IBM Software Group

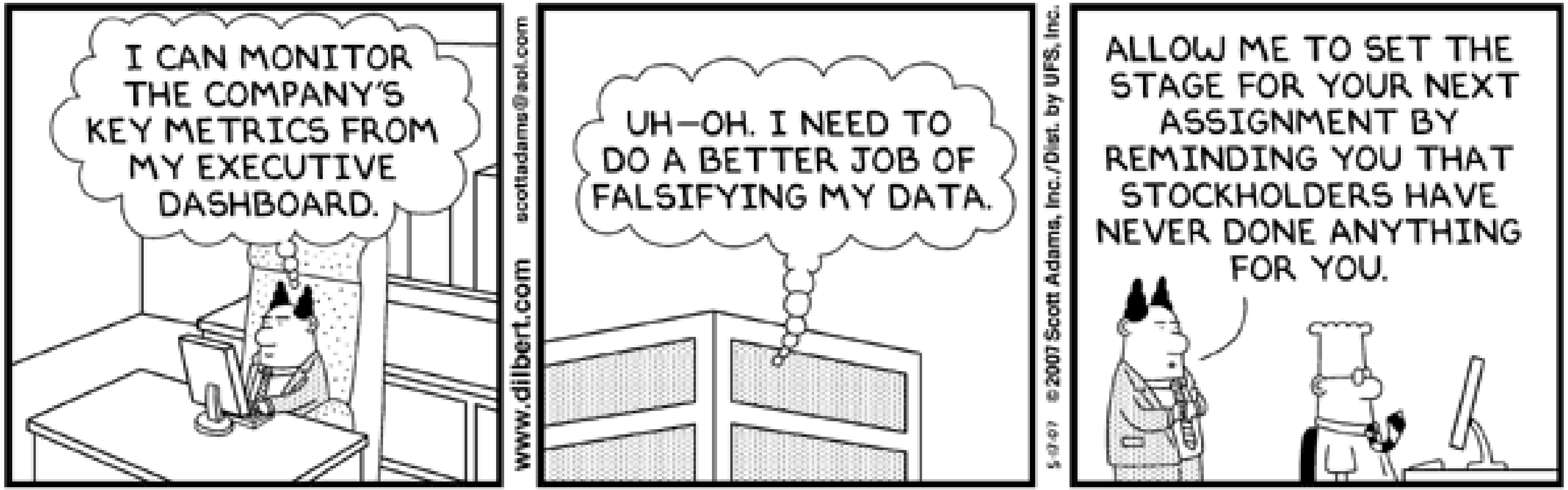
Measurements Drive Performance

Achieving confidence in an uncertain world

Steven Trevellion
North East Europe Rational Services Manager

Go to IBM

Do metrics drive performance?


I CAN MONITOR THE COMPANY'S KEY METRICS FROM MY EXECUTIVE DASHBOARD.
www.dilbert.com
scottadams@aol.com
UH—OH. I NEED TO DO A BETTER JOB OF FALSIFYING MY DATA.
© 2007 Scott Adams, Inc./Dist. by UFS, Inc.
ALLOW ME TO SET THE STAGE FOR YOUR NEXT ASSIGNMENT BY REMINDING YOU THAT STOCKHOLDERS HAVE NEVER DONE ANYTHING FOR YOU.

© Scott Adams, Inc./Dist. by UFS, Inc.

They clearly drive behaviour

The ***right ones*** help drive desired performance

A bit of bookwork: Part I

Some interesting facts

- The last 20 years
  - Cost estimation is not getting much better
  - US: Productivity for traditional systems comparatively flat
  - US: Quality levels have only improved slightly
- Productivity / Quality
  - Best is 3 times above average
  - Worst are 50% worse
- Historical Data
  - Leading edge companies have 10 times more data available for estimation / planning
- Sociological v. Economic
  - 20% of organisations have productivity / quality metrics based on functionality

Sources: COCOMO, Capers Jones: Applied Software Measurement

A bit of bookwork: Part II

“The good, the bad …..”

- Good metrics
  - Provide insight and therefore the ability to change for the better
  - Are a minimum set to support business need
  - Are well defined and understood by those affected
  - Are easy to collect accurately (preferably automated)
  - Are visible

- Questionable metrics (not the above!)
  - Crush individual/team spirit, innovation, desire to do a good job, ….
  - Are used for harassment or humiliation, …..
  - Appear to measure something tangible (i.e. # of pages in a specification) but do not actually provide a measure of progress or value

A bit of bookwork: Part III

“The past or the future”

- Leading Indicators
  - Give an indication of the future

- Lagging Indicators
  - Tell you about the past

Aircraft stall warning

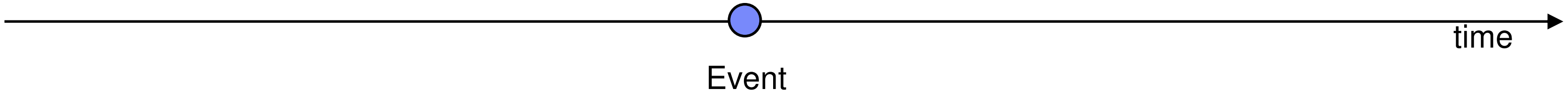
Defect trend e.g. bad = not reducing

Cost projections e.g. bad = increase month on month

Aircraft “black-box” data

Escaped defects (e.g. in the user base)

Cost at completion (e.g. overspent)



Maybe corrective action can be taken

Too late to do anything this time
➔ lessons learnt for next time

What is UK industry actually measuring?

Metrics our consultants find in use

- **Project Schedule Variance**. Time recording measurements (e.g. planned effort vs actual, effort against specific project tasks or phases), productivity
- **Project Cost Variance**. Project budgets/costs
- **Quality**. Defect density, defects by phase, defects by priority.
- **Requirements**. Number, types, traced/untraced, churn...
- **Development**. Source lines of code, Use Case points (rare), McCabe's cyclomatic complexity
- **Testing**. Passed tests, failed tests, test coverage.

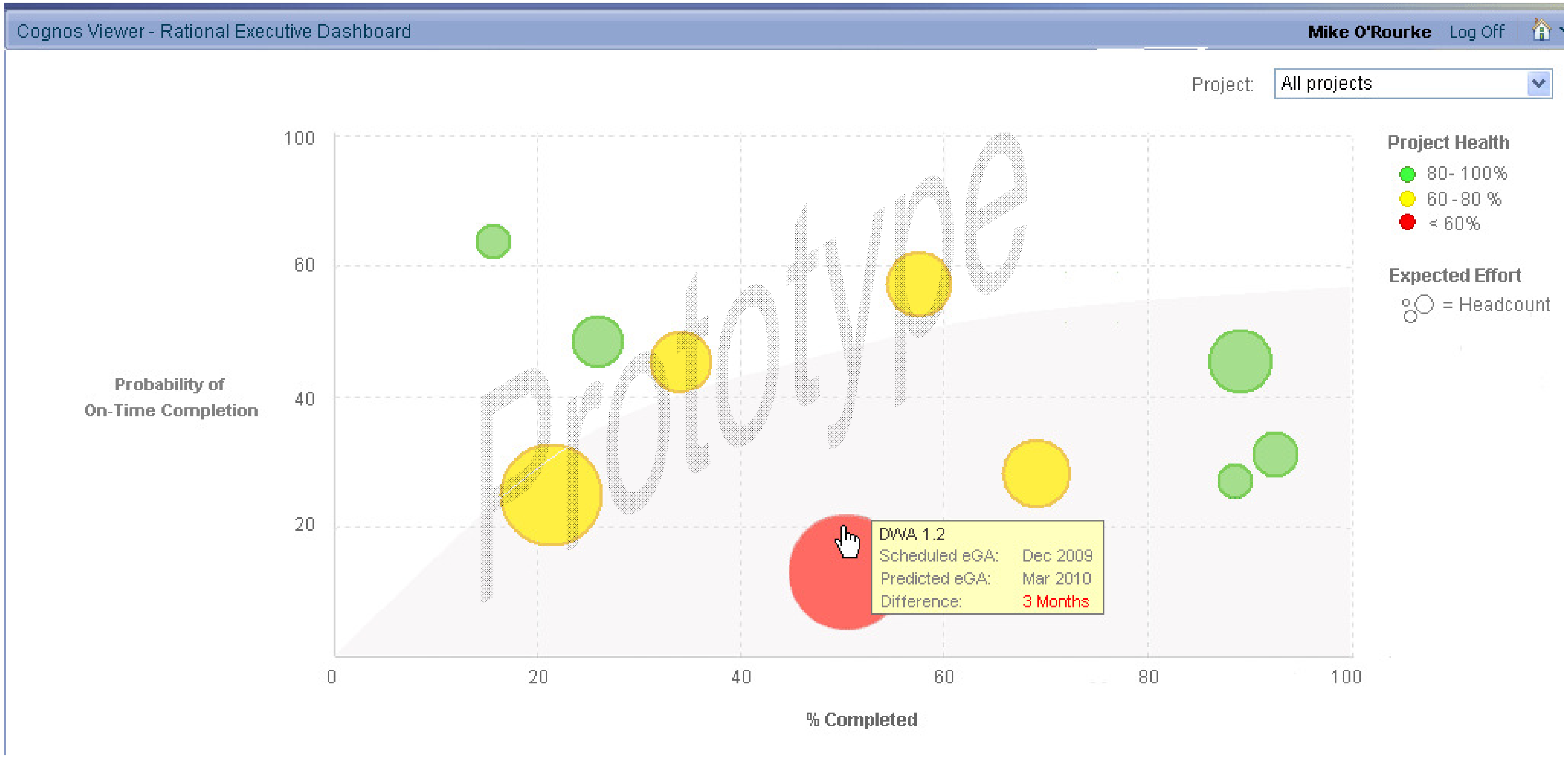
What is industry actually measuring?

UK High integrity systems developer

- 10+ years ago
  - Compelling need: avoid historical programme cost and schedule slippage
  - Solution: Iterative, risk driven software development
  - Primary measure: system usage scenarios (specified, implemented, tested)
  - Standard measures: SLOC productivity, defect trends, PM metrics
- Today
  - Engineering Dashboard
    - Operational objectives (flown down from BU to EU e.g. tender response time)
    - Financial (EBIT, overhead control)
    - Training
  - IS Tooling (Tool licence usage, denials)
- Future
  - MoD mandating Earned Value reporting

What is industry actually measuring?

Rational product development



Cognos Viewer - Rational Executive Dashboard

Mike O'Rourke Log Off

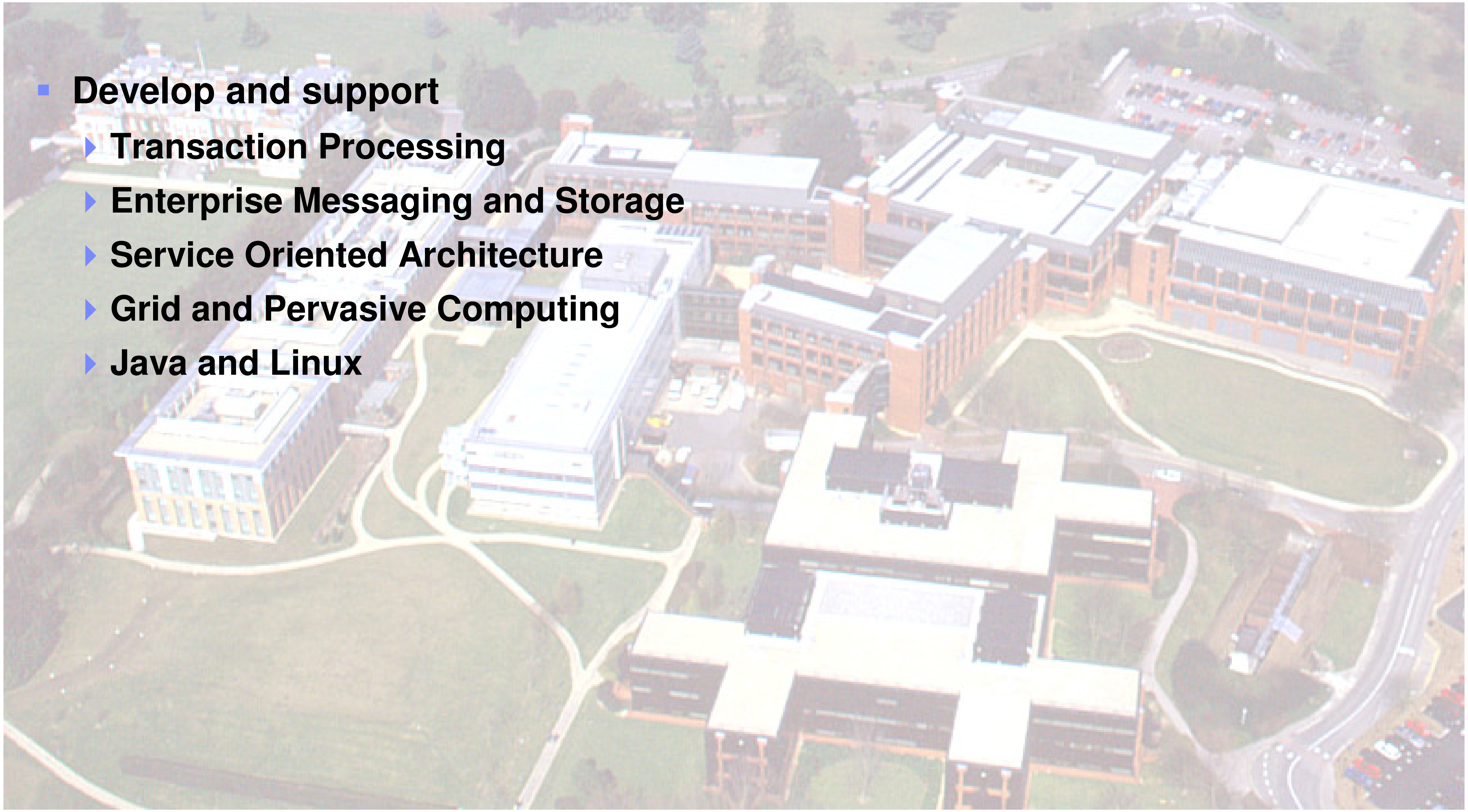
Project: All projects

DWA 1.2
Scheduled eGA: Dec 2009
Predicted eGA: Mar 2010
Difference: 3 Months

Prototype dashboard illustrating business level view of measures

What is industry actually measuring?

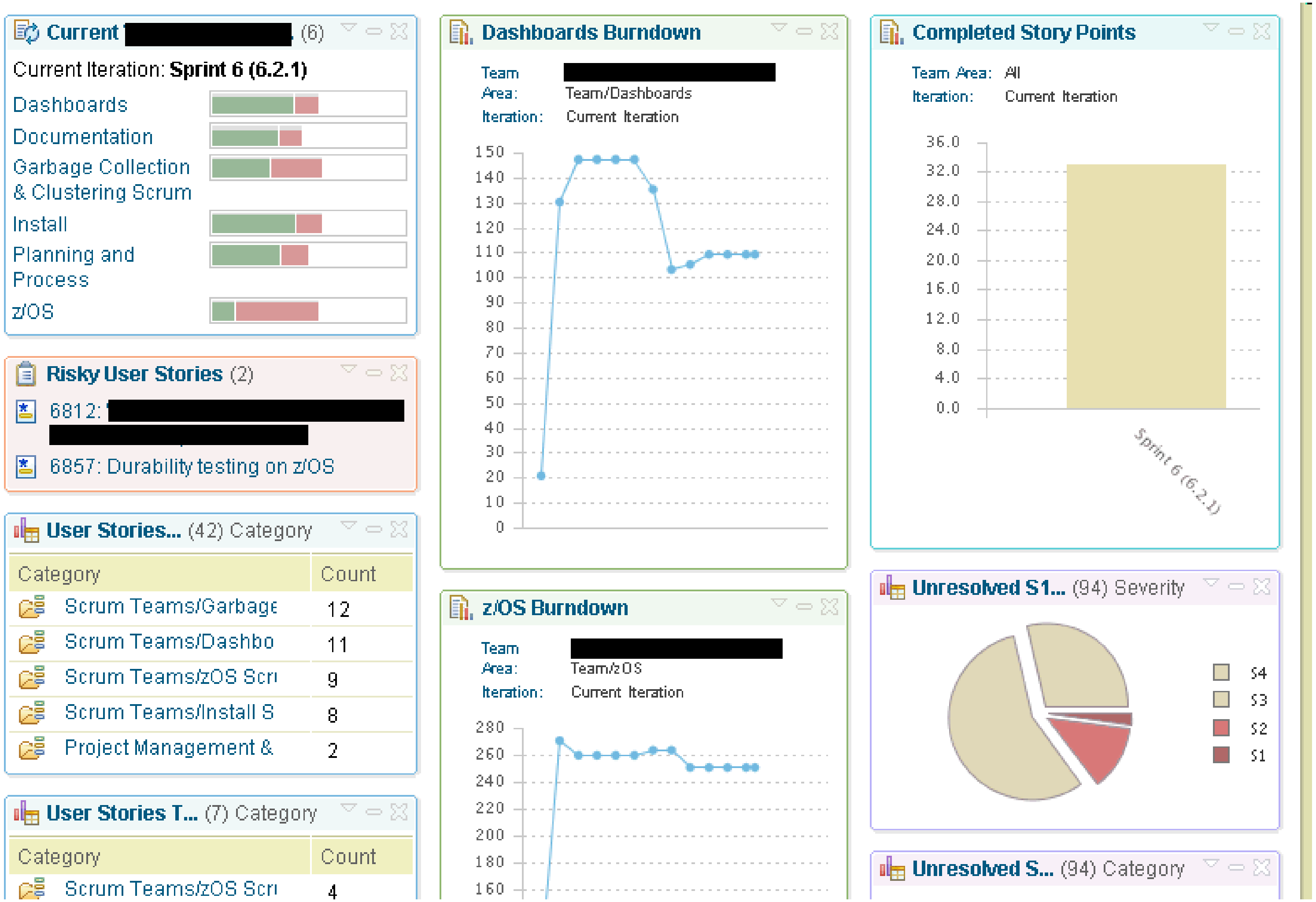
IBM Hursley Development Labs



- Develop and support
  - Transaction Processing
  - Enterprise Messaging and Storage
  - Service Oriented Architecture
  - Grid and Pervasive Computing
  - Java and Linux

IBM Hursley Development Lab

Example: Websphere Product X Development



Current [redacted] (6)

Current Iteration: **Sprint 6 (6.2.1)**

- Dashboards
- Documentation
- Garbage Collection & Clustering Scrum
- Install
- Planning and Process
- z/OS

Risky User Stories (2)

- 6812: [redacted]
- 6857: Durability testing on z/OS

User Stories... (42) Category

Category	Count
Scrum Teams/Garbage	12
Scrum Teams/Dashbo	11
Scrum Teams/zOS Scri	9
Scrum Teams/Install S	8
Project Management &	2

User Stories T... (7) Category

Category	Count
Scrum Teams/zOS Scri	4

Dashboards Burndown

Team Area: [redacted] Team/Dashboards

Iteration: Current Iteration

z/OS Burndown

Team Area: [redacted] Team/zOS

Iteration: Current Iteration

Completed Story Points

Team Area: All

Iteration: Current Iteration

Unresolved S1... (94) Severity

Unresolved S... (94) Category

IBM Hursley Development Lab

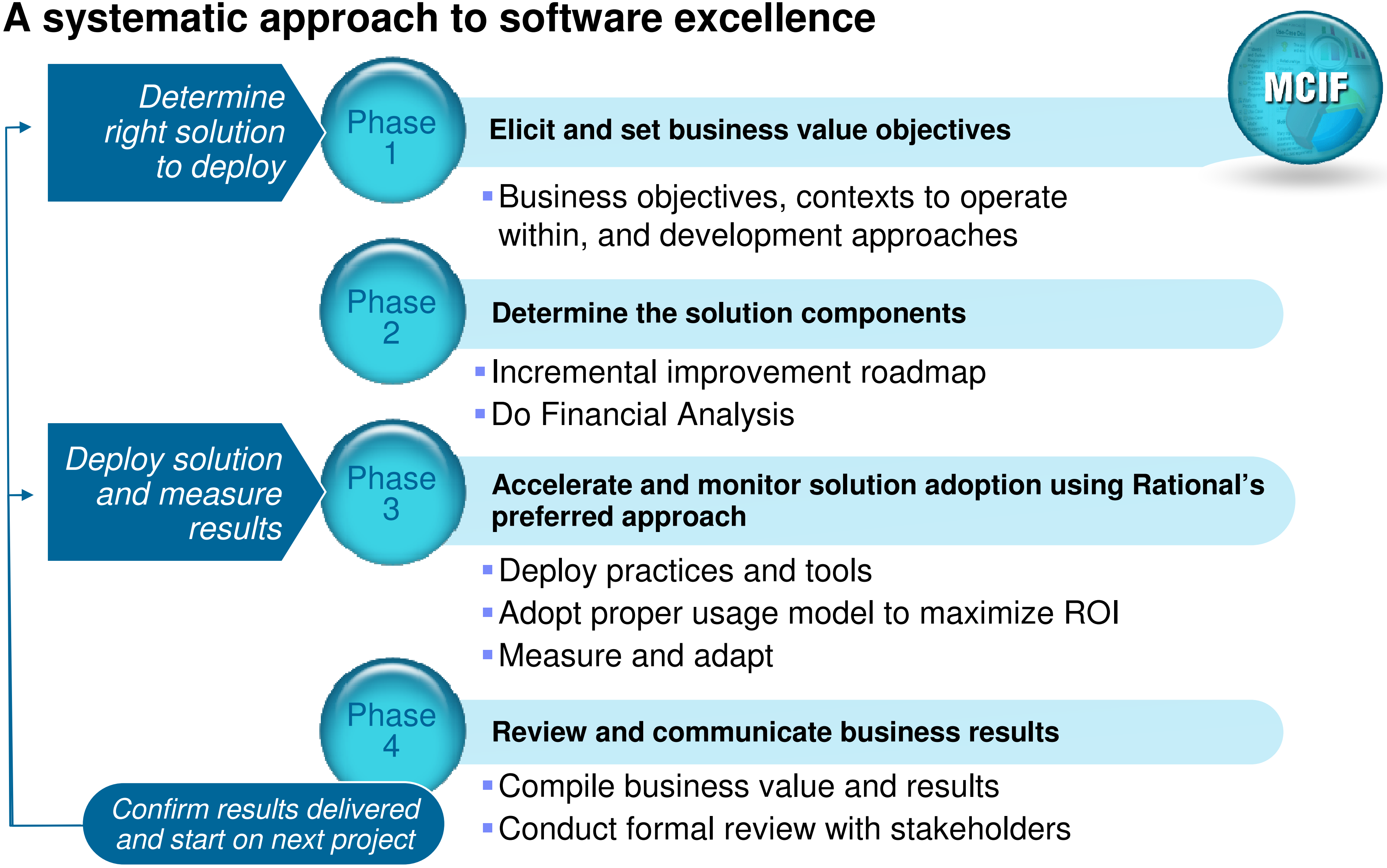
More Example Measures

- Platform Management
  - Test suite pass rate by build (identifies problem areas)
  - Test suite pass rate over time (Identifies test reliability)
  - Build v. Defect 3D Trend Chart (builds on X, time on Z, pass rate on Y identifies where problems were introduced)

- Tivoli (CQ-ALM)
  - Tasks per phase (workload management)
  - Activities ready for building (“fix” management)

Measured Capability Improvement Framework (MCIF)

A systematic approach to software excellence



Determine right solution to deploy

Phase 1 — **Elicit and set business value objectives**

- Business objectives, contexts to operate within, and development approaches

Phase 2 — **Determine the solution components**

- Incremental improvement roadmap
- Do Financial Analysis

Deploy solution and measure results

Phase 3 — **Accelerate and monitor solution adoption using Rational's preferred approach**

- Deploy practices and tools
- Adopt proper usage model to maximize ROI
- Measure and adapt

Phase 4 — **Review and communicate business results**

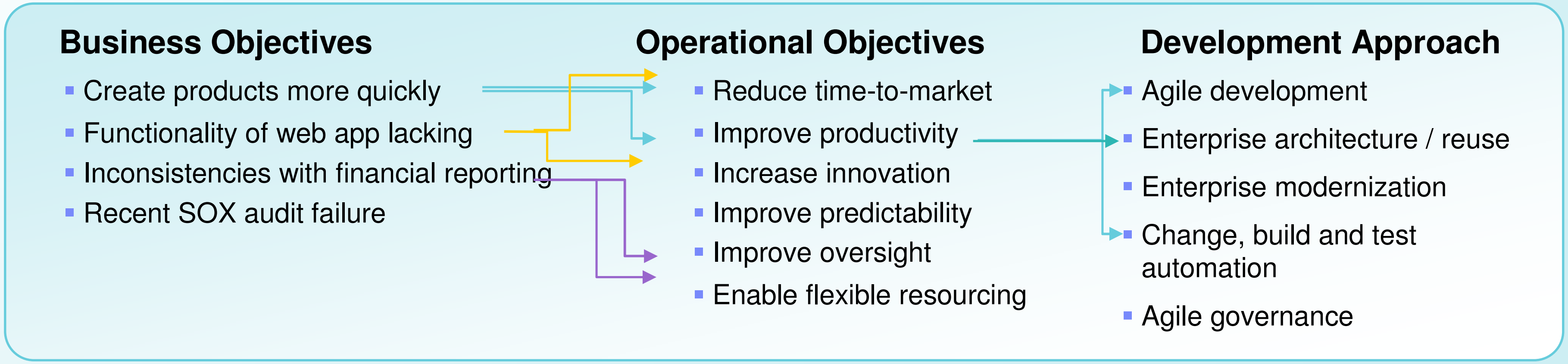
- Compile business value and results
- Conduct formal review with stakeholders

Confirm results delivered and start on next project

Phase 1: Elicit and set business value objectives

- Business and IT
  - Define business objectives
  - Map business objectives to operational objectives
  - Define development approach

Know what you want to improve and why



Business Objectives

- Create products more quickly
- Functionality of web app lacking
- Inconsistencies with financial reporting
- Recent SOX audit failure

Operational Objectives

- Reduce time-to-market
- Improve productivity
- Increase innovation
- Improve predictability
- Improve oversight
- Enable flexible resourcing

Development Approach

- Agile development
- Enterprise architecture / reuse
- Enterprise modernization
- Change, build and test automation
- Agile governance

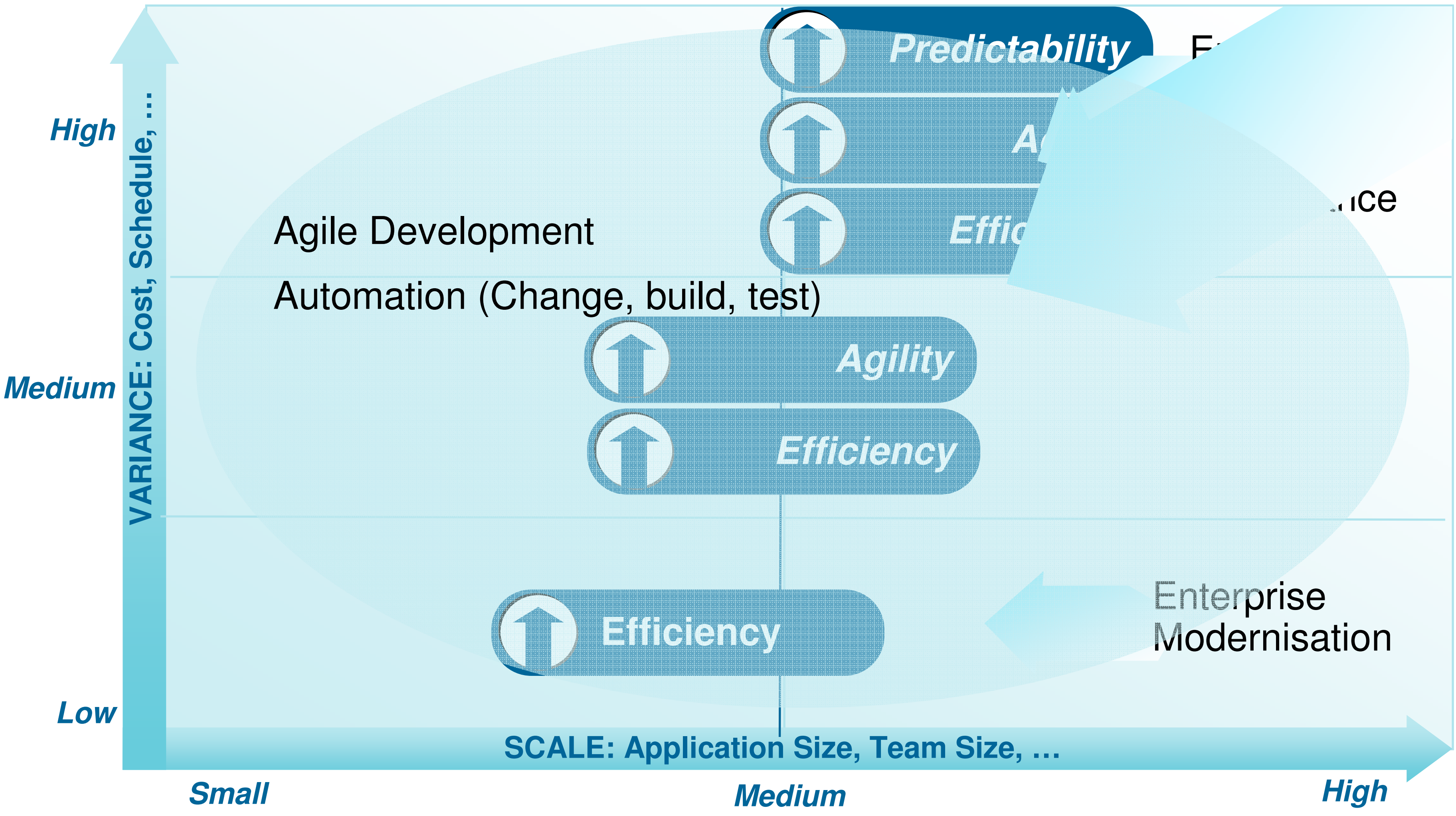
Establishing context for setting business objectives

Primary dimensions

- Scale / complexity
  - Application size, team size
  - Risk of loss of life or money, compliance need
- Variance
  - Technical, legal, and organisational risk
  - Organisational ability to deliver

Sources: COCOMO – 22 parameters

Determining improvement areas and approach



Agile Development

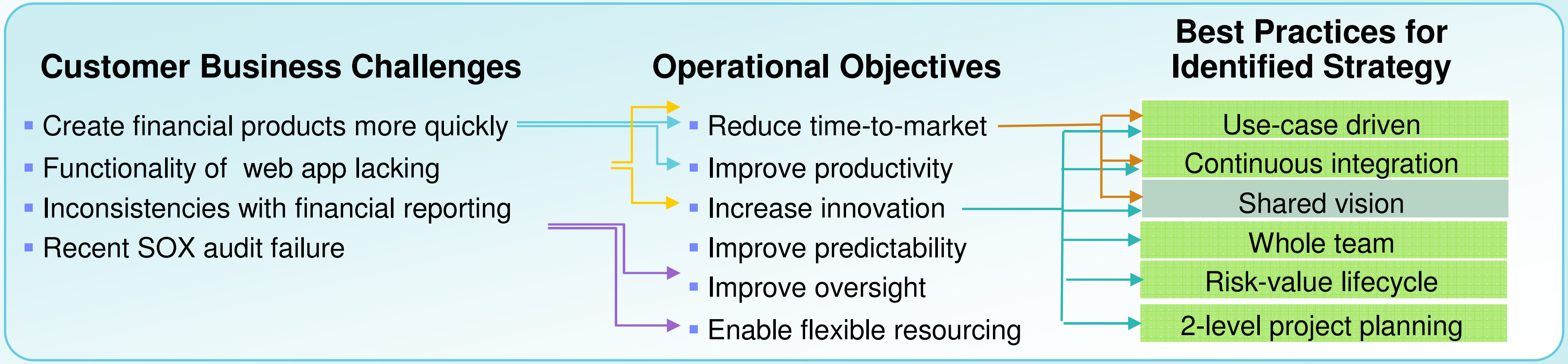
Automation (Change, build, test)

Enterprise Modernisation

Phase 2: Determine solution components

- Business and IT
  - Baseline “As-Is” and define “To-Be” model
  - Define solution roadmap
  - Do financial analysis; business case
  - Define measurement system

Be able to justify and know how the improvement will yield %x, $y in n months for an investment of $z



Customer Business Challenges

- Create financial products more quickly
- Functionality of web app lacking
- Inconsistencies with financial reporting
- Recent SOX audit failure

Operational Objectives

- Reduce time-to-market
- Improve productivity
- Increase innovation
- Improve predictability
- Improve oversight
- Enable flexible resourcing

Best Practices for Identified Strategy

Use-case driven

Continuous integration

Shared vision

Whole team

Risk-value lifecycle

2-level project planning

Practice-based approach

- Addresses one aspect of the software lifecycle
  - e.g., continuous integration, use-case driven development, etc.
- Can be incrementally and independently adopted
- Can be mapped to operational objectives and development pain points
- Adoption can be measured

Results:

✓ Avoids “too much process”

✓ Faster and more predictable results

Context determines preferred practices

Different practices apply to different contexts

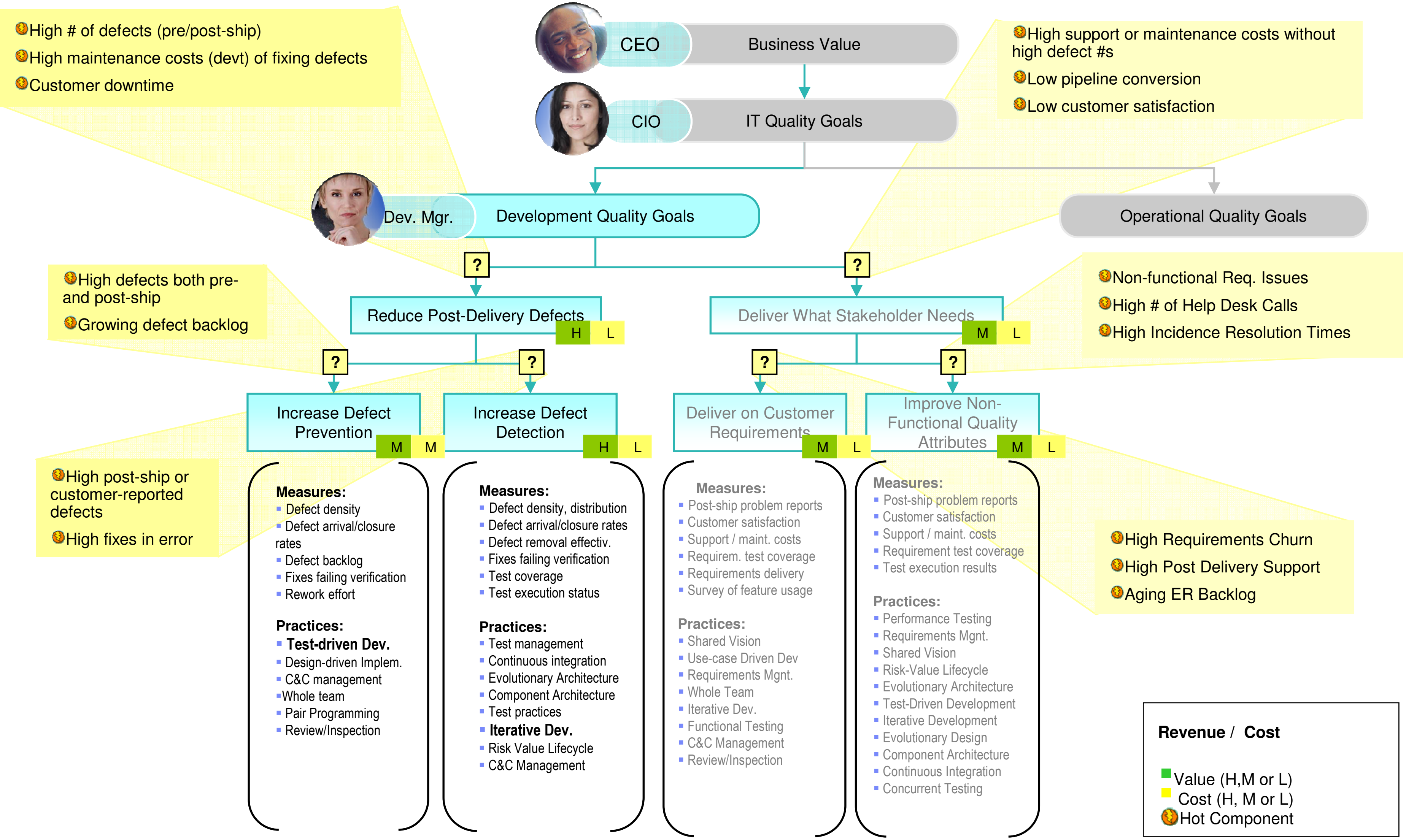
VARIANCE: Cost, Schedule, …

Predictability	High	Requirements elicitation Evolutionary architecture	Risk-value lifecycle Requirements Management Modeling practices
Agility	Medium	Iterative development 2-level planning Shared vision	Architecture management Requirements elicitation Iterative development 2-level planning
Efficiency	Low	Test management Release management Continuous integration	Test management Release management Continuous integration

SCALE: Application Size, Team Size, …

Small — Medium — High

Example mapping



●High # of defects (pre/post-ship)
●High maintenance costs (devt) of fixing defects
●Customer downtime

CEO — Business Value

CIO — IT Quality Goals

●High support or maintenance costs without high defect #s
●Low pipeline conversion
●Low customer satisfaction

Dev. Mgr. — Development Quality Goals

Operational Quality Goals

●High defects both pre- and post-ship
●Growing defect backlog

? Reduce Post-Delivery Defects H L

? Deliver What Stakeholder Needs M L

●Non-functional Req. Issues
●High # of Help Desk Calls
●High Incidence Resolution Times

? Increase Defect Prevention M M

? Increase Defect Detection H L

? Deliver on Customer Requirements M L

? Improve Non-Functional Quality Attributes M L

●High post-ship or customer-reported defects
●High fixes in error

Measures:
- Defect density
- Defect arrival/closure rates
- Defect backlog
- Fixes failing verification
- Rework effort

Practices:
- **Test-driven Dev.**
- Design-driven Implem.
- C&C management
- Whole team
- Pair Programming
- Review/Inspection

Measures:
- Defect density, distribution
- Defect arrival/closure rates
- Defect removal effectiv.
- Fixes failing verification
- Test coverage
- Test execution status

Practices:
- Test management
- Continuous integration
- Evolutionary Architecture
- Component Architecture
- Test practices
- **Iterative Dev.**
- Risk Value Lifecycle
- C&C Management

Measures:
- Post-ship problem reports
- Customer satisfaction
- Support / maint. costs
- Requirem. test coverage
- Requirements delivery
- Survey of feature usage

Practices:
- Shared Vision
- Use-case Driven Dev
- Requirements Mgnt.
- Whole Team
- Iterative Dev.
- Functional Testing
- C&C Management
- Review/Inspection

Measures:
- Post-ship problem reports
- Customer satisfaction
- Support / maint. costs
- Requirement test coverage
- Test execution results

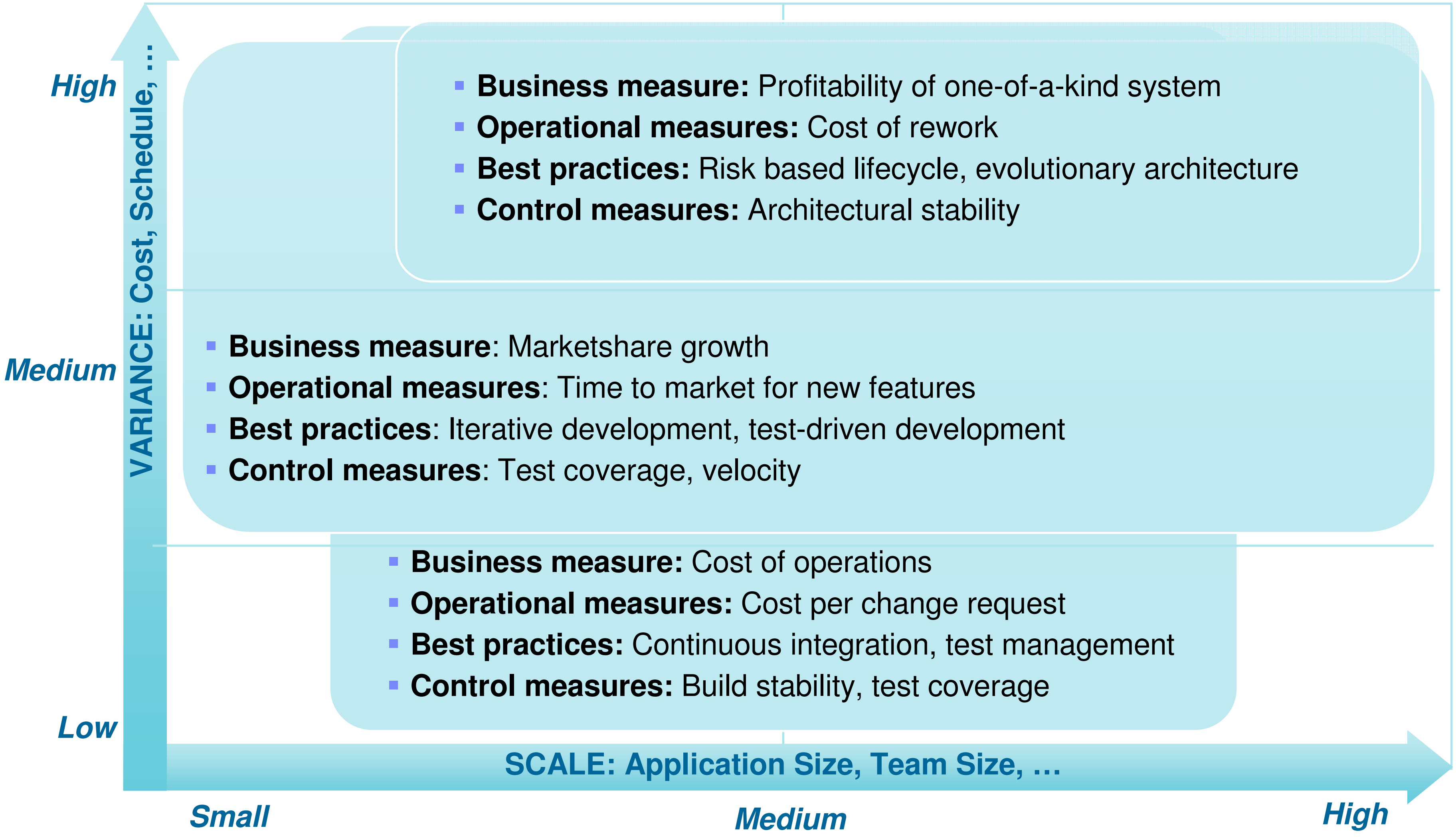
Practices:
- Performance Testing
- Requirements Mgnt.
- Shared Vision
- Risk-Value Lifecycle
- Evolutionary Architecture
- Test-Driven Development
- Iterative Development
- Evolutionary Design
- Component Architecture
- Continuous Integration
- Concurrent Testing

●High Requirements Churn
●High Post Delivery Support
●Aging ER Backlog

Revenue / Cost

🟩Value (H,M or L)
🟨Cost (H, M or L)
●Hot Component

Tailor measurements to context



High

- **Business measure:** Profitability of one-of-a-kind system
- **Operational measures:** Cost of rework
- **Best practices:** Risk based lifecycle, evolutionary architecture
- **Control measures:** Architectural stability

Medium

- **Business measure**: Marketshare growth
- **Operational measures**: Time to market for new features
- **Best practices**: Iterative development, test-driven development
- **Control measures**: Test coverage, velocity

Low

- **Business measure:** Cost of operations
- **Operational measures:** Cost per change request
- **Best practices:** Continuous integration, test management
- **Control measures:** Build stability, test coverage

VARIANCE: Cost, Schedule, …

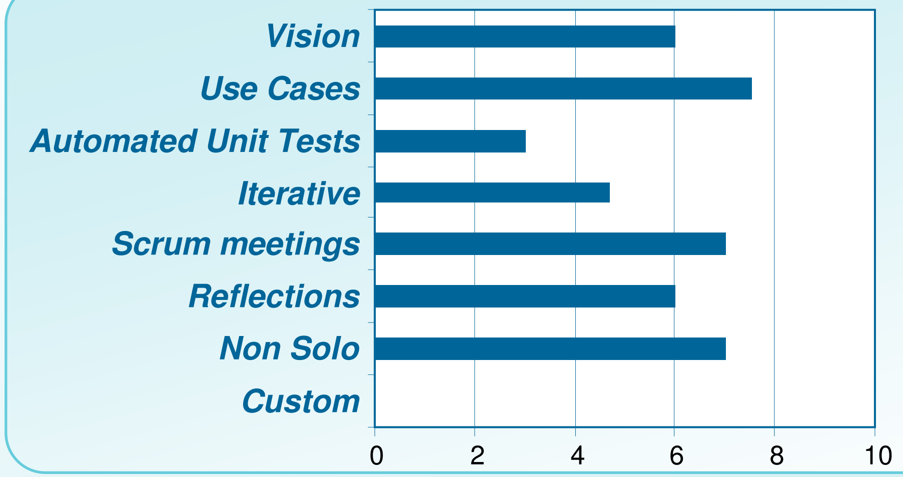
SCALE: Application Size, Team Size, …

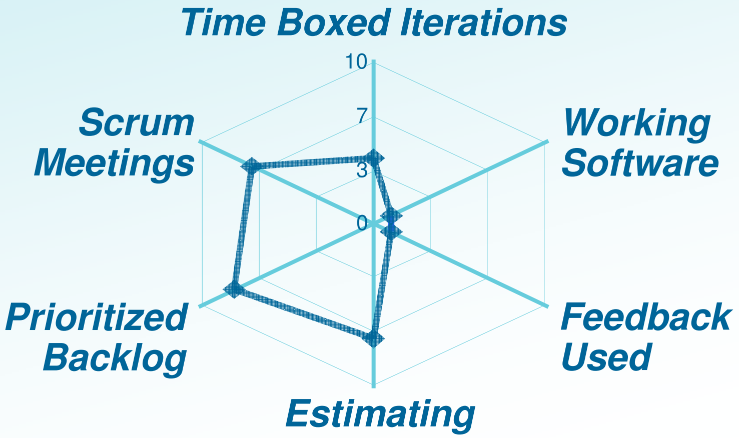
Small Medium High

Phase 3: Accelerate and monitor adoption

- Business and IT
  - Implement roadmap (incremental adoption of practices)
  - Monitor results and steer adoption
  - Showcase improvements

Demonstrate tangible improvements in incremental steps

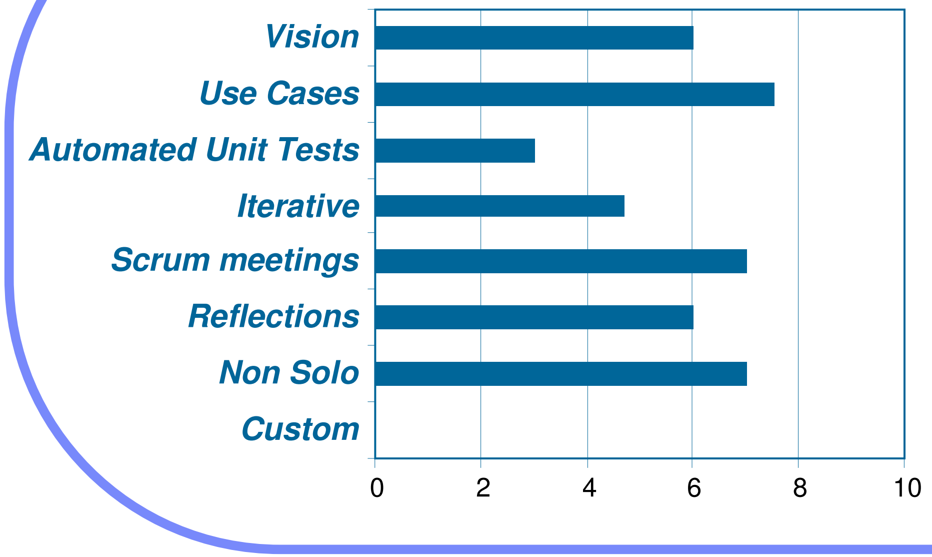




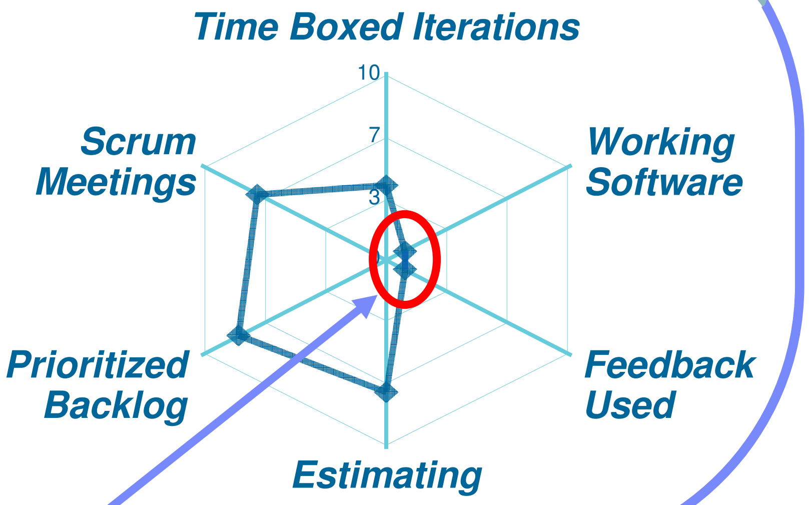
Monitor Adoption

Example: Are we iterative?

The Big Picture



Are we iterative?

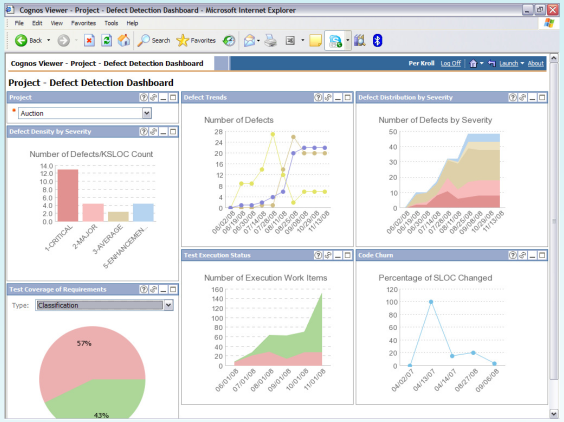


No – iterative needs demonstrable results – there is little working software or feedback

Phase 4: Review and communicate business results

- Business and IT
  - Review and steer as required
  - Publish results

Communicate and steer on-going improvements



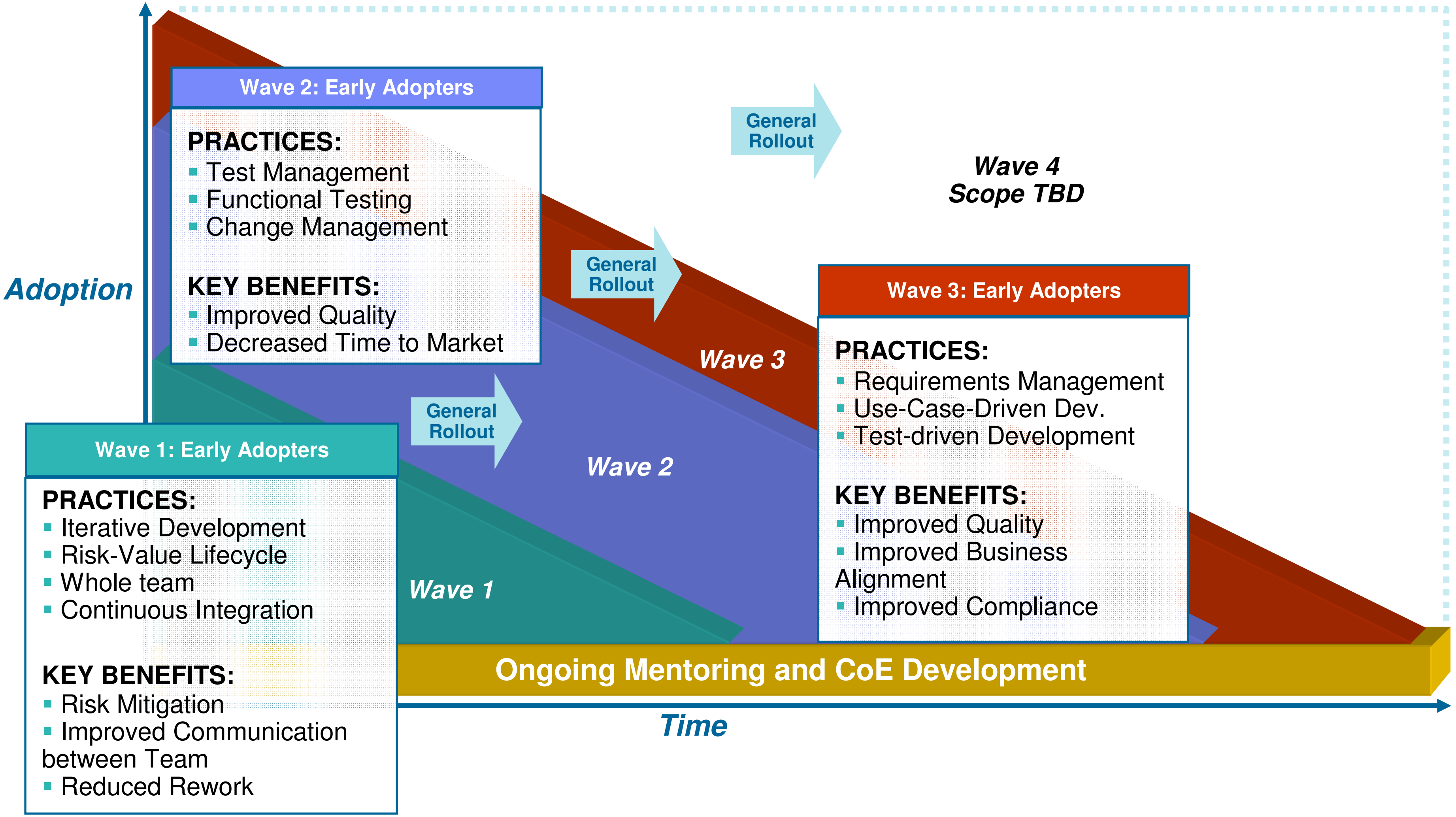
Cognos Viewer - Project - Defect Detection Dashboard - Microsoft Internet Explorer

Project - Defect Detection Dashboard

Project: Auction

Test Coverage of Requirements — Type: Classification

Example of incremental adoption



Wave 1: Early Adopters

PRACTICES:
- Iterative Development
- Risk-Value Lifecycle
- Whole team
- Continuous Integration

KEY BENEFITS:
- Risk Mitigation
- Improved Communication between Team
- Reduced Rework

Wave 2: Early Adopters

PRACTICES:
- Test Management
- Functional Testing
- Change Management

KEY BENEFITS:
- Improved Quality
- Decreased Time to Market

Wave 3: Early Adopters

PRACTICES:
- Requirements Management
- Use-Case-Driven Dev.
- Test-driven Development

KEY BENEFITS:
- Improved Quality
- Improved Business Alignment
- Improved Compliance

Case study: Improving IT efficiency

Company profile

- Global telecom equipment manufacturer
  - 2500 engineers in years 1-2, ramping up to 6,000 in 3-5
  - Average project size: 20 people
  - Average project duration: 6 months

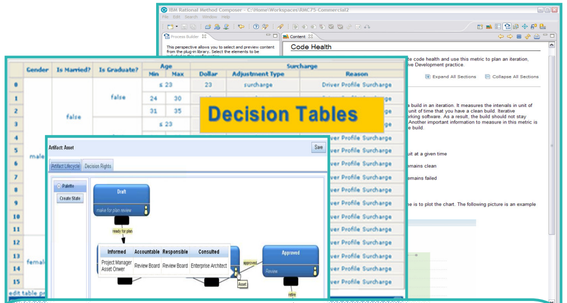
Assessment findings

- Address IT as a “cost center:” Scope creep on "customer specials" eroding profitability
  - Manage requirements scope and improve analysis and cost estimates rigor
- 80% of projects are maintenance
  - Reduce defect rates; improve test efficiency, invest in innovation
- Multiple build processes across acquired companies
  - Streamline processes to reduce build failures

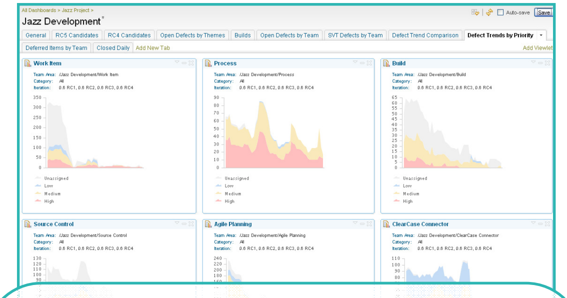
	Short term	*Short term*	*Mid term*	*Long Term*
Benefit	**Reduce probability of failed builds by 50-75%**	**Reduce defect rates by 5%**	**Improve test team efficiency by 8%**	**Defect rates lowered 5% more, test costs lowered 30% more**
Strategy	Streamline build management processes across acquired companies; automate build management	Improve/integrate defect &requirements management Establish defect management between defects, requirements & test	Automate manual tests, reduce 40-60% overhead. Test teams create/tear down test infrastructures	Add requirements definition (e.g., storyboards, use cases, performance requirements) Automate test lab management
Timeframe	6 months	4 months	3 months	12 months

Jazz platform direction

Operationalise capability improvement

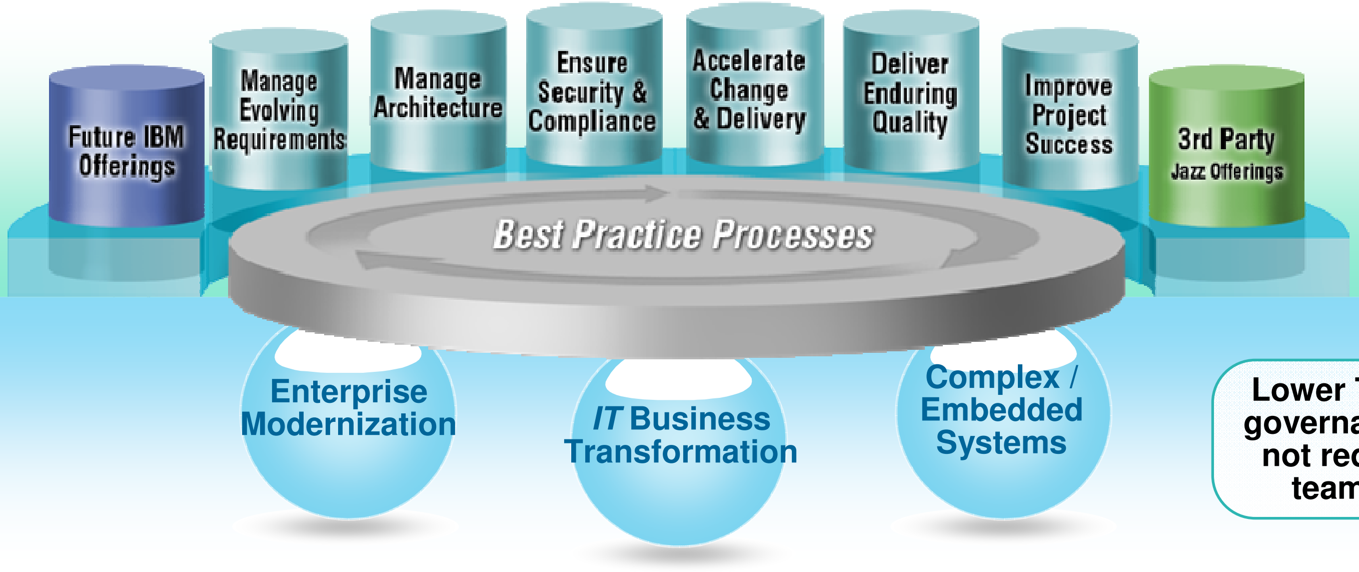


Define Business and Operational Objectives
Define Practices, RACI Specifications and Policies



Operationalize processes, accountability and policies across your organization

Measure business and operational results
Measure practices adoption



Lower TCO through governance platform not requiring each team to retool.

Learn more at:

- IBM Rational software
- Rational launch announcements
- Rational Software Delivery Platform
- Accelerate change & delivery
- Deliver enduring quality
- Enable enterprise modernization
- Ensure Web security & compliance
- Improve project success
- Manage architecture
- Manage evolving requirements
- Small & midsized business
- Targeted solutions
- Rational trial downloads
- developerWorks Rational
- Leading Innovation
- IBM Rational TV
- IBM Business Partners
- IBM Rational Case Studies