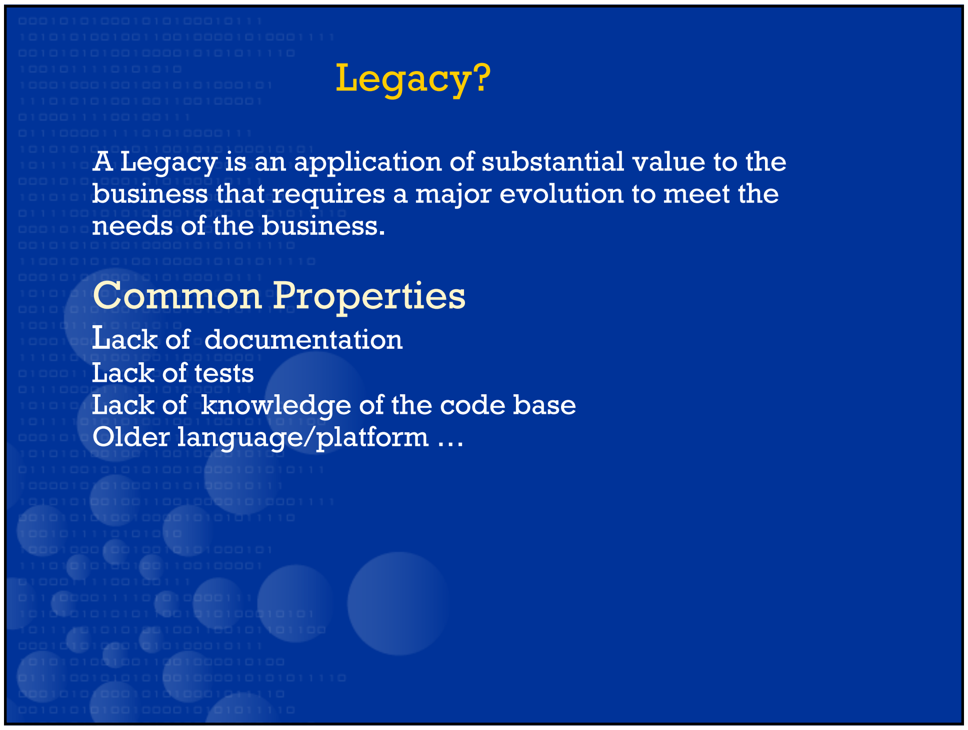
Legacy Evolution – The Innovation Opportunity

Dave Thomas

Chief Scientist Kx Systems,
Founder YOW! Conferences and Workshops

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 in Practice

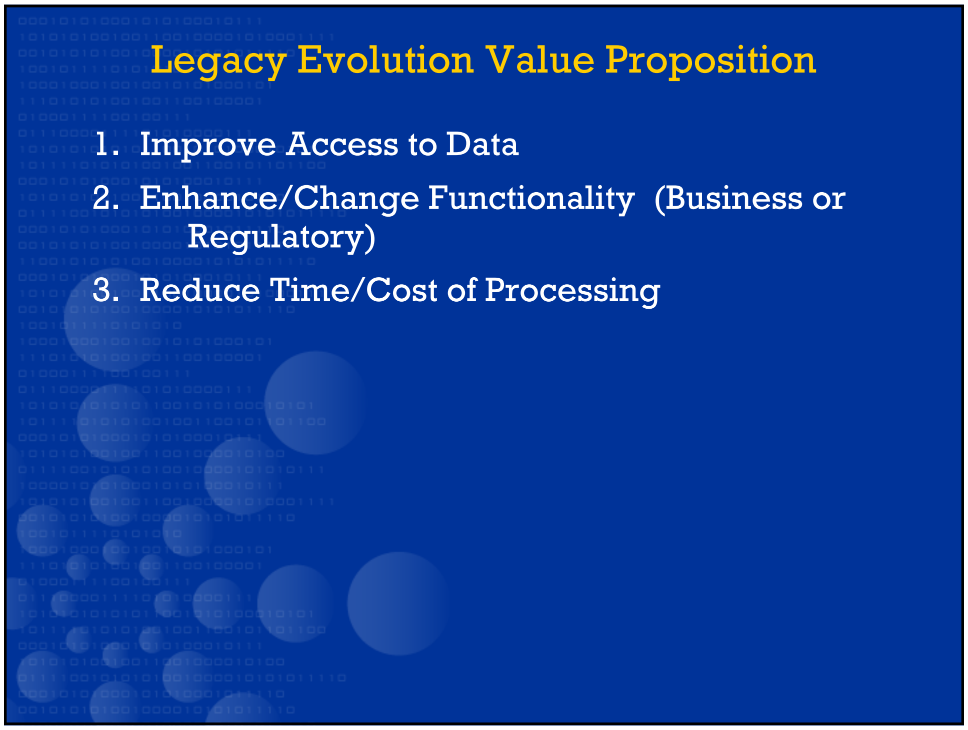


Legacy?

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

Lack of documentation
Lack of tests
Lack of knowledge of the code base
Older language/platform …



Legacy Evolution Value Proposition

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

Legacy Code ? No Fear … 1, 2, 3 Charge!

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

1. Outsource It!
2. Rewrite It Using Modern Language, Platform
3. Use Agile and TDD It!
4. Just make it all cloud microservices!

We must reduce our technical debt! It will take years and lots of money but we will attack and refactor, rewrite legacy mountain

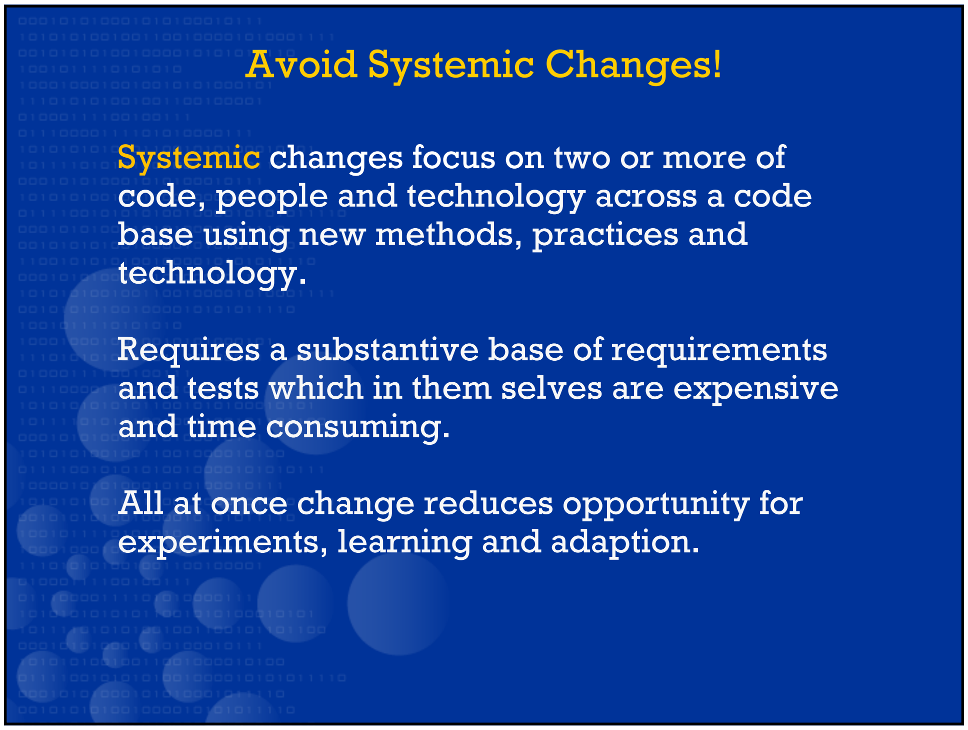
Technical Debt as Defined versus as Used

Ward's definition …

"The whole debt metaphor, is the ability to pay back debt, and make the debt metaphor work for your advantage depends upon your writing code that is clean enough to be able to refactor as you come to understand your problem." i.e. write a little code, refactor a little code.

Refactoring – A Disciplined Practice fo Small Changes – Equivalence Preserving made easy by Tests

- Most Refactorings are disguised Rewrites!
- Existing tool and practices don't offer any serious support for large code bases

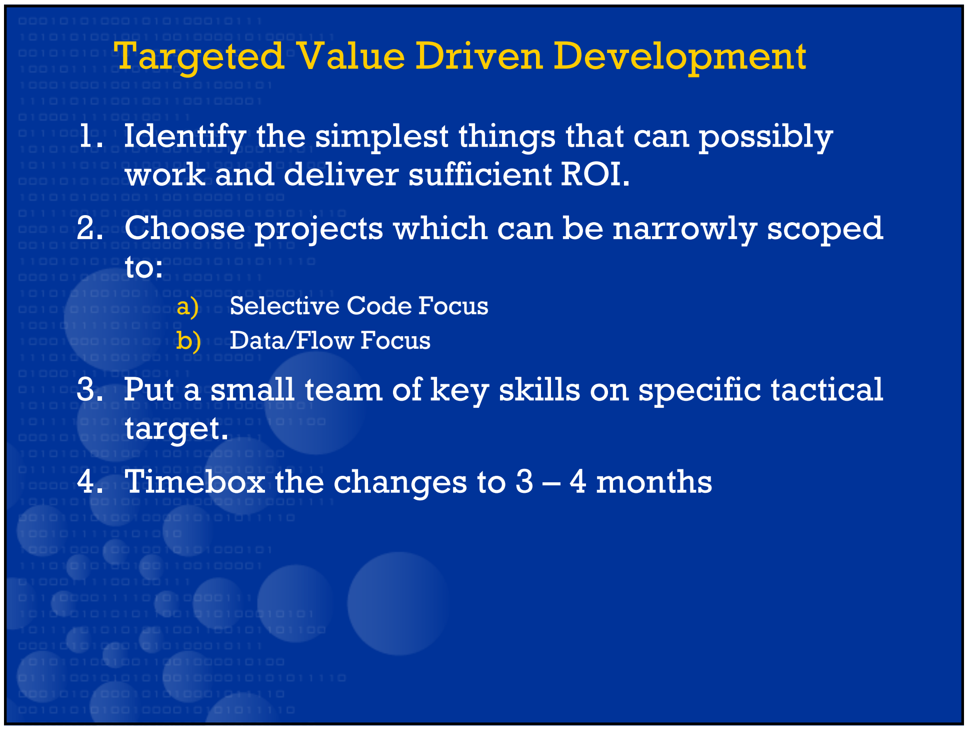


Avoid Systemic Changes!

Systemic changes focus on two or more of code, people and technology across a code base using new methods, practices and technology.

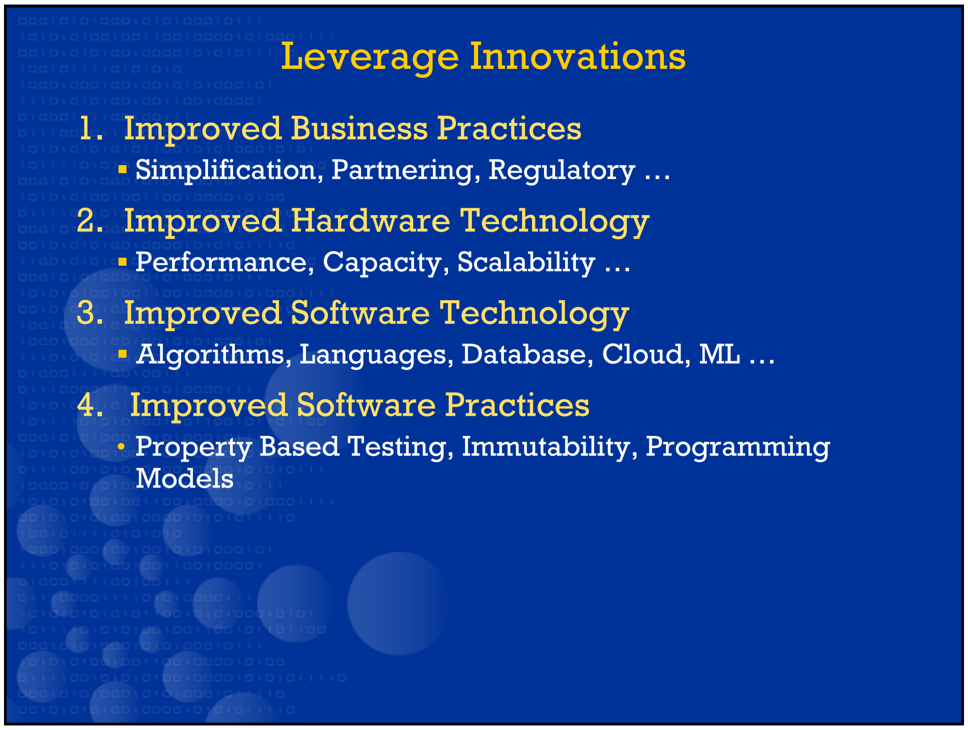
Requires a substantive base of requirements and tests which in them selves are expensive and time consuming.

All at once change reduces opportunity for experiments, learning and adaption.



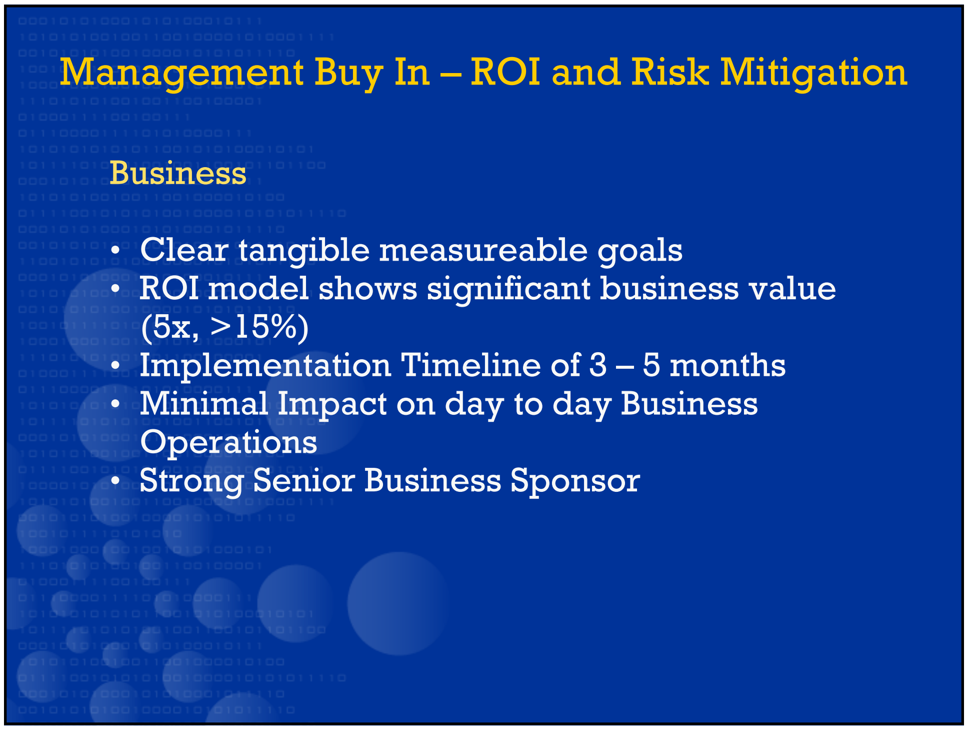
Targeted Value Driven Development

1. Identify the simplest things that can possibly work and deliver sufficient ROI.
2. Choose projects which can be narrowly scoped to:
    a) Selective Code Focus
    b) Data/Flow Focus
3. Put a small team of key skills on specific tactical target.
4. Timebox the changes to 3 – 4 months



Leverage Innovations

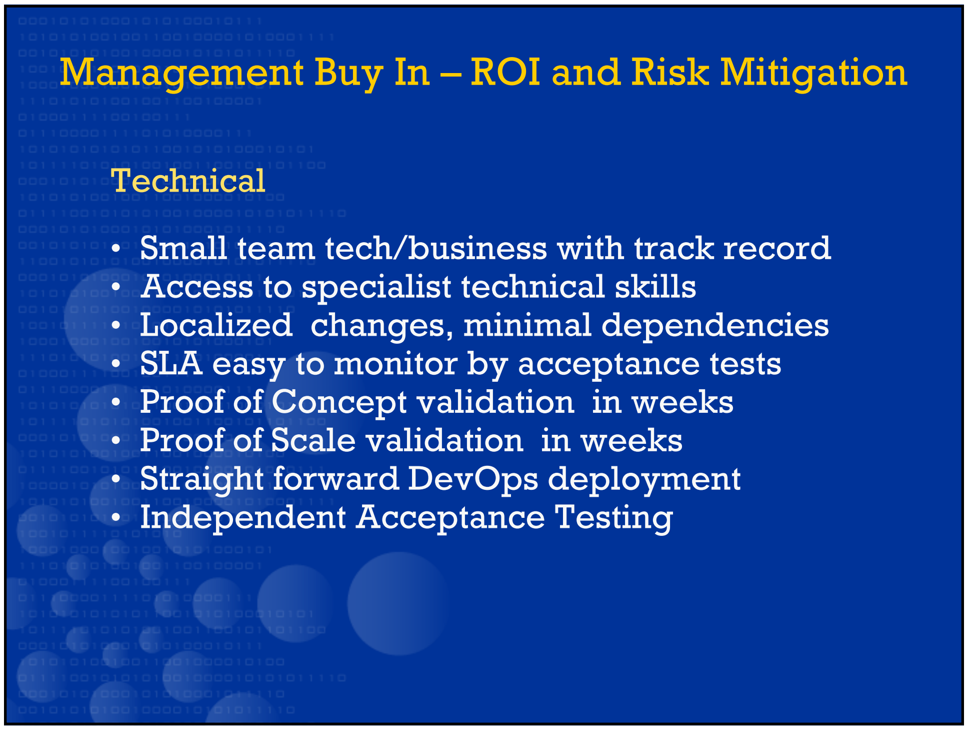
1. Improved Business Practices
   - Simplification, Partnering, Regulatory …
2. Improved Hardware Technology
   - Performance, Capacity, Scalability …
3. Improved Software Technology
   - Algorithms, Languages, Database, Cloud, ML …
4. Improved Software Practices
   - Property Based Testing, Immutability, Programming Models



Management Buy In – ROI and Risk Mitigation

Business

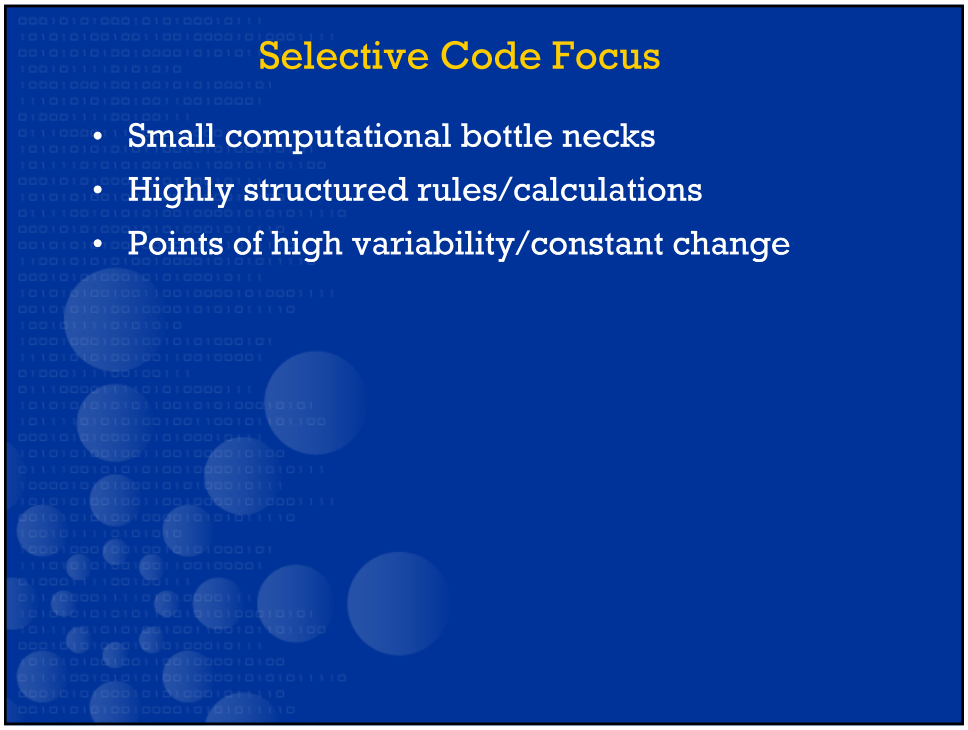
- Clear tangible measureable goals
- ROI model shows significant business value (5x, >15%)
- Implementation Timeline of 3 – 5 months
- Minimal Impact on day to day Business Operations
- Strong Senior Business Sponsor



Management Buy In – ROI and Risk Mitigation

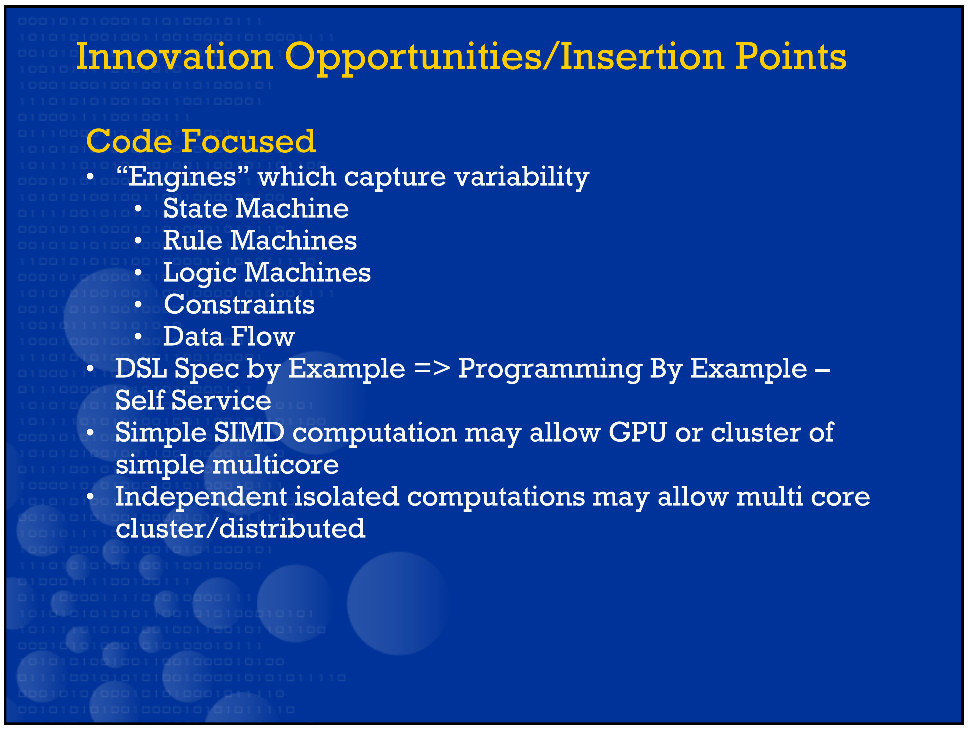
Technical

- Small team tech/business with track record
- Access to specialist technical skills
- Localized changes, minimal dependencies
- SLA easy to monitor by acceptance tests
- Proof of Concept validation in weeks
- Proof of Scale validation in weeks
- Straight forward DevOps deployment
- Independent Acceptance Testing



Selective Code Focus

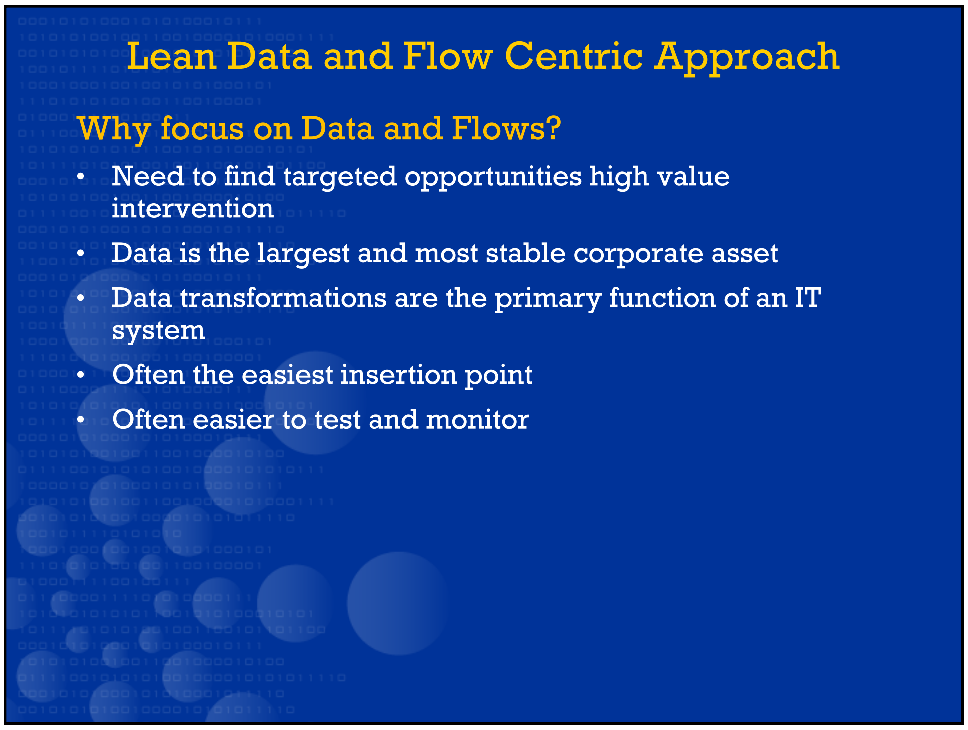
- Small computational bottle necks
- Highly structured rules/calculations
- Points of high variability/constant change



Innovation Opportunities/Insertion Points

Code Focused

- “Engines” which capture variability
  - 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



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
- Often easier to test and monitor

Innovation Opportunities/Insertion Points

Data and Flow Focused

- Database, File, Log, Messaging, Serialization,
- Shared Memory, Disk, Network…
- Functional Transformers - ETL Interface; Map Reduce; GPU …
- Sync Replicate

Legacy Innovation Patterns

1. Make it Table/Data Driven to accommodate variability

Code => Tables | Rules |Constraints …

Case Study – Global HR Provider

Case Study - Commercial Insurance Provider

Legacy Innovation Patterns

1. Make it Table/Data Driven to accommodate variability

   Code => Tables | Rules | Constraints …

2. Make it look like the Web

   Integration => HTTP/ATOM/REST vs APIs

Case Study - Process Control

Legacy Innovation Patterns

1. Make it Table/Data Driven to accommodate variability

   Code => Tables | Rules | Constraints …

2. Make it look like the Web

   Integration => HTTP/ATOM/REST vs APIs

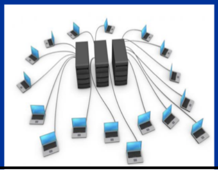
3. You really only need ONE API

   Make it look like a SELECT

   a. Make it look like a data base => ODBC for x
   b. Make it look like a collection => LINQ/Rx
   c. Make it look like federated query => GraphQL

Case Study - Legacy Manufacturing Application

– Providing a Uniform API for Apps



Legacy Innovation Patterns

1. Make it Table/Data Driven to accommodate variability

   Code => Tables | Rules |Constraints …

2. Make it look like the Web

   Integration => HTTP/ATOM/REST vs APIs

3. You really only need ONE API

   Make it look like a SELECT

   a. Make it look like a data base => ODBC for x
   b. Make it look like a collection => LINQ/Rx
   c. Make it look like federated query => GraphQL

4. Apply a Functional Transformer

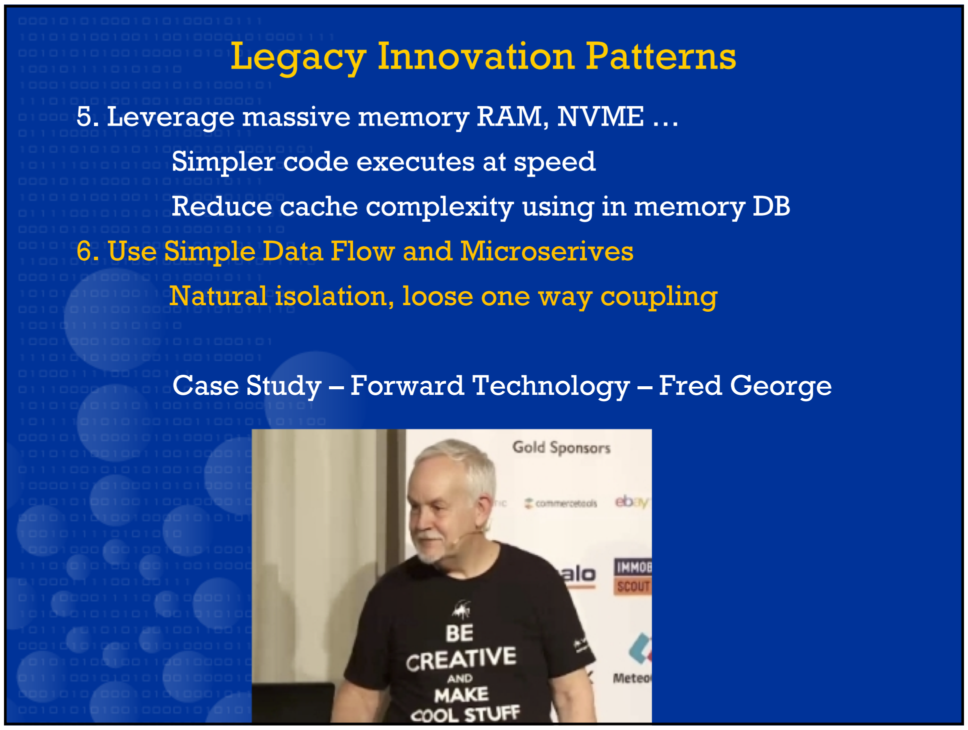
Legacy Innovation Patterns

5. Leverage massive memory RAM, NVME …

   Simpler code executes at speed

   Reduce cache complexity using in memory DB

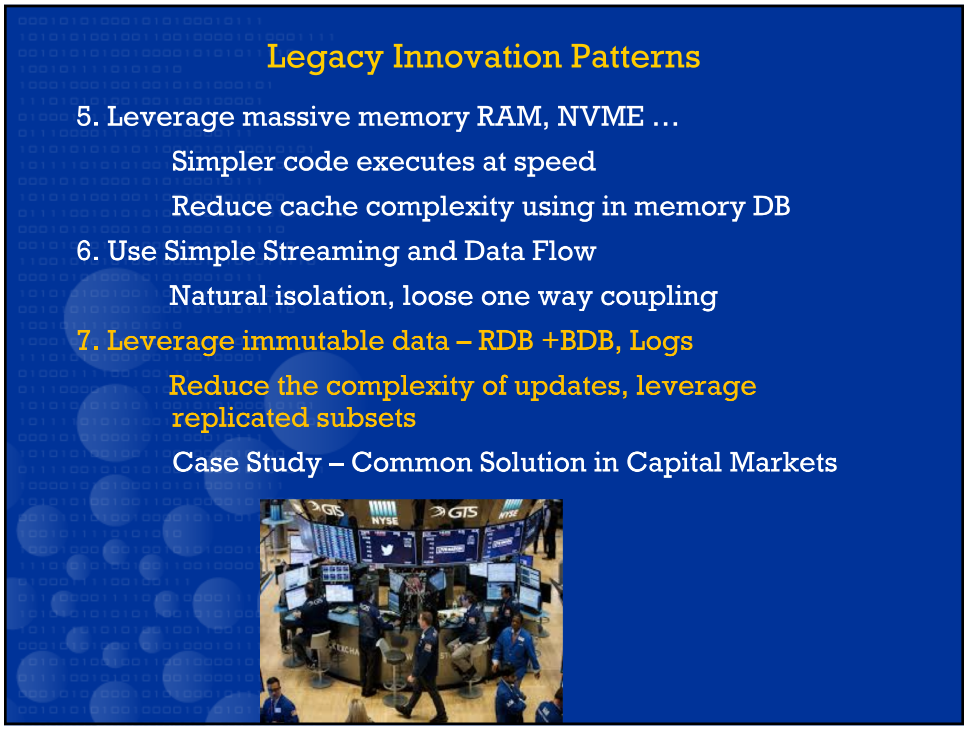
Caste Study – Cyber Analytics



Legacy Innovation Patterns

5. Leverage massive memory RAM, NVME …
   - Simpler code executes at speed
   - Reduce cache complexity using in memory DB
6. Use Simple Data Flow and Microserives
   - Natural isolation, loose one way coupling

Case Study – Forward Technology – Fred George



Legacy Innovation Patterns

5. Leverage massive memory RAM, NVME …
   - Simpler code executes at speed
   - Reduce cache complexity using in memory DB
6. Use Simple Streaming and Data Flow
   - Natural isolation, loose one way coupling
7. Leverage immutable data – RDB +BDB, Logs
   - Reduce the complexity of updates, leverage replicated subsets

Case Study – Common Solution in Capital Markets

Legacy Innovation Patterns

5. Leverage massive memory RAM, NVME …
   - Simpler code executes at speed
   - Reduce cache complexity using in memory DB
6. Use Simple Streaming and Data Flow
   - Natural isolation, loose one way coupling
7. Leverage immutable data – RDB +BDB, Logs
   - Reduce the complexity of updates, leverage replicated subsets
8. TDD and More – Property Based Testing;
   - Independent Implementation of Validation…

Case Study - Database Restructuring

Embrace your Legacy
and
Innovate in It!

Thanks!