

## “Legacy Evolution” – The Innovation Opportunity

Dave Thomas

[dave@bedarra.com](mailto:dave@bedarra.com)  
[www.davethomas.net](http://www.davethomas.net)

©2010 Bedarra Research Labs. All rights reserved.

## Outline

1. Legacy Evolution Value Proposition
2. Typical Code Driven Approaches
3. A Lean Data and Flow Driven Approach
4. Leverage Technology Innovations
5. Management Buy In and Risk
6. Innovation Opportunities/Insertion Points
7. Innovation Patterns



©2009 Bedarra Research Labs. All rights reserved.

## Legacy?

A Legacy is an application of substantial value to the business that requires a major evolution to meet the needs of the business.

### Common Properties

- Older language/platform ...
- Lacks documentation
- Lacks tests
- Lacks humans who know the code base and/or domain

©2009 Bedarra Research Labs. All rights reserved.

## Legacy Evolution Value Proposition

1. Improve Access to Data
2. Enable Access to Federated Data Sources
3. Enhance/Change Functionality (Business or Regulatory)
4. Reduce Time/Cost of Processing

### *Often Cited but ROI Questionable?!*

1. Improve Maintainability (Reduce Technical Debt)
2. Modernize to retain Staff (facilitate above)

©2009 Bedarra Research Labs. All rights reserved.

## Legacy Code ? No Fear ... 1, 2, 3 Charge!

*Our vendor, our consultant, our outsourcer, our team  
has the solution !!!*



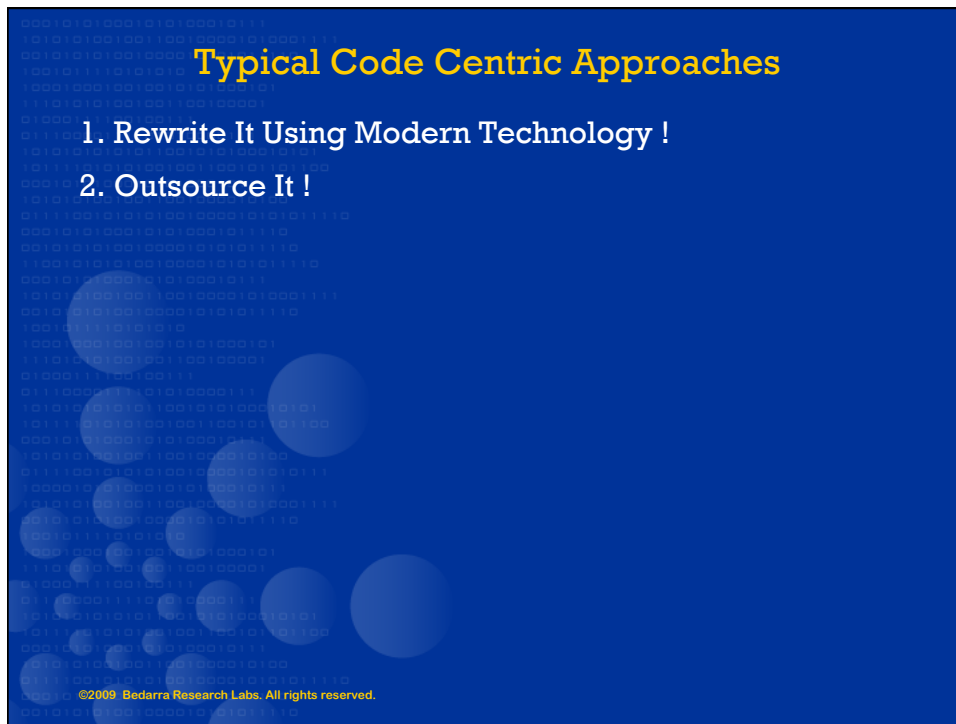
*It will take years and lots of money but we will tame  
the code of legacy mountain*

©2009 Bedarra Research Labs. All rights reserved.

## Typical Code Centric Approaches

### 1. Rewrite It Using Modern Technology !

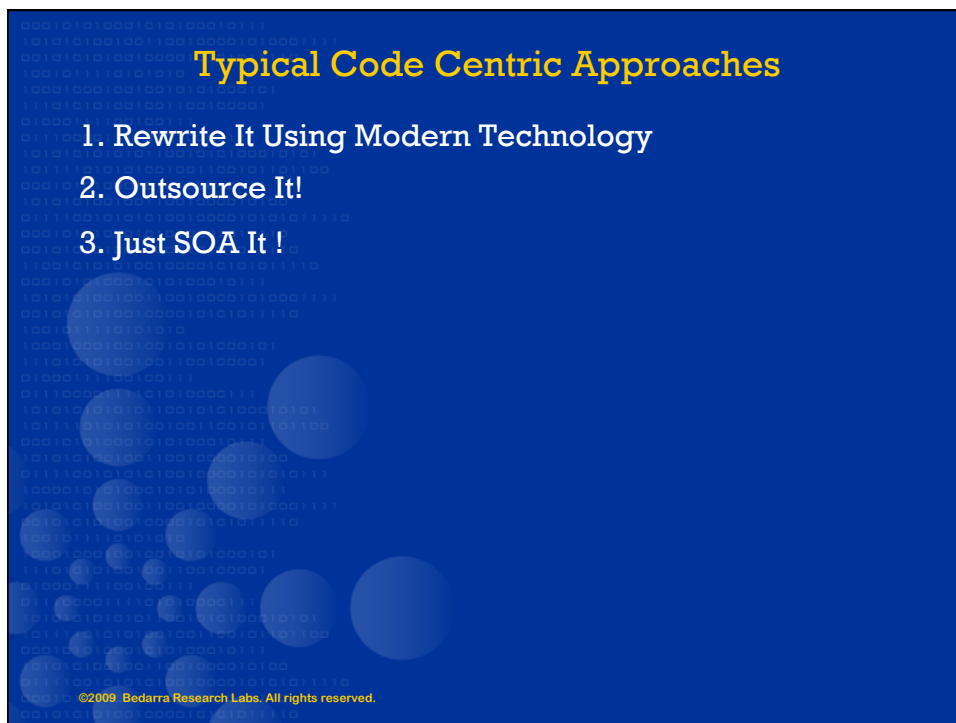
©2009 Bedarra Research Labs. All rights reserved.



## Typical Code Centric Approaches

1. Rewrite It Using Modern Technology !
2. Outsource It !

©2009 Bedarra Research Labs. All rights reserved.



## Typical Code Centric Approaches

1. Rewrite It Using Modern Technology
2. Outsource It!
3. Just SOA It !

©2009 Bedarra Research Labs. All rights reserved.

## Typical Code Centric Approaches

1. Rewrite It Using Modern Technology !
2. Outsource It !
3. Just SOA It !
4. Agile It !

Focus is on systemic changes to two or more of

- code
- people
- technology

Assume a substantive base of requirement and tests which in them selves can be expensive to create/evolve.

©2009 Bedarra Research Labs. All rights reserved.

## Lean Data and Flow Centric Approach

Why focus on Data and Flows?

- Need to find targeted opportunities high value intervention
- Data is the largest and most stable corporate asset
- Data transformations are the primary function of an IT system
- Often the easiest insertion point can be easily monitored

©2009 Bedarra Research Labs. All rights reserved.

## Selective Code Focus

- Small computational bottle necks
- Highly structured rules/calculations
- Points of high variability/constant change
- Points with large numbers of defects

©2009 Bedarra Research Labs. All rights reserved.

## Leverage Innovations

### Improved Business Practices

- Simplification, Partnering, Regulatory ...

### Improved Hardware

- Performance, Capacity, Latency, Functionality – atomic clock, mobility, voice, cloud ...

### Improved Software

- Usability, Productivity, Flexibility – NoSQL, JSON, REST; Fault Tolerance – isolation, BASE vs ACID, Languages, Frameworks...

### Improved Software Practices

- Agile, Micro Services, Continuous Deployment, Algorithms, Self Service ...

©2009 Bedarra Research Labs. All rights reserved.

## Management Buy In – ROI and Risk Mitigation

### Business

- Clear and tangible measureable goal
- ROI model shows significant business value (5x, >10%)
- Strong Senior Experienced Business Sponsor
- Implementation Timeline of 3 months, max 5
- Minimal Impact on Business Operations

©2009 Bedarra Research Labs. All rights reserved.

## Management Buy In – ROI and Risk Mitigation

### Technical

- Single small team tech plus business with track record
- Localized code changes
- Proof of Concept validation in weeks
- Proof of Scale validation in weeks
- SLA easy to monitor
- Life cycle costs...e.g. special skills, dependency partner?
- Straight forward DevOps deployment
- Minimum dependency
- Independent Acceptance Testing

©2009 Bedarra Research Labs. All rights reserved.

## Innovation Opportunities/Insertion Points

### Data and Flow Interfaces

- Database Interface
- File Interface
- Sterilization Interface
- Messaging Interface
- Functional Transformers - ETL Interface; Map Reduce; GPU ...
- Query Interface
- Disk/San Interface
- Shared Memory Interface
- Log Interface – Event Sourcing; Historical DB (Batch) + Realtime DB (Stream)
- Sync Replicate Interface - CDN leverage; Mobile Occasionally Disconnected; BASE Fault Tolerant
- Reactive MVC Interface

©2009 Bedarra Research Labs. All rights reserved.

## Innovation Opportunities/Insertion Points

### Target Code Focus

- Engines
  - State Machine
  - Rule Machines
  - Logic Machines
  - Constraints
  - Data Flow
- DSL Spec by Example => Programming By Example – Self Service
- Simple SIMD computation may allow GPU or cluster of simple multicore
- Independent isolated computations may allow multi core cluster/distributed

©2009 Bedarra Research Labs. All rights reserved.



## Legacy Innovation Patterns

- Make it Table/Data Driven ; Code => Tables | Rules | Constraints
- Make it look like a data base => ODBC for x
- Make it look like a collection => LINQ/Rx
- Make it look like the Web – Integration => HTTP/ATOM/REST vs APIs
- Open Data/ Data Standards vs Open Source – JSON, Linked Data

©2009 Bedarra Research Labs. All rights reserved.

## Legacy Innovation Patterns

- Move Beyond Simple TDD – Random, Independent Implementation
- Make it ALL Query capable – Search; Map-Reduce; Dynamic Schema
- Use Micro Services – Actors (isolation and messaging) Actra/ Harmony
- Give it to me Raw – Legacy Data – high performance serialization ; database refactoring ... from physical storage
- Batch It and Stream It– Historical + Real-time DB; Event Sourcing

...

©2009 Bedarra Research Labs. All rights reserved.

## Lots of Opportunities!

- Use the Cloud for Testing
- Use Ruby, Clojure ... for Scripting Tests SPE/BDD, DevOps Puppet
- Use NoSQL for speed/cache => SQL for reporting
- OLAP => High Performance FP
- Continuous Release and Deployment => DevOps Clouds
- Use F# or Scala for algorithms, C# and Java for *middlware*
- Engineering & Financial Models => SPE -> PBE

©2009 Bedarra Research Labs. All rights reserved.

*Embrace your Legacy  
and  
Innovate in It!*

*Thanks!*

©2009 Bedarra Research Labs. All rights reserved.