Once Upon A Legacy

1. Dirty Jobs or Fame & Fortune?
2. No Fear - 1,2,3 Charge!
3. Agile in a Legacy Environment?
4. Time Something Completely Different?
5. Thinking Out of The Legacy Box - Case Studies of Legacy Innovation

"Legacy Code is code that has No tests"

- has insufficient tests
- is older than 3 years
- technical debt written by others . . .
- is written in a language, tool, platform we don’t know or like
- is difficult to understand or change . . .

70 - 90 % of Software Budgets are spent on Legacy Systems!
**Legacy - The Big Ball of Mud**

---

**Legacy = Big Ball of MUD which has significant business value!!**

---

**Oh My! It is our mess (debt)!!**

---

**Legacy? No Fear ... 1, 2, 3 Charge!**

---

© 2010 - 2011 Bedarra Research Labs. All rights reserved.
Rewrite It Using Modern Technology

Rewrite in my language/platform
  high risk - most major rewrites fail
  rewrites seldom provide sufficient business value
  rewrites can be more difficult to maintain than the legacy

Automatic Migration - Rewrite Magic
  Migration to equivalent runtime perhaps
  Translation to another language is seldom worth it

Second Systems Effect: Read Mythical Man Month AGAIN!

Outsource It!

Out Source: Make it someone else’s problem
  • may work for a short time
  • risks of loss of skills, knowledge of code base and domain knowledge
  • increased investment in requirements and testing
  • Stress fractures where local systems touch outsourced systems compounded by different processes and tool chains

Just SOA It!

Just Use SOA: Wrap It Into Services
  • Assumes you have interfaces
  • Assumes you have tests
  • Assumes the service is easily surfaced and encapsulated when in practice this may be buried in several modules
  • Assumes that ESB/XML/Process Server Performance, RMA is acceptable when it often is unknown
  • Analysis paralysis between enterprise architects, vendor architects and development teams
  • Which ESB, Process Server ...
  • SOAP, REST ...
  • XML … J SON …
  • BPM/BPEL …
Do you believe in Magic?

- Wizard's work
- Absence of tests significantly increases risk
- Tools are not up to refactoring legacy code bases or databases
- Off the shelf tools may not be available for your legacy language
- Takes too long for the business to sustain it

“I don’t know about you... but I find these 'legacy' refactoring tools leave a lot to be desired … Yeah, Let’s try some dynamite”

But Can Help!

- Archaeology
- Agile Team
- Tool Chain
- Best Practices

Legacy Archaeology

- Legacy Story Telling
- Listen to the dev and ops Data
- Look at the Code and Data Assets
- Look at the development value chain - people, process and tools
- Publish the Stories, Data and Value Chain and put the essence on the wall
Legacy Agile Team
- appropriate skills for all areas
- cross train/pair Legacy and non-Legacy Developers
- Use playing Coaches who know the territory

Legacy Agile Tool Chain
- Workstation IDE with cross development
- Modern SCM environment (in front of legacy SCM)
- Unit and Acceptance Testing tools
- Continuous Build and Test appropriate to the technology e.g. HW/SW Emulators
- Sufficient Disk space for logging and testing

10 Year Old Legacy Java Application

Challenge
- unsupported 10 yr old Java 1.3 Swing, EJ B 1.2 application
- application sits in middle of existing core systems
- urgent need to implement major regulatory

Solution
- Agile practices BDD, TDD ...
- Major investment in automated testing provided an essential safety net
- Regulatory changes implemented to legacy platform
- In parallel re-platform code base & merge regulatory changes
A Few Practices for Legacy Developers

1. **Isolate variability**
2. **Make it data driven** - validation, decision tables, state tables, constraints ... it allows changes at runtime!
3. **Invest in mirroring, sampling backups, caching, emulation** if you can’t test live ... Hardware is cheap!
4. If necessary consider **live database TDD Using Transaction Roll Back**

5. Screen Scrape if you must, but automate from data definition
6. When you renovate a schema ask for lots of space, add fields, blobs, extension tables
7. Use in-memory DB testing or a server-side ODBC proxy to split production DB changes from test changes
8. Enable continuous inspection (remote inspector) and monitoring or using system logging via web enablement RSS/ATOM

We need a miracle! Innovation!
Thinking Out of The Legacy Box
Case Studies of Innovation

1. Insurance legacy + EAP + ISV
2. HR Benefits System Bottleneck
3. Enhancing a 1M+ assembler product
4. Telecomm Legacy Product Enhancement
5. Banking Platform Migration
6. Factory Process Control Modernization
7. Massive Legacy Data Base Migration
8. Real Models to Code

Insurance Legacy + SOA + EAP

Challenge
Legacy L1, L2, L3, L4; Vendor EAP V1, V2, V3, V4, V5;
Integration Services I5, I6; Commercial Insurance V1, L1
Personal Insurance L2; Vehicle Insurance V2
Rating Engine L2, L3, V3; Billing System V4
Policy System of Record L4 + I5 + V5
Enterprise Software Bus I6

Solution
1. Outstanding BAs define all products in tables
2. Agile experts generate applications from BA tables
3. SI experts build simple interfaces to ESB + Interface Acceptance Tests
4. All vendors required to deliver acceptance tests

© 2010 - 2011 Bedarra Research Labs. All rights reserved.

HR Bottleneck

Challenge
- calculations very complex.
- analysts capture in Excel.
- 100 Devs in COBOL too slow
- Agile OO gave only 15% in productivity

Solution
1. Retain 2 Agile OO experts
2. Excel rule checker in java
3. Spreadsheet in java on mainframe
4. Legacy team deployed on other systems

© 2010 - 2011 Bedarra Research Labs. All rights reserved.
**Enhance Mission Critical Assembler Application**

**Challenge**
Enhance a 1M lines of assembler product
No legacy knowledge of code base

**Solution**
Translate assembler into Prolog rules
Use Prolog to answer dataflow, control flow questions to enable safe modification of the code

**Modernize Legacy Factory Automation**

**Challenge**
$$ investment in proprietary automation system
Must to integrate new systems and new GUI-based control rooms

**Solution**
1. Modify interface to control units using TCP/IP
2. Implement HTTP and ATOM
3. Client devices are Web 2.0 appliances or PCs

**Enhancing a Legacy Telecomm HW/SW Product**

**Challenge**
Successful Legacy Telecom product needs be upgraded
Product built in proprietary hardware, OS, languages ...
3 Attempts to Greenfield replace have failed!

**Solution**
1. Move to VME bus, coprocessor, TCP/IP and DMA to legacy processor
2. New capabilities run on new processor
3. Eclipse tools + legacy unit + CI servers
4. DMA interface enables monitoring and debugging
Migrate Banking Legacy Apps

**Challenge**
Legacy Hardware/Software Platform Too Costly to Maintain! 
Rewrite estimate exceeds budget and has huge risk!

**Solution**
1. Expert ISV ports software platform to modern commodity platform. ISV validates using ATs derived from monitoring.
2. Expert SI Migrates banking apps ATs using regression testing and selected new acceptance tests
3. Upgrade modern IDE tooling

**Total Cost** 10-20% of rewrite estimate in 25% of estimated time!

---

Product at Risk - Massive Legacy Data Migration

**Challenge**
Legacy Database Migration required due to DB Vendor Risk 
Product Vendor dependent on DB vendor
Customers want improved query and reporting
Customers have massive data locked in DB vendor at risk

**Solution**
1. Product Team implements ODBC interface to legacy data
2. Product Team migrates product to ODBC and provides improved query and reporting...
   **BUT** Customers and Product locked to legacy physical data!
   and Product needs to change schema!
3. Expert SI retained by Product to perform high performance bulk conversion to modern database. (Agile DB Refactoring)
4. Independently developed data comparison program used for acceptance testing

---

Real Models to Code

Use a high level language based DSL to describe model the intricate and often variable part of the application
Generate or tailor high performance code for the specific application

**Examples**
Smalltalk for Tailoring Analog Chip Simulators
Smalltalk for Tailoring Finite Element Analysis
Haskell and Scheme for Financial Models
Haskell for HW Models

---
Lots of Opportunities to Innovate!

Use the Cloud for Testing
Use Ruby, Clojure ... for Scripting Tests & DevOps
Use F# or Scala for algorithms, C# and Java for muddleware - popular in financial engineering
Use NoSQL for speed => SQL for reporting
OLAP => High Performance FP
Continuous Release and Deployment - DeVOps Clouds
Go Native - Provide access to the HW

Go for Gold!
Embrace and Extend Your Legacy

Be Agile! Think Lean!
Innovate! and Prosper!

Thanks!