Experiences in Migrations of Legacy Systems

Bill Wood, Mike Gagliardi, and Phil Bianco

Software Solutions Conference 2015

November 16–18, 2015

Software Engineering Institute | Carnegie Mellon University

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.

DM-0002952

Background

An agency wanted to migrate two paired and tightly coupled systems

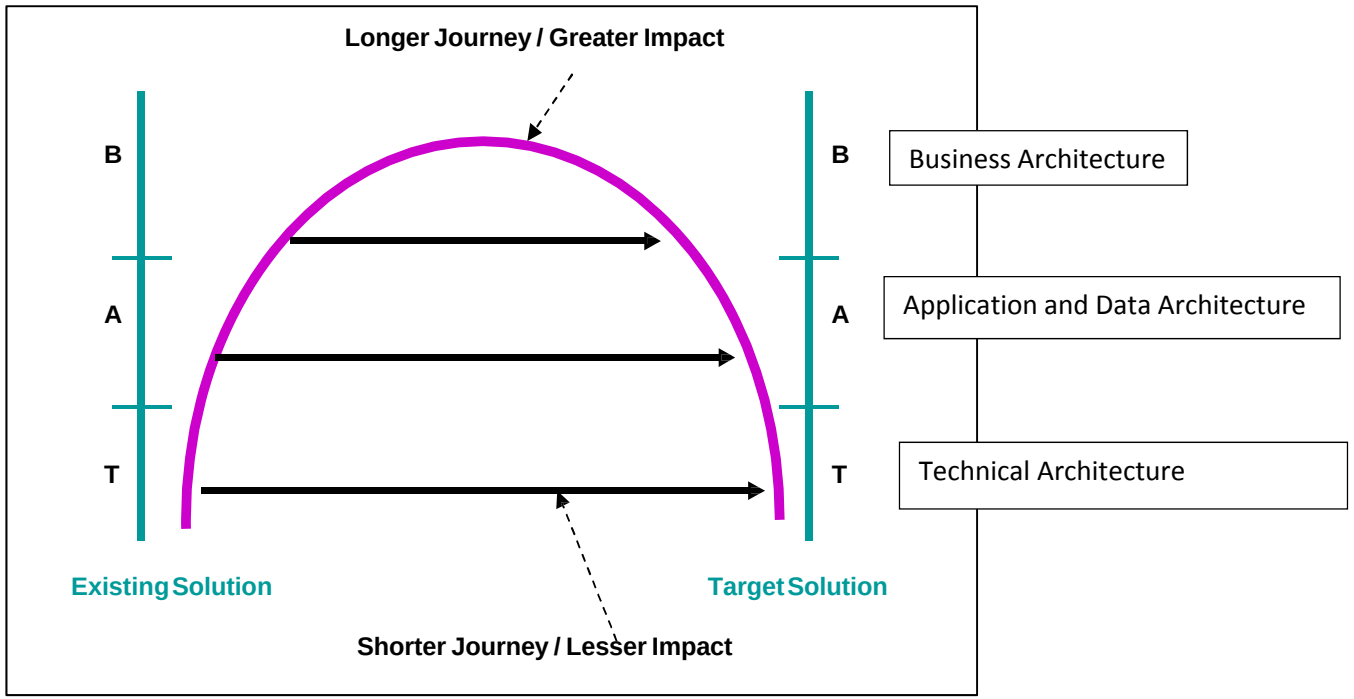
- Both are 24/7/365 and support disaster recovery at multiple sites
- Both have many classes of users

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

Horse Shoe Model



Phases of Our Legacy Migration Activities

RFI Analysis per Response

		A	B	C	D	E	F	G	H	I	J	K	L	M	N
RFI-Specific Questions															
Q1 Migrate Hierarchical DB DB															
Q2 Migrate Application Code															
Q3 Integration & Testing															
Q4 Application Maintenance															
Q5 Acquisition & Contracting															
Q6 Past Performance															
Technical Approach Analysis Framework															
F1 Design / Analysis	Code														
	Data														
	Docs														
F2 Migration Code, Data, Infra (Q1 & Q2)															
F3 Integ & Test (Q3)															
F4 Sync & Cutover	Sync														
	Cut														
F5 App Maintenance (Q4)															
Approach 1-4		1	3	2	2	2	4	4	4	2	2	1	1	4	4
Cost and Timeframe ROMS															
Cost (ROM) $ Mil		16	12	30	5-10	28			30	5-10		24			
Timeframe (ROM) Months		24	18	24	24-48	24	24		12	9-12		24			

Green: Explicit

Yellow: Implicit

Red: Ignored

Pros and Cons of Top 4 Alternatives

	Description	1	2	3	4	Explanation for Yellow
1	# of data of record systems	2	1	1	1	More than one authoritative data source is a synchronization and fault management challenge
2	Move over users to the new system incrementally	Yes	No	No	No	The confidence from the use of the functionality is delayed until everything is cut over
3	Initialize new database with old data	Incremental	Big Bang late	Big Bang early	Big Bang late	Big Bang Early requires long-term synchronization
4	Streaming data between systems during phases and over months	Forward and back	None	Forward	None	Subset of 1 (above)
5	Fallback due to new system failure during development	Recover new system and re-synch	Not needed	Multiple ways	Not needed	Subset of 1 (above)
6	Fallback due to new system failure after cutover	Recover new system and re-synch	None	Defined ways	None	Problems will always arise after cutover. Need to design for fallback.
7	Queries executed during development	Hierarchical new DBMS	Hierarchical	Hierarchical	Hierarchical	Increased complexity for query management.
8	Benchmarking for performance	By testing and operation	By testing	By testing	By testing	You can't operationally benchmark until cutover; performance issues will be discovered late.
9	Data/code coupling	Separable into segments	Separable into segments	Too tightly coupled for separability	N/A	If the code and data coupling is overly complex, will be difficult to segment.
10	Legally mandated updates during development before cutover	In both code and data if moved; in one if unmoved	In both code and data if moved; in one if unmoved	In both code and data if moved; in code if one is unmoved	Code and data in both	No response; handled this very well.

Green: Preferred

Yellow: Not Preferred

It’s Complicated

Understanding the Legacy System Architecture

- Infrastructure tools
- Application component relationships (data and code)
- Business process threads (BPTs)

Understanding the Target System

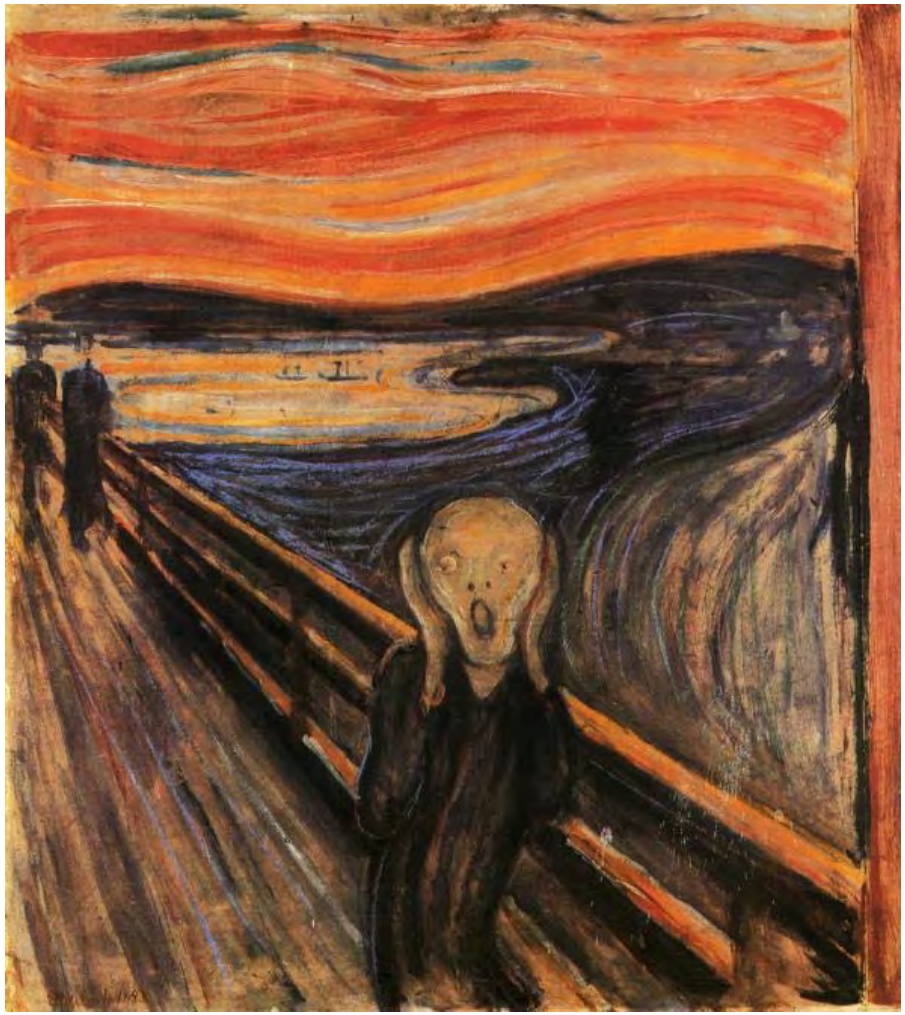
- Target reference architecture (TRA)
  - SOA; layered infrastructure
- Designing services on top of TRA
- Business process threads

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 vs. TRA, COTS)
- Ineffective BPTs

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 stable? Is the TRA stable?
- Are COBOL-to-Java transformation tools up to the job?
- Can we operate with two systems overlapping authoritative data?
- Do we have sufficient technology expertize in legacy system, TRA, 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 create a maintenance nightmare?
- It's a 24/7/365 operation – no downtime!



Experiences

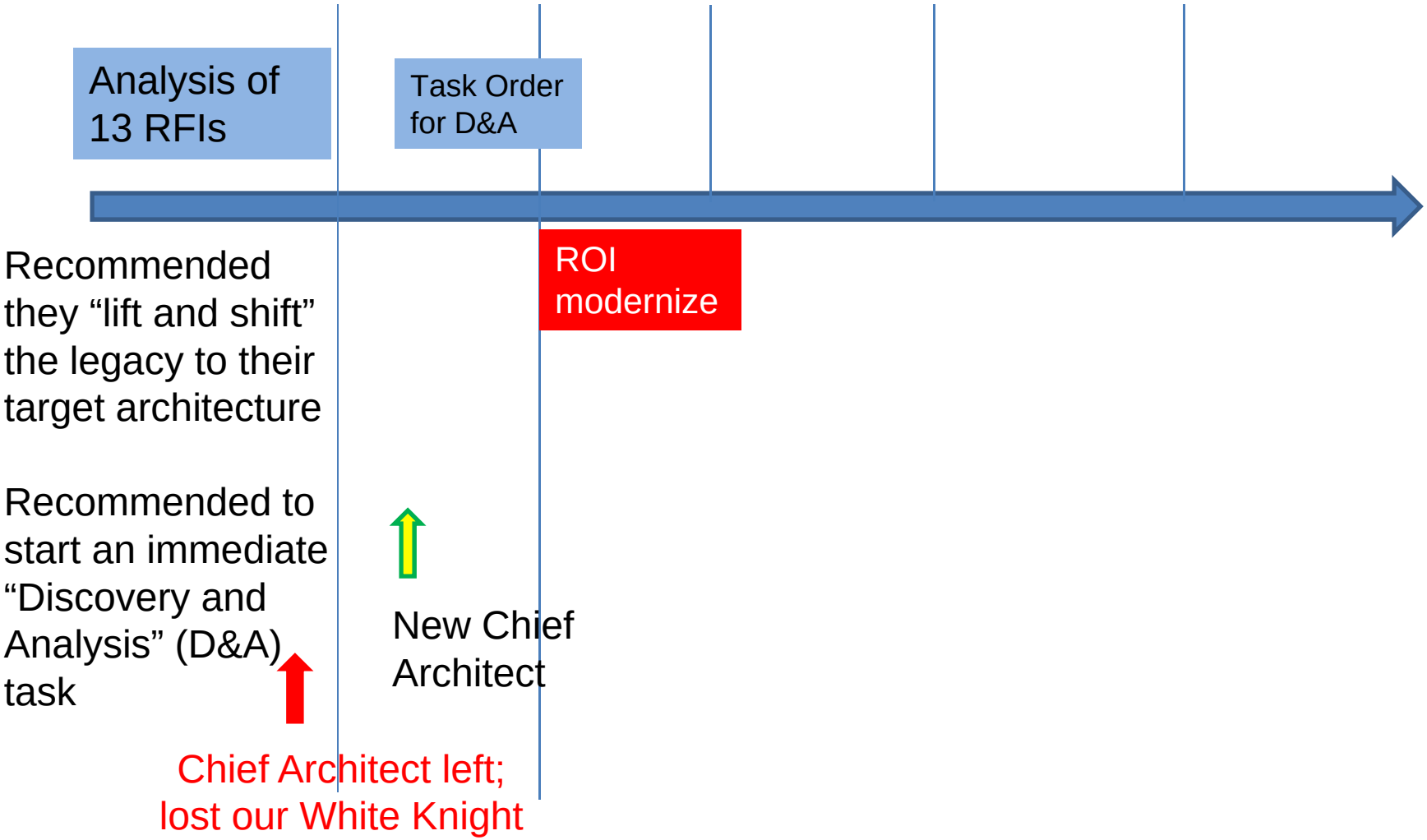
Customer

- Rush to use this data to get started removing legacy
- Different and misplaced emphases between people

SEI Team

- Strong differences of opinion during the analysis
- Schedule driven; a small subset completed it
- Proposed alternatives and a migration plan

Phases of Our Legacy Migration Activities



Experiences

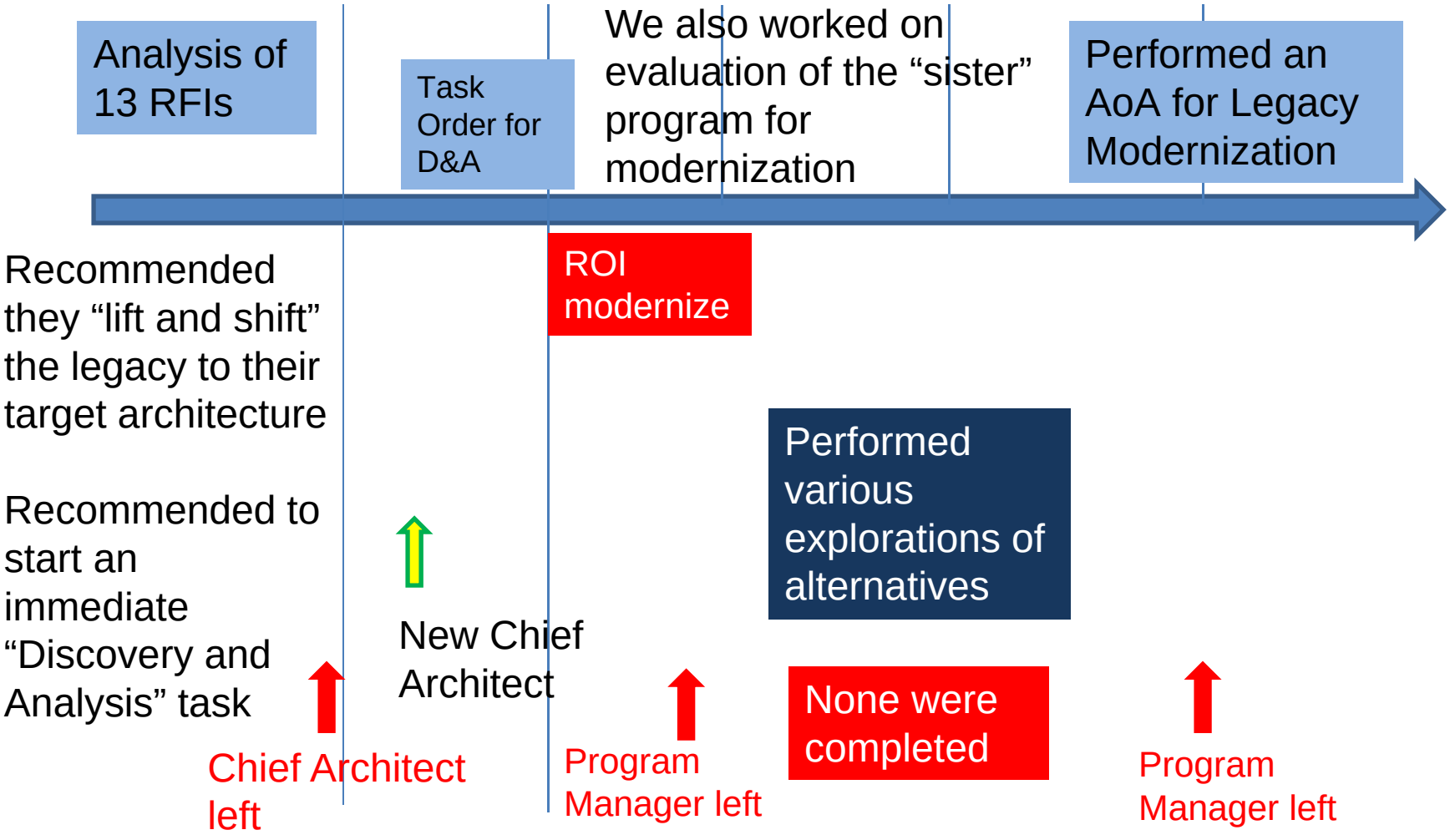
Customer

- Built a task order with costs for Discovery and Analysis (D&A)
- Forced to build a return on investment (RoI) for complete modernization
- They were not able to get money for the D&A

SEI Team

- Built the task order
- Supported the ROI effort
- Frustrated by lack of $

Phases of Our Legacy Migration Activities



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!
- The TRA is a good start, but an application architecture with data modeling is needed
- Operating with multiple sources of authoritative data is a concern

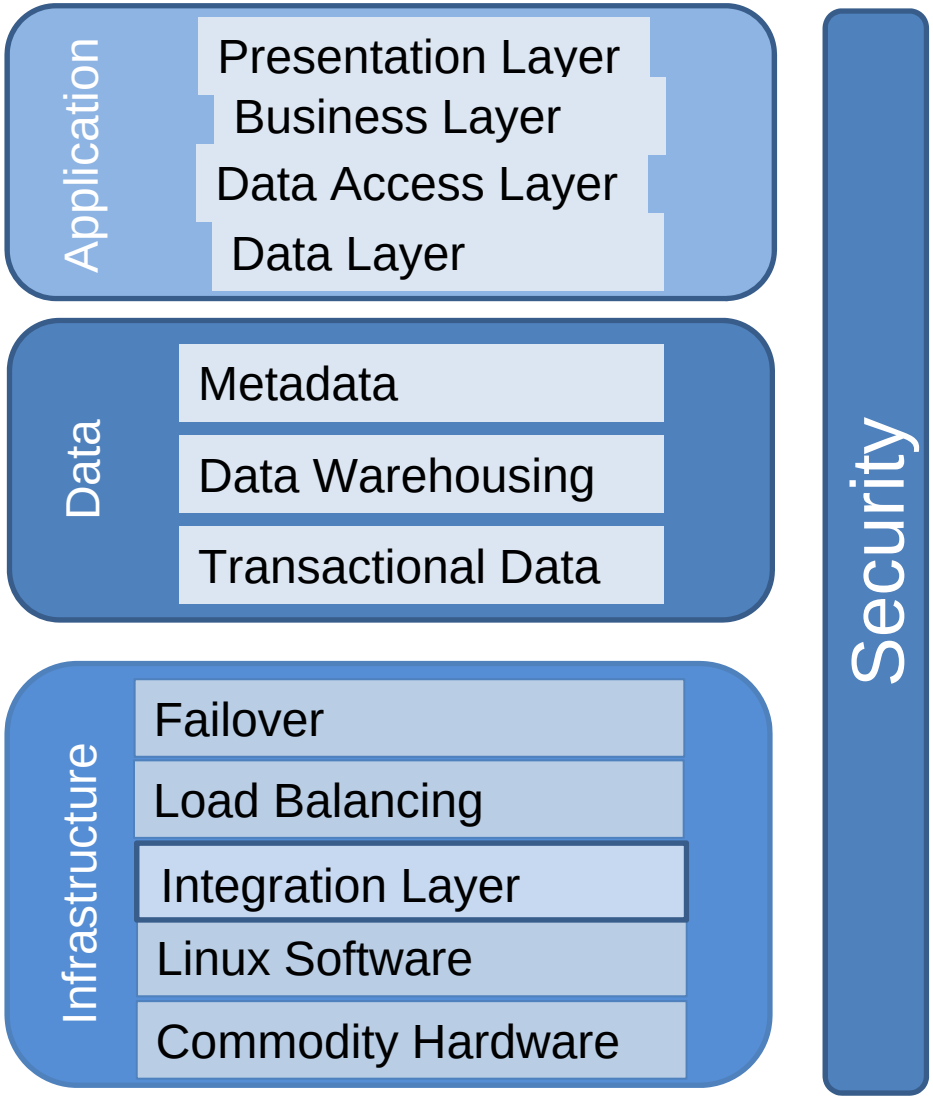
Target Reference Architecture

This is a good start

But it is not enough

Need a system and application software architecture

- End state
- Each release



Migration Approaches

Options considered based on RFI proposals

1. Do nothing (baseline)
2. L&S (baseline): switch to relational DB and new platform
3. L&S then modernize
4. L&S then re-engineer
5. Re-engineer
6. Hybrid: L&S, modernize, re-engineer

Migration Process

Determine and score options

List the options
Develop evaluation criteria
Score the options
Make a selection

Explore implementation alternatives for 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

Build an end-state architecture

Build a roadmap

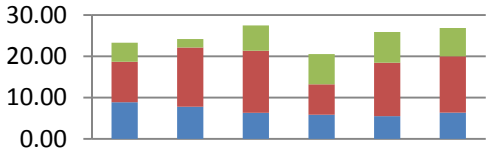
Evaluation Factors

Cost

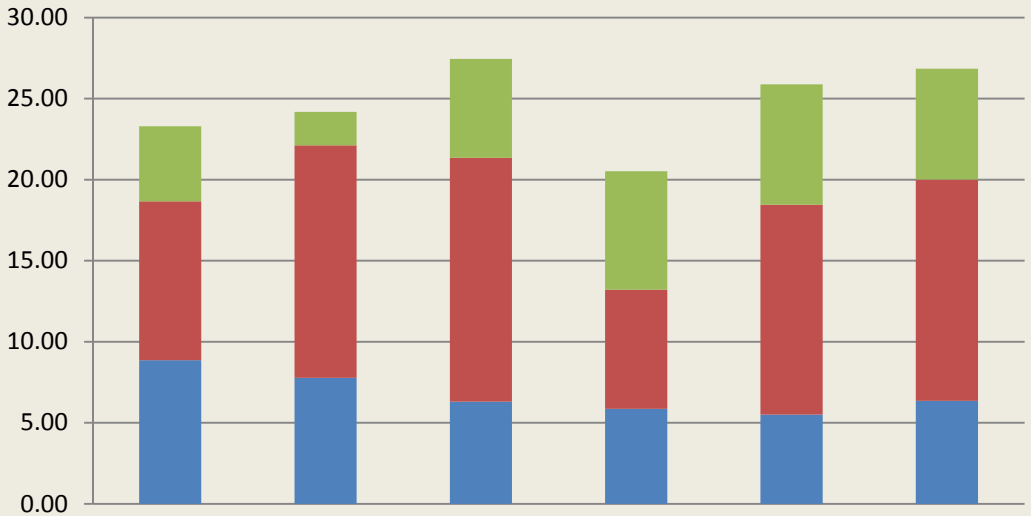
- How much will the migration labor cost?
- How much recovery of investment cost after cancellation?
- How early will cost savings happen? License and support during migration?
- How much will the ongoing legacy license and/or support cost post migration?

We discov
• Merge

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



Performance



- How modernized are the user screens wrt the TRA?

Risk

for
on
he
curve,
ended
A,
Users
be
ture
will be
es out

Summary

Technical

- Included OMB guidelines in evaluation factors
- Fit quality attributes in evaluation factors
- Plan – but don't overdo it

Management

- Changing political environment is hostile to technical work
- Need to have PoC at management decision-making level
- Drift and redirection can become a way of life
- Management happy with the summary sheet
  - Don't overdo the supporting details
- Did the system engineering in an Agile manner