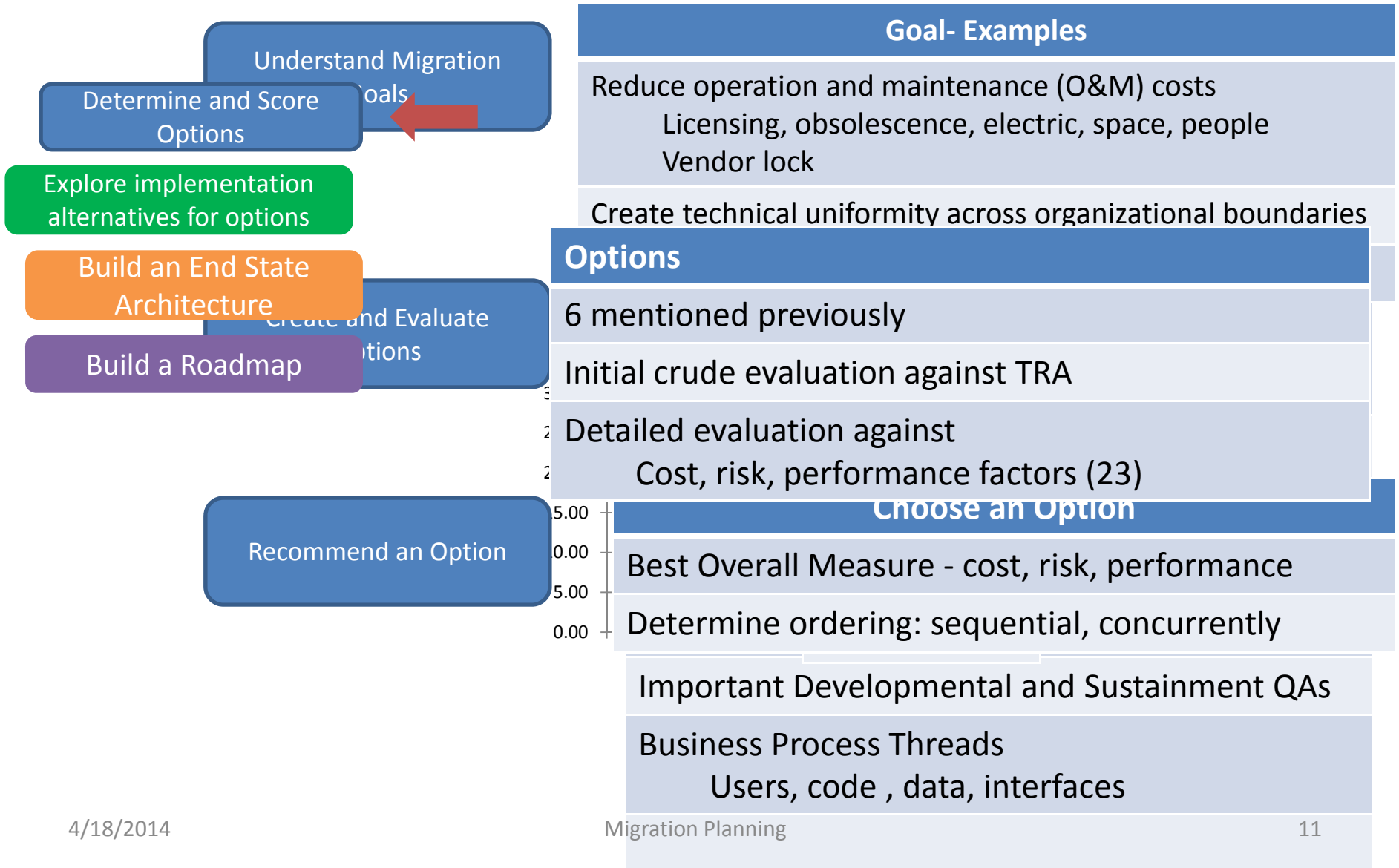
Build an End State Architecture

Build a Roadmap

List the Options
Develop evaluation criteria
Score the options

- Goals for sequencing
- Constraints on Phasing
- Approach to Migration
 - Data, code, user, business processes ordering
 - Quality attribute considerations
 - Migration tooling
- Define phases
 - Groups
 - Legacy: functionality, code, data BP, users
 - Tiers/layers
 - Transient code in Legacy and TA
 - Throwaway
 - Align with Infrastructure Roadmap

Determine and Score Options



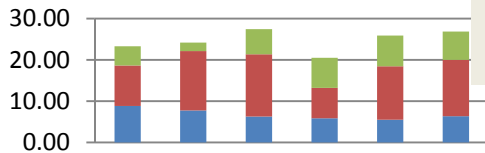
Evaluation Factors

Cost

- How much will be the migratic labor cost?
- How much recovery of investment cost after cancellation
- How early will cost savings happen- license and support during migration?
- How much will be the on-g legacy license and/or support costs post migration?

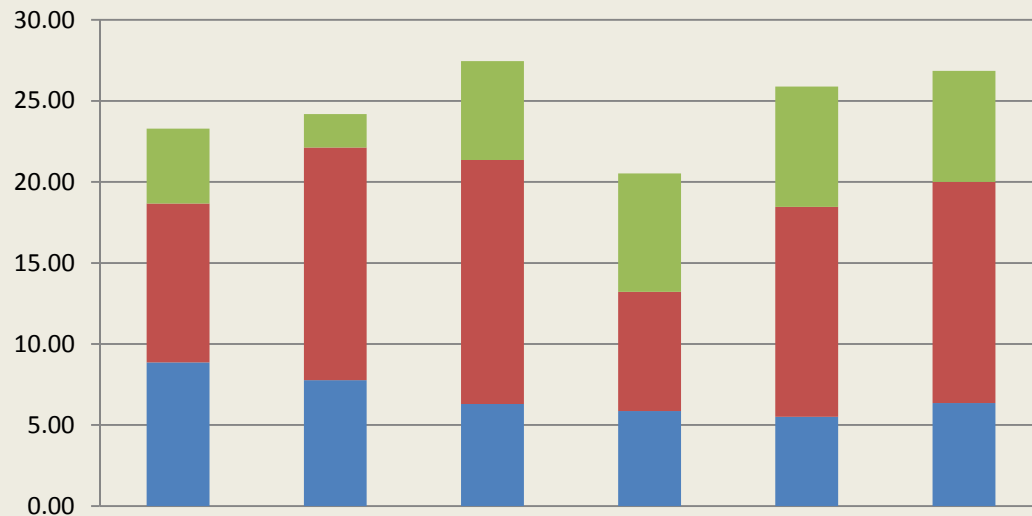
We dis
• Me

W	Factor	1	2	3	4
3	How is data organized wrt TA : relational, normalized, distributed, partitioned	1	1	5	8



4/18/2014

Performance



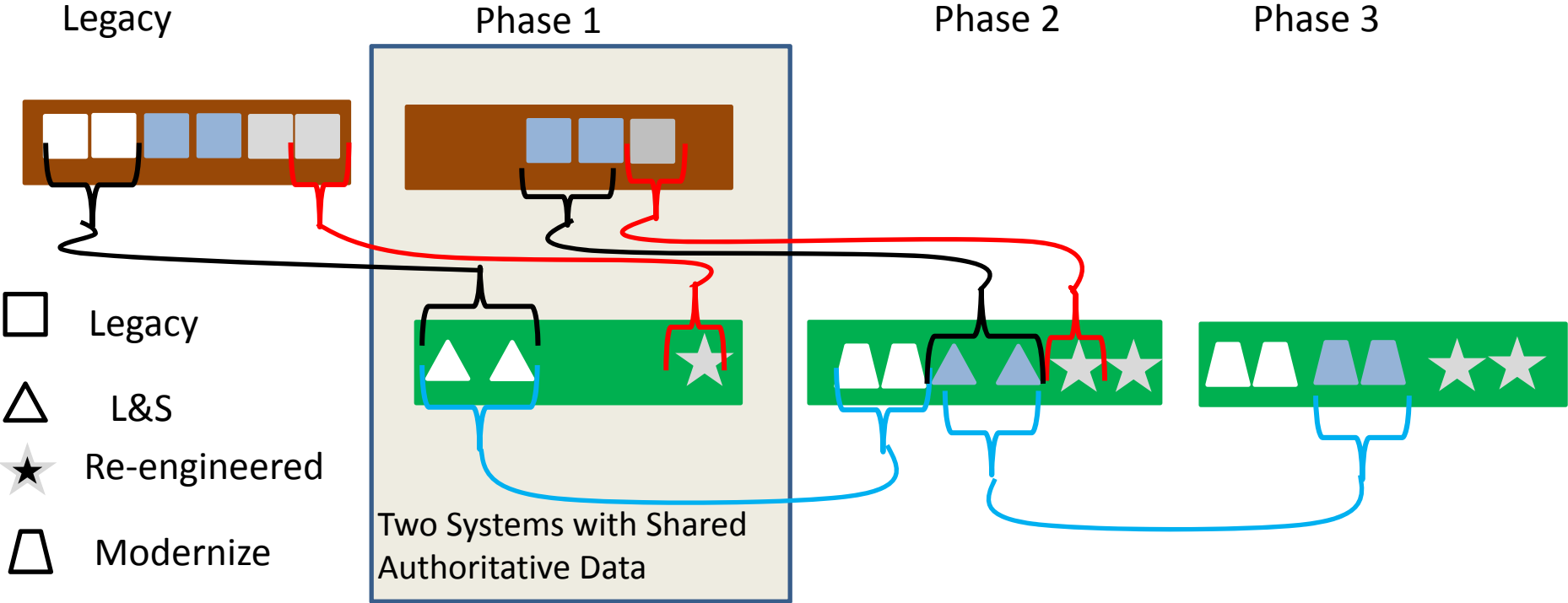
- How modernized are the user screens wrt the TRA

Migration Planning

Risk

a for
on
the
curve
future
will be
oes out

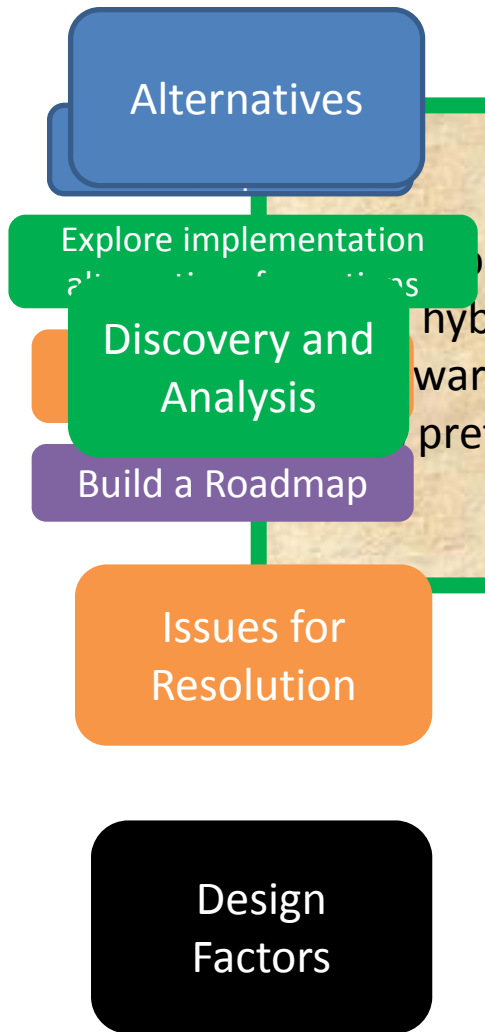
Selected Hybrid Option- Example



How to partition: L&S and modernize; and Re-engineer

- Significant Business Process Changes – re-engineer
- Layered stable legacy code- L&S and modernize
- Experiment with modernization tools

Explore Implementation Alternatives- Lift and Shift



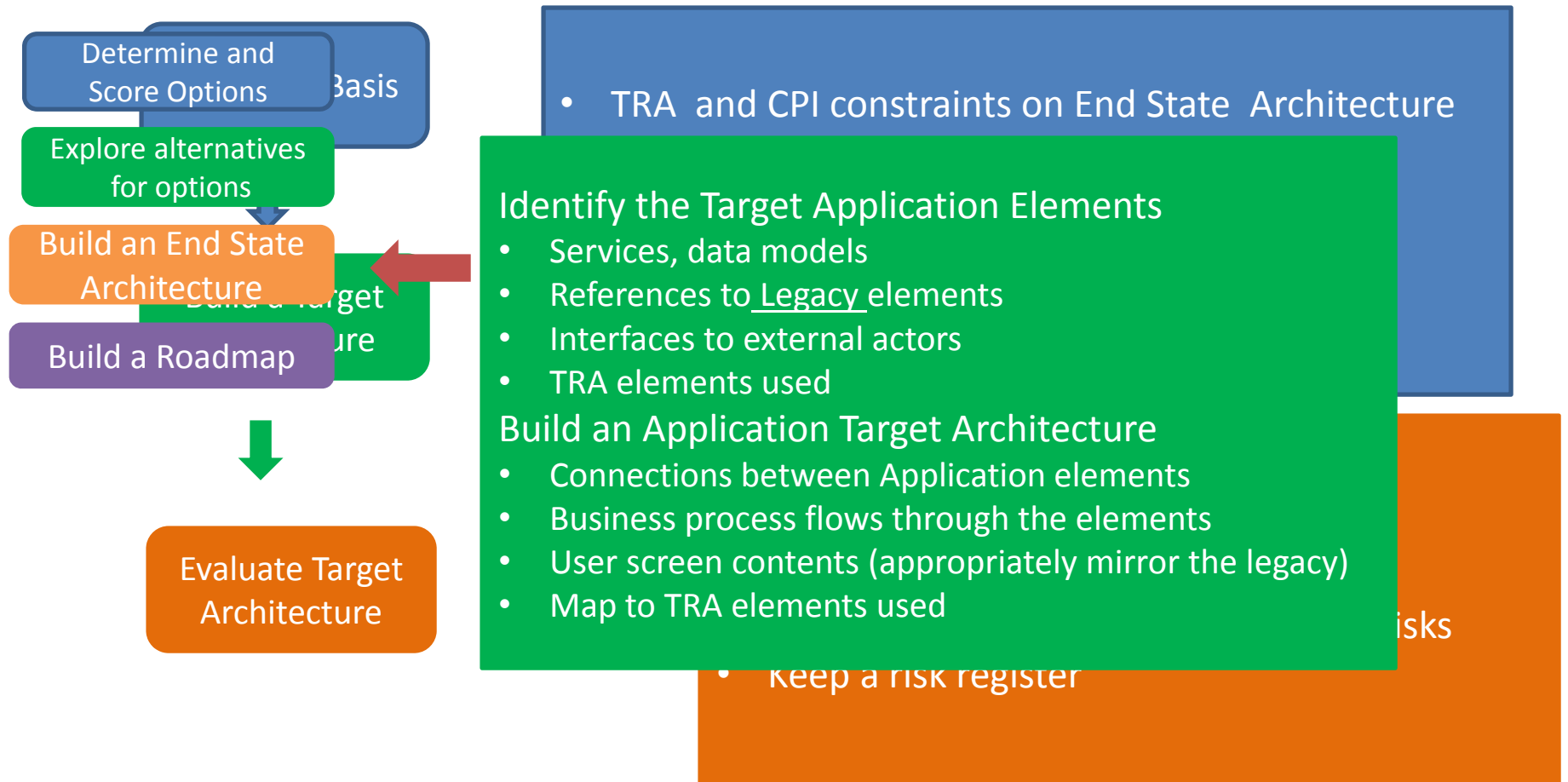
Finds the relationships between

- The subsystems and their programs
- The Data Stores and their files
- The subsystems and the data stores
- End-to-end business threads/User classes/subsystems/data stores
- Identify Dead code and dead data
- Legacy API's usage
- Legacy transactions usage
- Batch usage

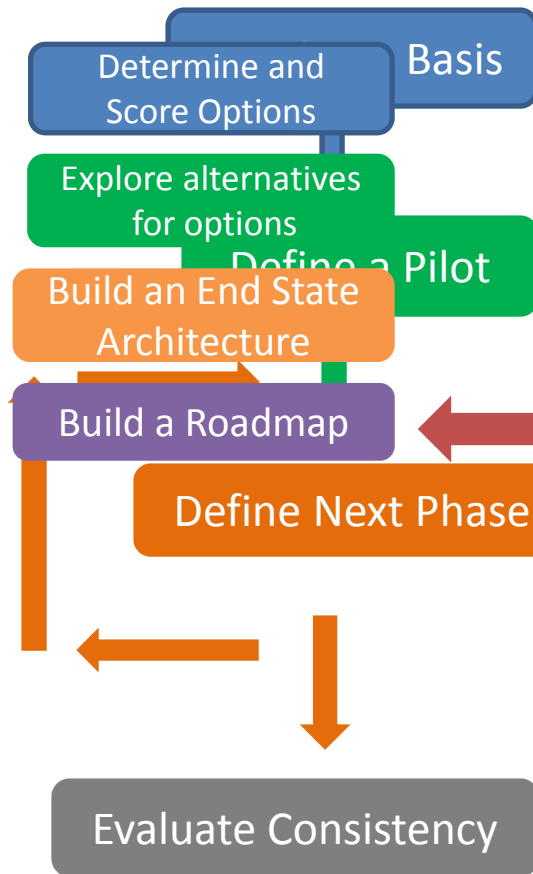
What commercial toolsets are best to do this?

	copying data, modifying APIs & JCL, writing adaptors?	
3	Replacing transactional support	
4	Should we do some cleanup before L&S	
5	Performance	

Build an End State Architecture



Build an Architectural Roadmap



Select a Pilot Migration for First Phase

- Lightweight, low risk, few challenges
- Gain *target* and *transformation* tool experience
- Include data extraction, analysis, and reporting
- Perhaps no cutover
- Happy path functionality
 - But include some quality attribute drivers
- Aligned with TRA infrastructure development schedule
- Identify target architectural elements needed
 - Business Process Threads, Screens
 - Services, data models, COTS tools
- Identify legacy architectural items being migrated
 - Screens, Reports
 - Subsystems (partial), tables (partial), data sources

Identify technical challenges and risks
Tools to assist migration

Summary

Technical

- Criteria for selecting between options
 - OMB guidelines were good
- Plan, but don't overdo it
- Architect
 - Understand Legacy
 - Include quality attributes
 - High Level End State
- Roadmap
 - Detailed architecture at each phase

Management

- Changing political environment
- Drift and Re-direction can become a way of life