



IBM Rational Software Conference 2009
As Real as It Gets!



Achieving Agility at Scale Improving Software Economics

Speaker's Name:
Speaker's Title:

Rational. software

Executive Summit Exec02

Agenda

- **Transitioning to Agile Software Delivery**
 - ▶ Economic governance
 - ▶ Steering and managing uncertainty

- **Metrics and Measures**
 - ▶ Instrumentation for gaining control, improving efficiency and optimizing value

- **Improving Software Economics**
 - ▶ A framework for improvement priorities

Software development obsolesced by software delivery

Software Development

Distinct development phase

Distinct handoff to maintenance

Requirements-design-code-test sequence

Phase and role specific tools

Collocated teams

Standard engineering governance

Engineering practitioner led

Software Delivery

Continuously evolving systems

No distinct boundary between development and maintenance

Sequence of released capabilities with ever increasing value

Common platform of integrated process / tooling

Distributed, web based collaboration

Economic governance tailored to risk / reward profiles

Business value and outcome led

Critical culture shifts in improving software economics

Conventional Governance

Activity-based management

Mature processes, PMI/PMBOK
Plan in detail, then track variances

Adversarial relationships

Paper exchange, speculation

Requirements first

Assumes certainty in desired product
Avoid change

Early false precision

“More detail = higher quality”

Apply too much or too little process

Process is primary, blind adherence

Agile Governance

Results-based management

More art than engineering
Plan/steer/plan/steer...

Honest collaborative communication

Progressions/digressions, facts

Architecture (*risk mitigation*) first

Admits uncertainties
Manage change

Evolving artifacts

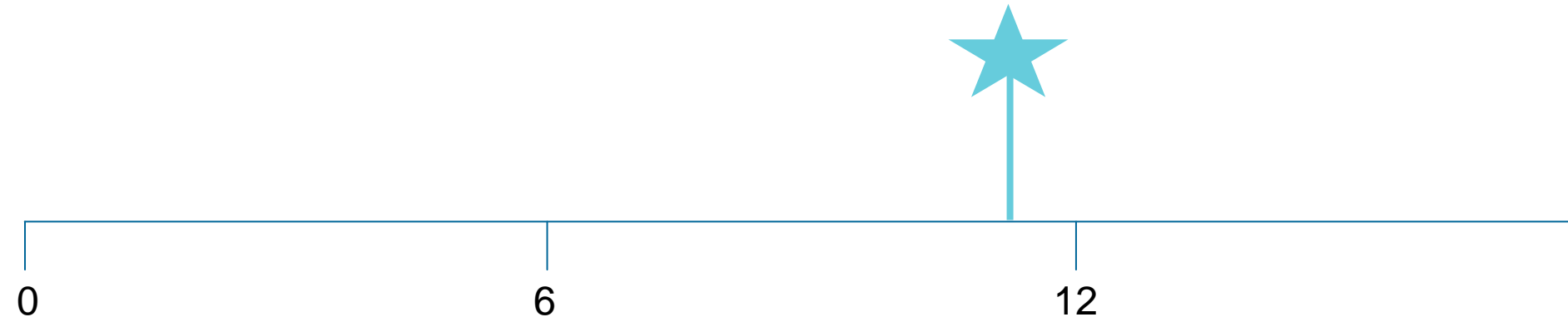
Scope (Problem specs)
Design (Solution specs)
Constraints (Planning specs)

Right-size the process

Desired results drive process
Manage variances

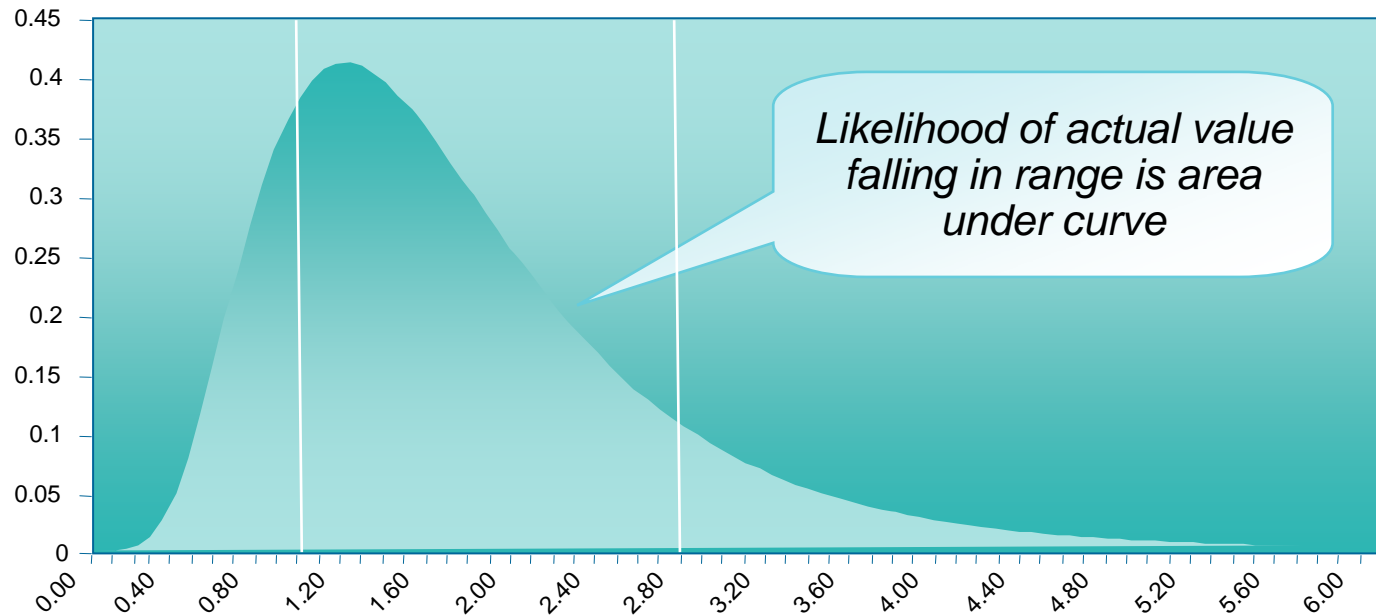
Schedule risk: Imagine you have 12 months to deliver a business critical system

- Your estimators tell you it will be done in 11 months
- What do you do with the information?
 - ▶ Rest easy, believing there is no risk?



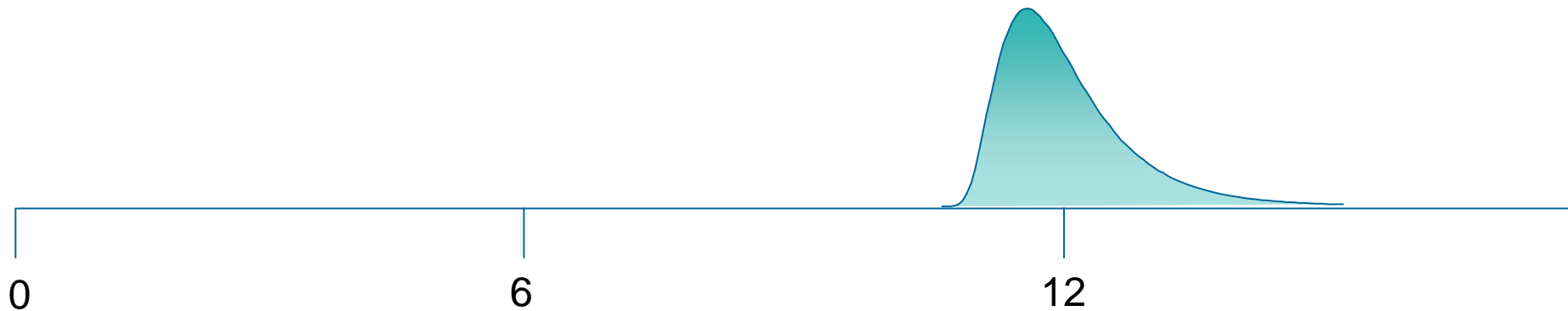
Maybe you realize that program parameters (cost, schedule, effort, quality, ...) are random variables

- Area under curve describes probability of measurement falling in range



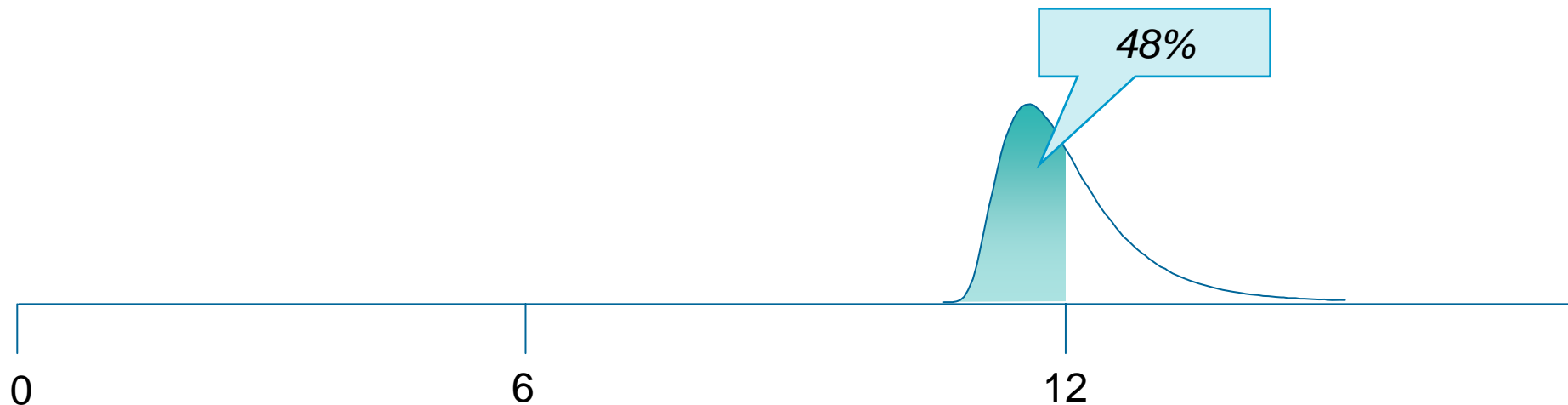
Imagine you have 12 months to deliver a business critical systems

- So you ask for the distribution and discover there is some uncertainty



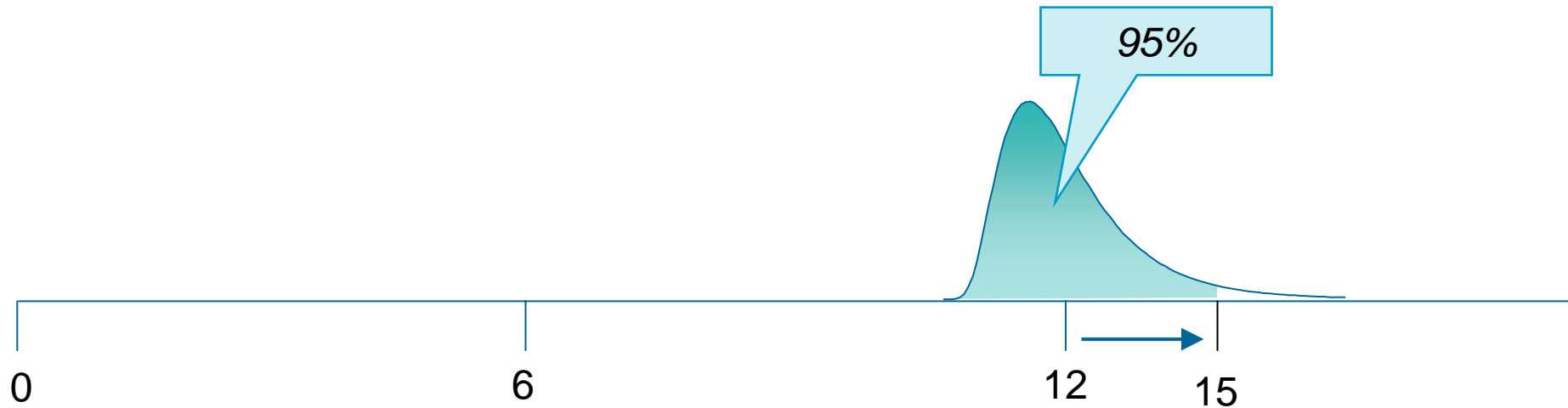
Imagine you have 12 months to deliver a business critical systems

- In fact there is less than 50% chance of making the date



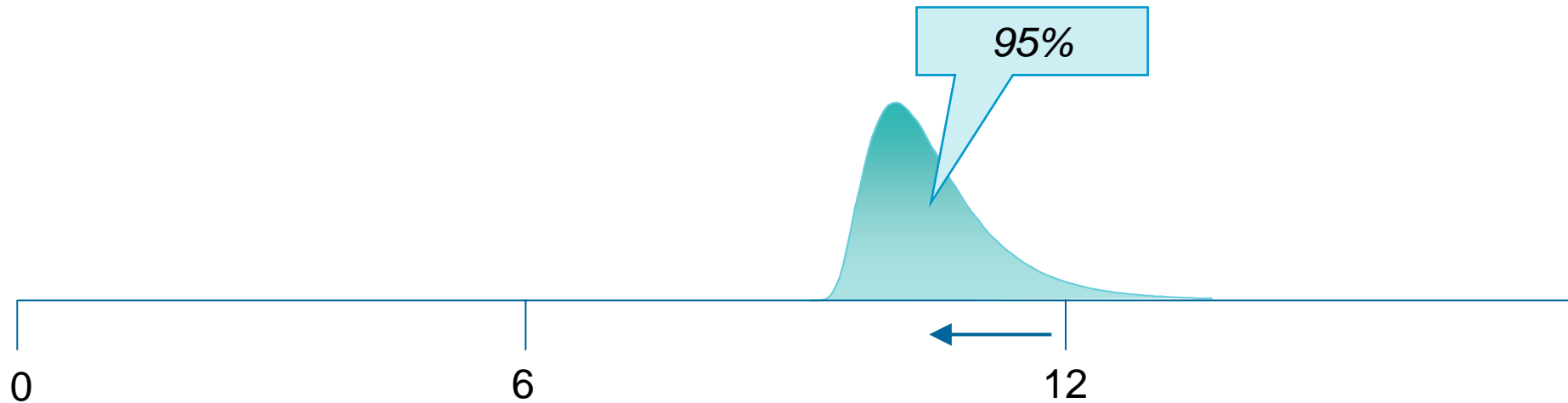
Then what?

- Move out the date to improve likelihood of shipping?



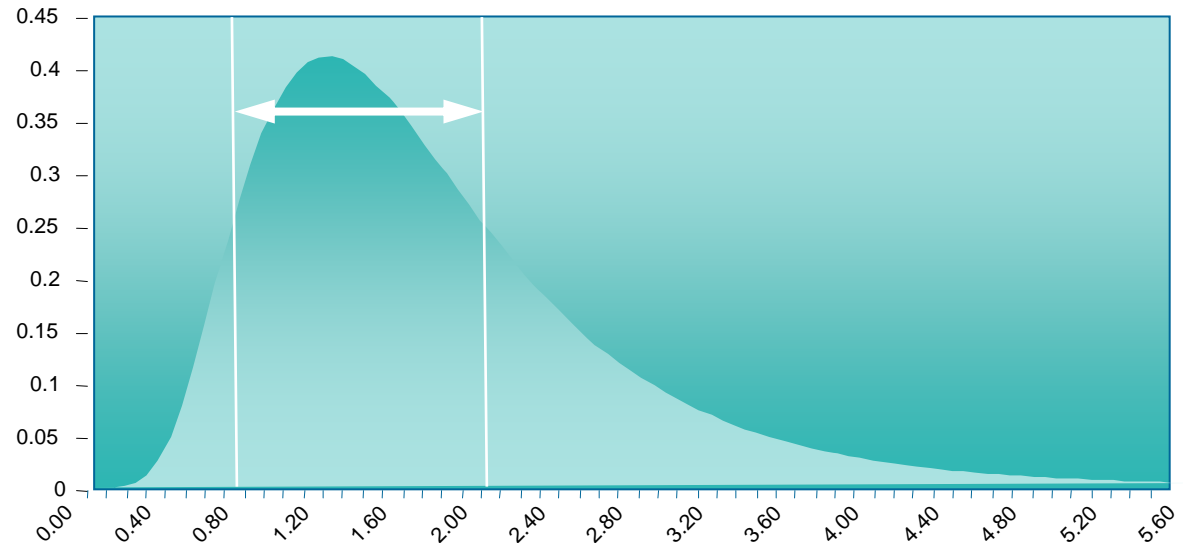
Then what?

- Or move in the estimate by sacrificing quality or content?



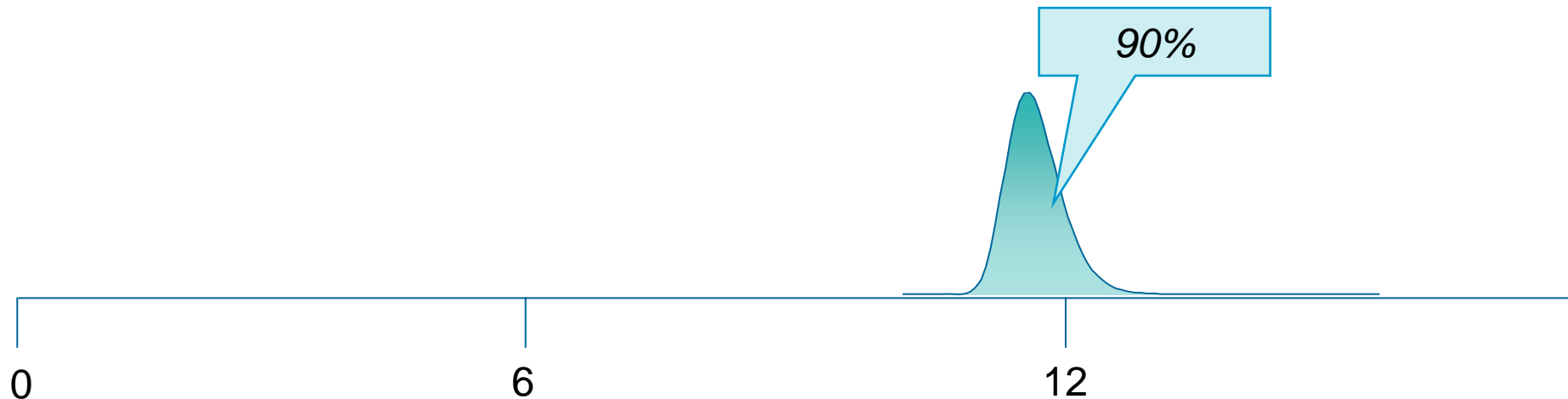
Managing variances in scope, solution, plans: The real key to improving software economics

- Sources of uncertainty and variance
 - ▶ Lack of knowledge
 - ▶ Lack of confidence
 - ▶ Lack of agreement
- Reduction of variance reflects
 - ▶ Increased predictability of outcome
 - ▶ Increased knowledge about
 - Client needs
 - Technology capability
 - Team capability
 - ▶ Good decisions



Then what?

- Determine the source of the variance
- Over the project lifecycle, reduce the variance to improve likelihood of shipping



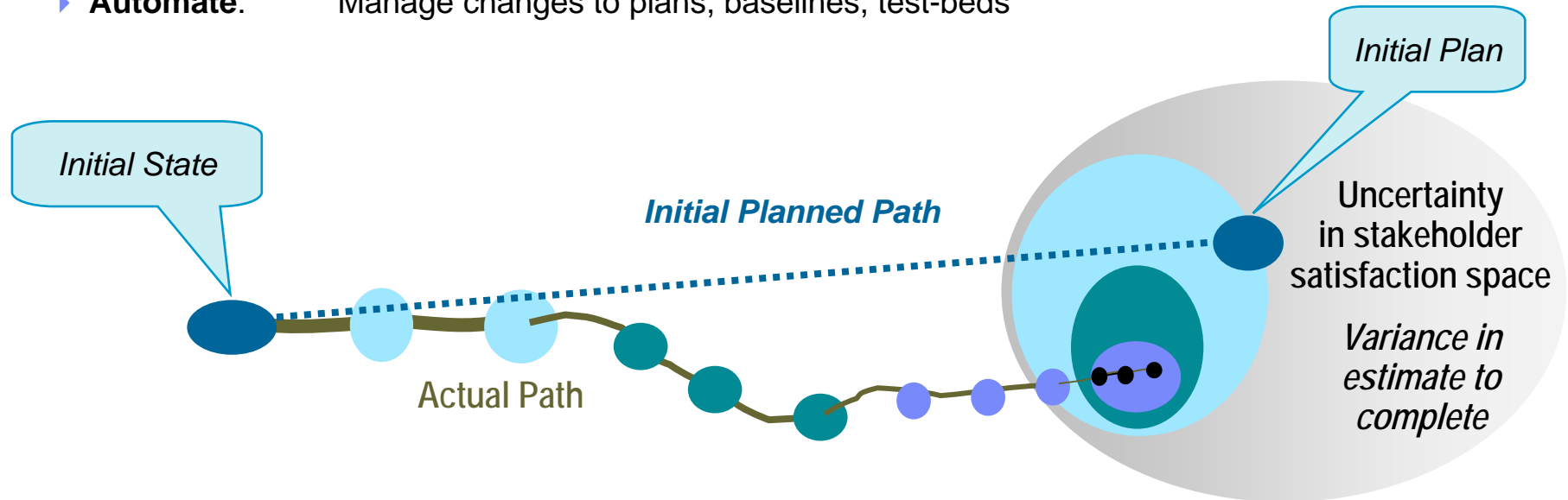
Then what?

- Over the lifecycle, reduce the variance further to improve likelihood of shipping



Measure and steer

- At onset of program
 - ▶ **Report:** Establish estimates/variances of effort, cost, establish initial plan
 - ▶ **Collaborate:** Set initial scope and expectations with stakeholders
 - ▶ **Automate:** Establish a collaborative development environment
- At each iteration, improve estimates and report
 - ▶ **Report:** Values and variances of progress achieved, quality achieved, resources expended
 - ▶ **Collaborate:** With stakeholders to refine scope and plans
 - ▶ **Automate:** Manage changes to plans, baselines, test-beds

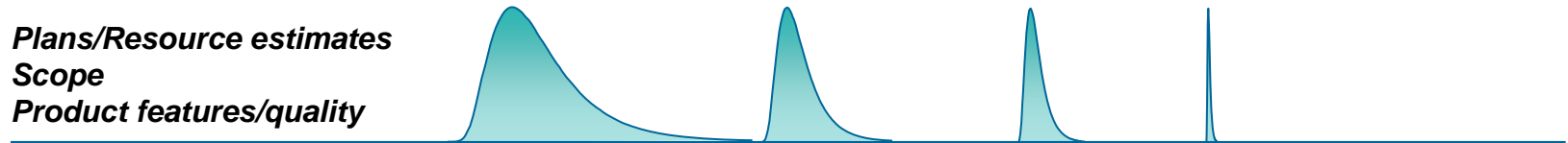


Agile Governance = Managing Uncertainty = Managing Variance

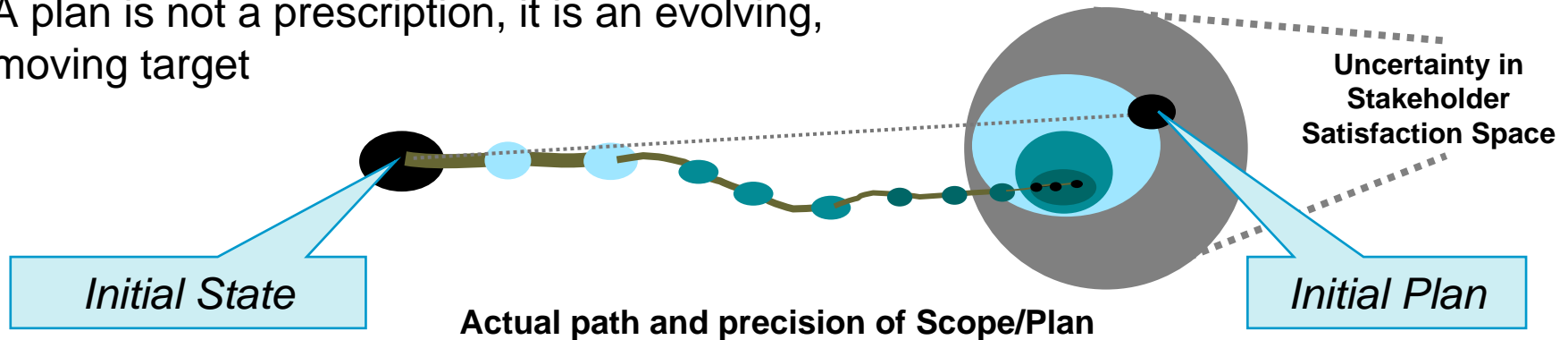
- A completion date is not a point in time, it is a probability distribution



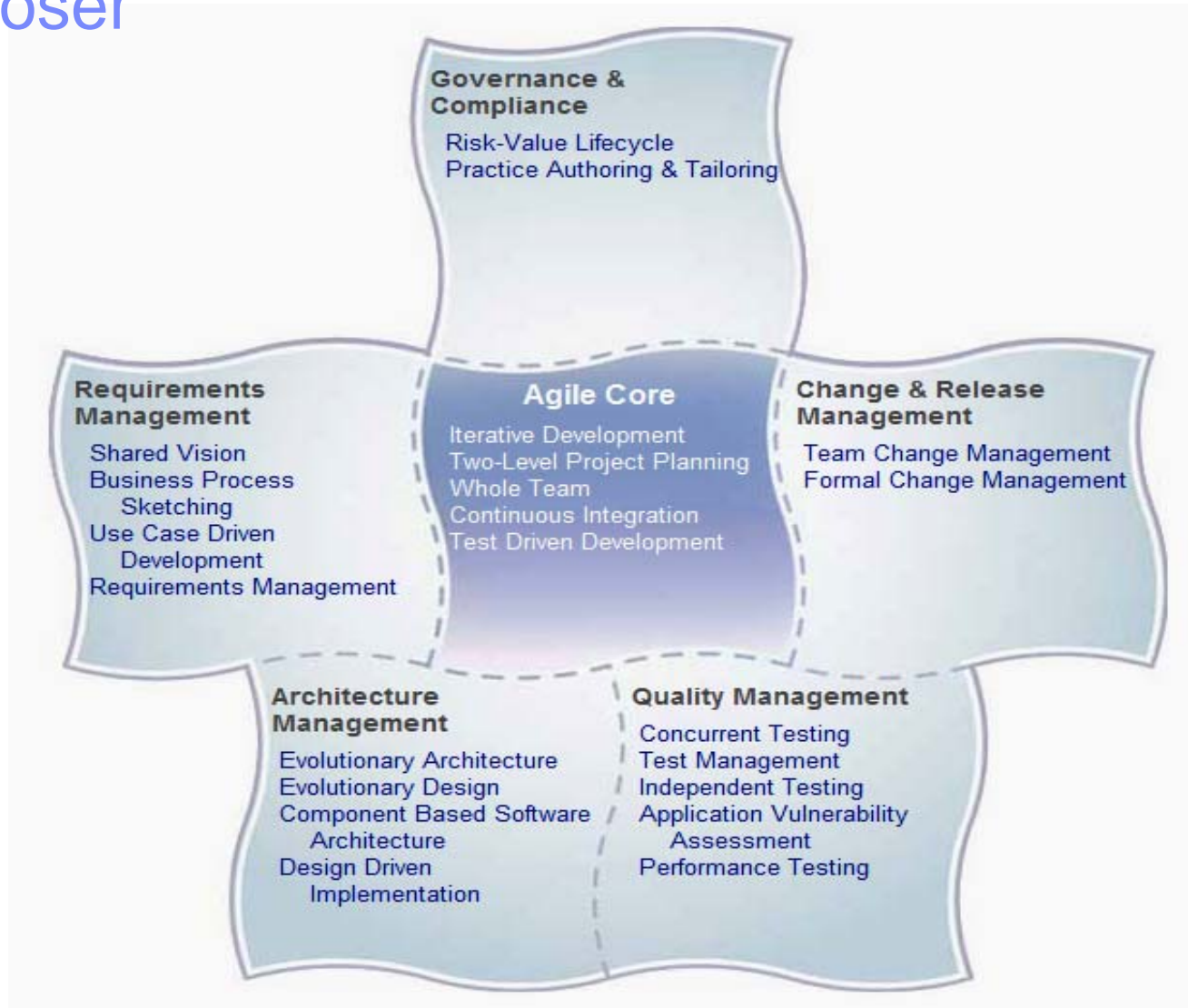
- Scope is not a requirements document, it is a continuous negotiation



- A plan is not a prescription, it is an evolving, moving target



Practices included as part of Rational Method Composer



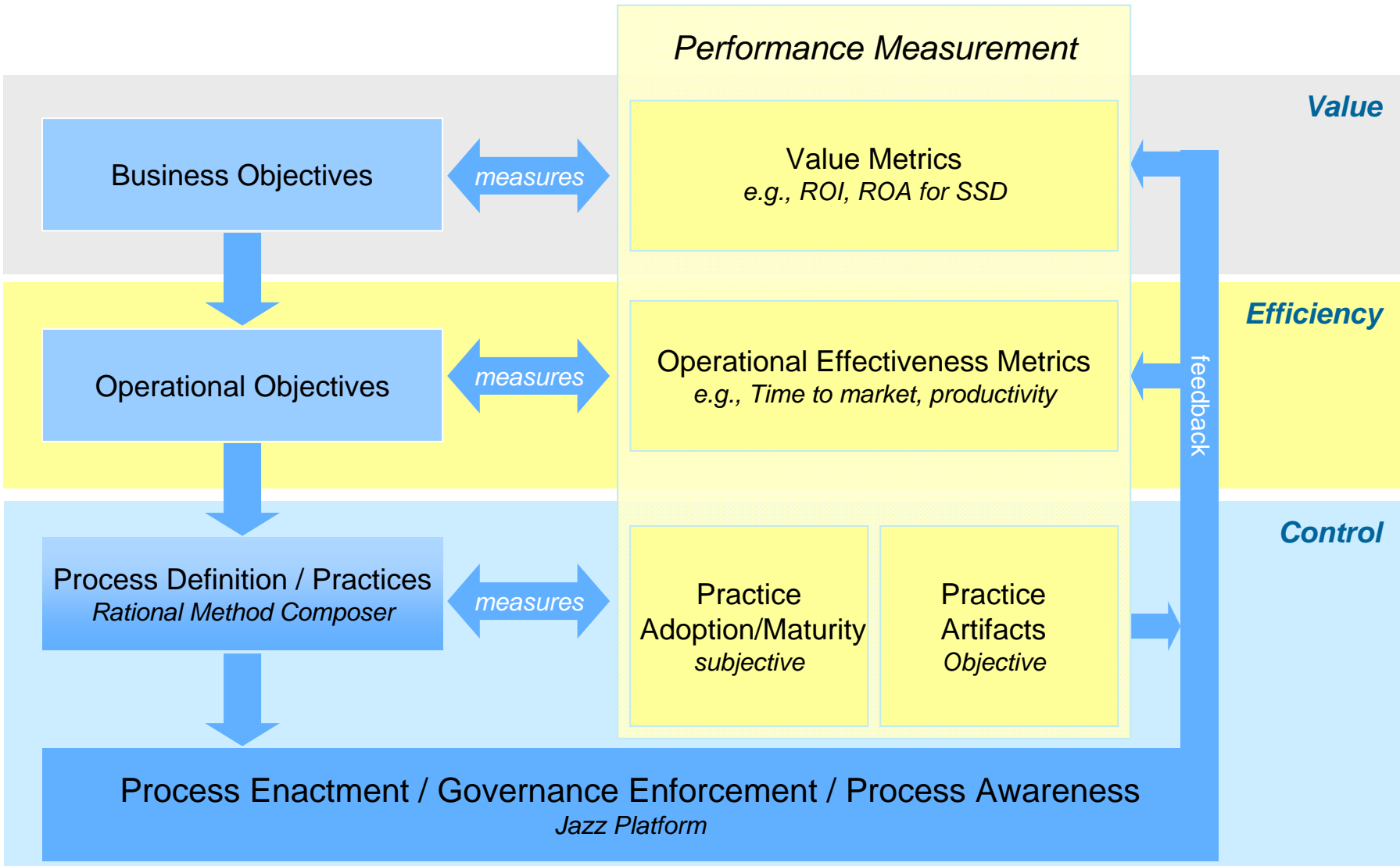
Agenda

- Transitioning to Agile Software Delivery
 - ▶ Economic governance
 - ▶ Steering and managing uncertainty

- **Metrics and Measures**
 - ▶ **Instrumentation for gaining control, improving efficiency and optimizing value**

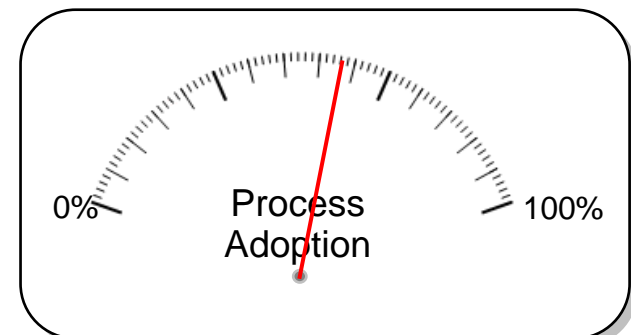
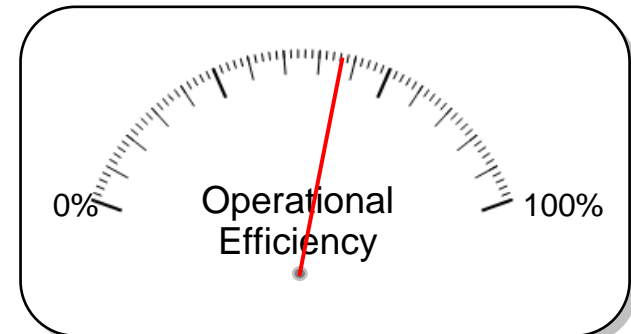
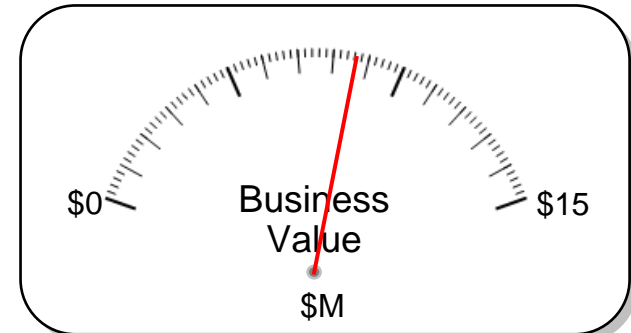
- Improving Software Economics
 - ▶ A framework for improvement priorities

Software and systems need a control framework



Meters for software and systems development and delivery improvement

- Value
 - ▶ Return on Investment (ROI)
 - ▶ Return on Assets (ROA)
 - ▶ Product revenue profile
- Efficiency
 - ▶ Time to market, productivity
 - ▶ Program portfolio investment profile
 - ▶ Defect phase containment, scrap and rework rates
 - ▶ Application service levels
 - ▶ Defect densities, requirements churn, design churn
 - ▶ Skills improvement, training cost reduction
- Control
 - ▶ Practice adoption, project checkpoints
 - ▶ Artifact time between gates
 - ▶ Collaboration, skills mix



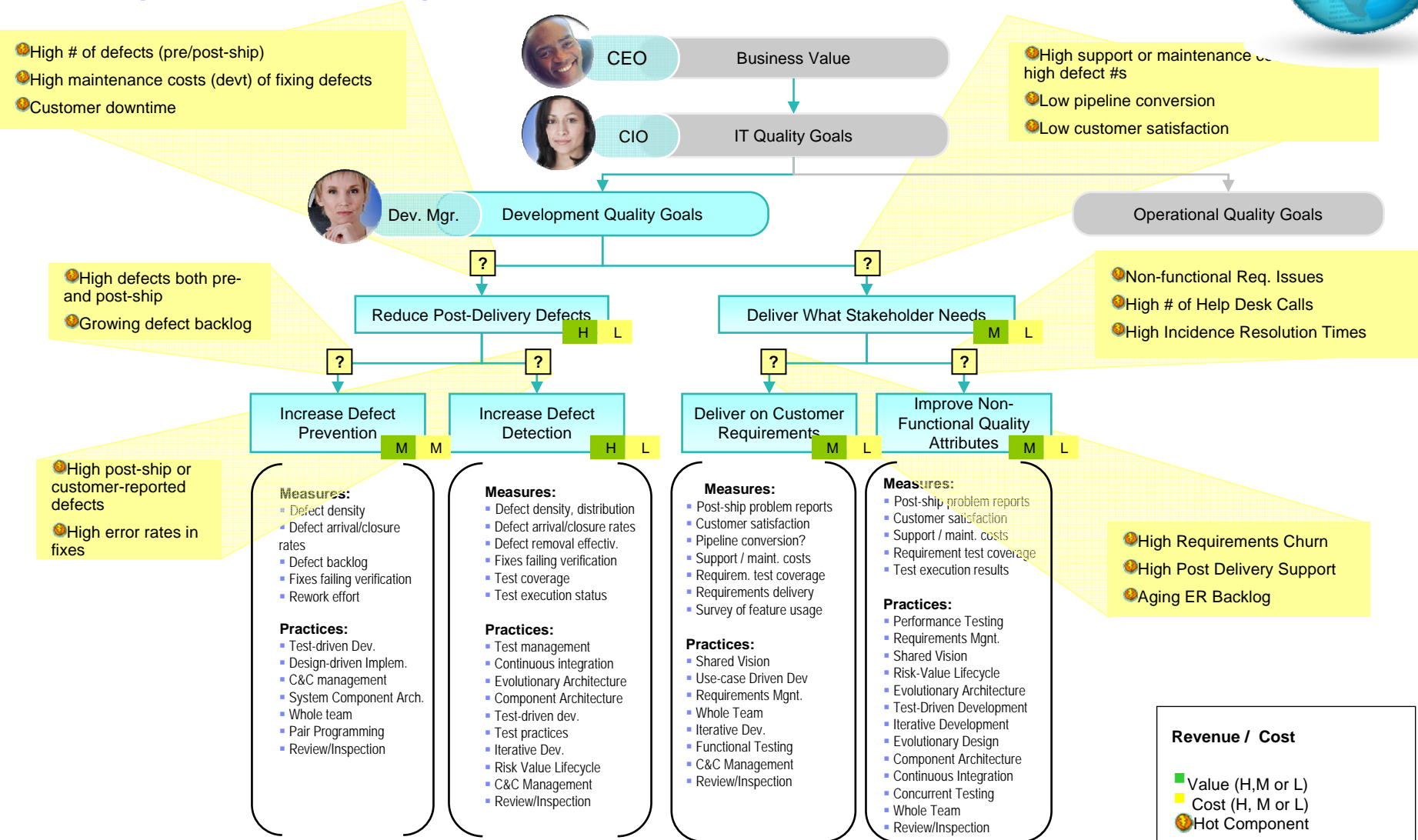
Tailor to organizational and project context

- Agree on business value measures: Cost, profit, return on assets, market share, etc.
- Determine project mix type
 - ▶ Choose appropriate operational measures
 - ▶ Choose practices to achieve measures for project mix
 - ▶ Establish measures and feedback channels for closed loop control

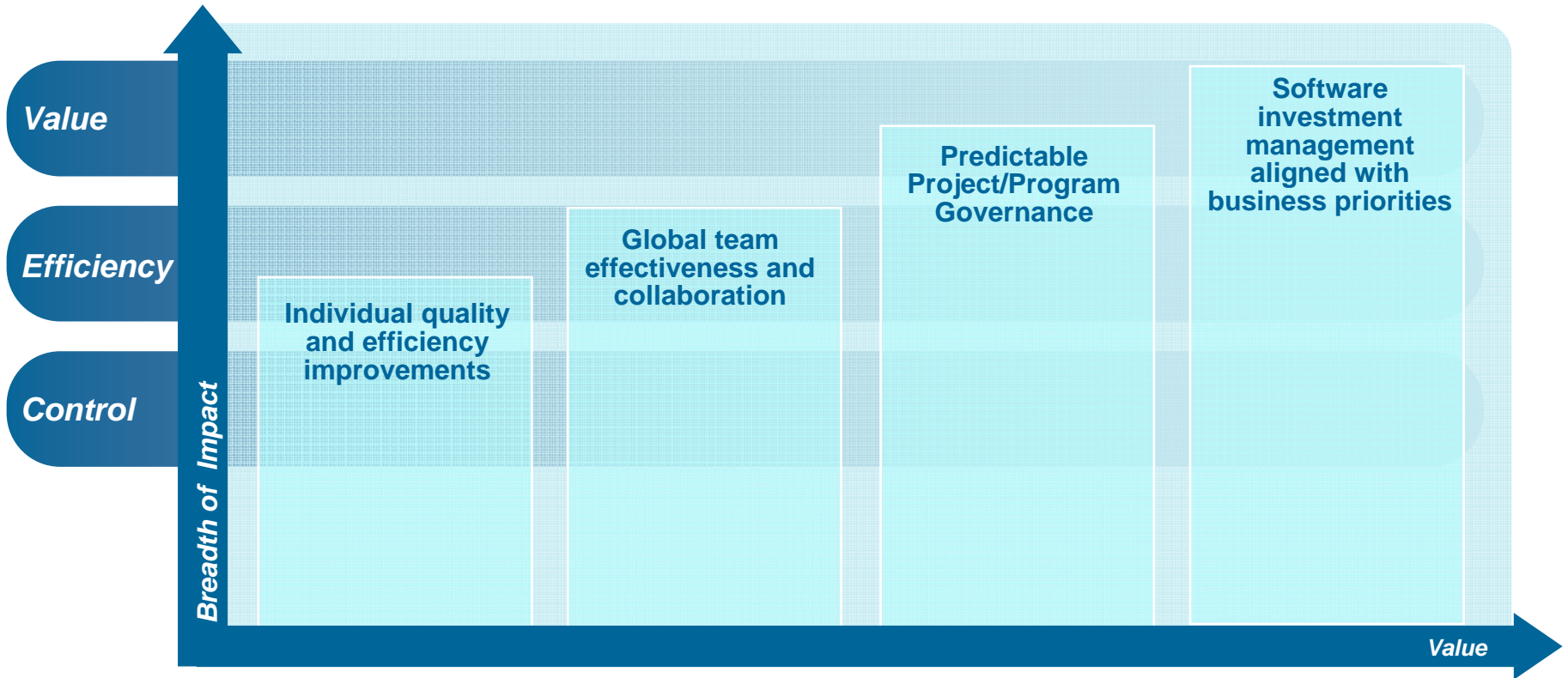
| | Variance Examples | | |
|--|---|--|--|
| | <i>Low</i> | <i>Medium</i> | <i>High</i> |
| Value (Business Measures) | <ul style="list-style-type: none"> ▪ Cost of operations | <ul style="list-style-type: none"> ▪ Market share growth ▪ Time to market for new features | <ul style="list-style-type: none"> ▪ Profitability of one-of-a-kind system |
| Efficiency (Operational Measure) | <ul style="list-style-type: none"> ▪ Cost per change request ▪ Individual productivity | <ul style="list-style-type: none"> ▪ Cost per change request ▪ Team Productivity | <ul style="list-style-type: none"> ▪ Architectural stability ▪ Organizational productivity |
| Controls | <ul style="list-style-type: none"> ▪ Self check for practices | <ul style="list-style-type: none"> ▪ Beta releases ▪ Defect densities, removal rates | <ul style="list-style-type: none"> ▪ Stakeholder demonstrations |
| Practices | <ul style="list-style-type: none"> ▪ Requirements management ▪ Change management ▪ Iterative development | <ul style="list-style-type: none"> ▪ Agile planning ▪ Test driven development | <ul style="list-style-type: none"> ▪ Shared vision ▪ Risk based lifecycle ▪ Evolutionary Architecture |



Select practices and measures based on business and operational objectives



Effective software delivery enabled by agility and measurement



Measures of increasing value

- | | | | |
|--|---|---|---|
| <ul style="list-style-type: none"> ▪ More creative time, less overhead time ▪ Painless governance ▪ More automation support | <ul style="list-style-type: none"> ▪ Fewer meetings ▪ Less scrap/rework ▪ Earlier defect detection ▪ Honest metrics | <ul style="list-style-type: none"> ▪ More reusable assets, services, skills, practices and measures ▪ More predictable outcomes ▪ Higher ROI | <ul style="list-style-type: none"> ▪ Optimized investments and supply chains ▪ Software development as an first class business process ▪ Business optimization |
|--|---|---|---|

Agenda

- Transitioning to Agile Software Delivery
 - ▶ Economic governance
 - ▶ Steering and managing uncertainty

- Metrics and Measures
 - ▶ Instrumentation for gaining control, improving efficiency and optimizing value

- **Improving Software Economics**
 - ▶ **A framework for improvement priorities**

Four patterns of success in achieving Agility at Scale

1. Scope management → *Asset based development*

Solutions evolve from requirements AND requirements evolve from available assets
As opposed to getting all the requirements right up front

2. Process management → *Rightsize the process*

Process and instrumentation rigor evolves from light to heavy
As opposed to the entire project's lifecycle process should be light or heavy depending on the character of the project

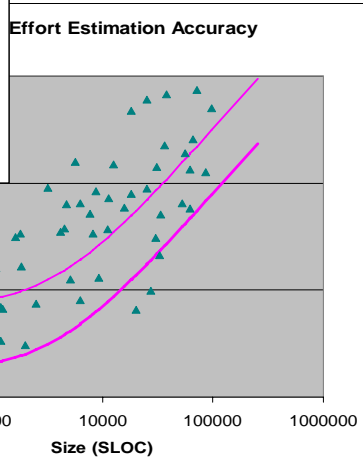
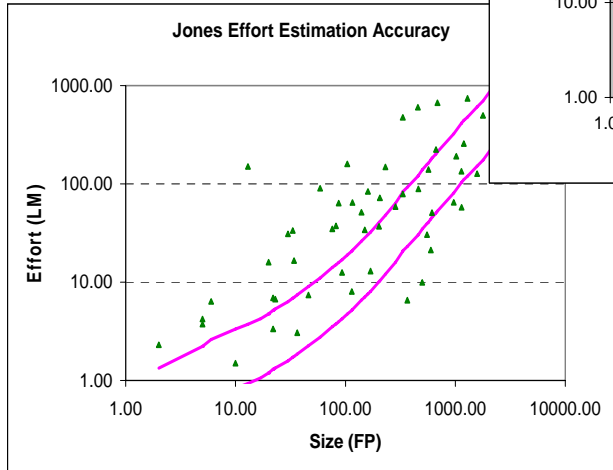
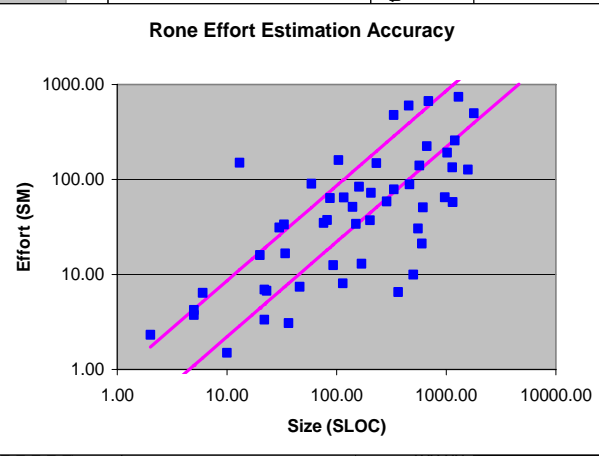
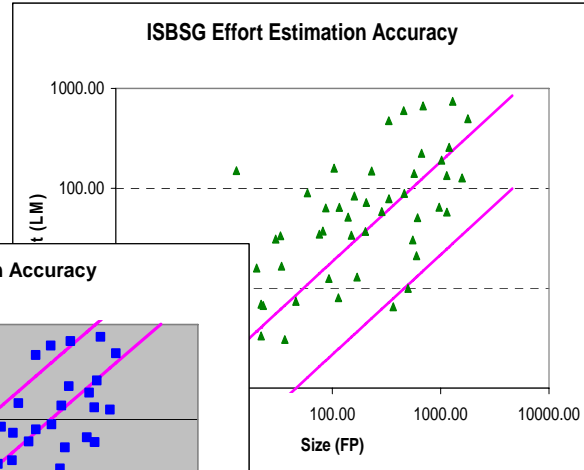
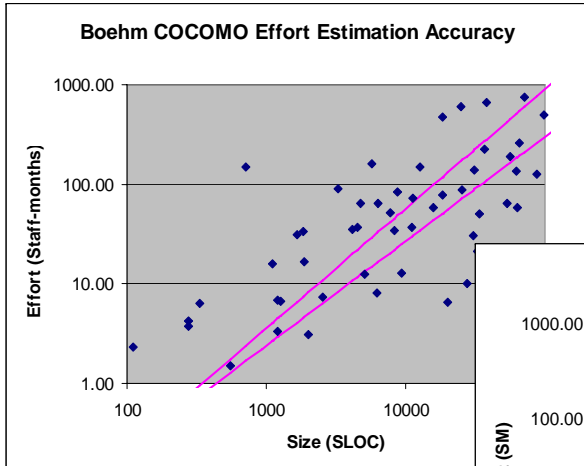
3. Progress management → *Honest assessments*

Healthy projects display a sequence of progressions and digressions
As opposed to progressing to 100% earned value with monotonically increasing progress against a static plan

4. Quality management → *Incremental demonstrable results*

Testing needs to be a 1st class, full lifecycle activity
As opposed to a subordinate, later lifecycle activity

Software cost models



From George Stark, Paul Oman, "A comparison of parametric Software Estimation Models using real project data", in press

Improving software economics

- Empirical software cost estimation models for:
 - ▶ Enterprise modernization, software maintenance
 - ▶ New developments, new releases, early prototypes
 - ▶ Packaged applications, systems engineering

Time or Cost To Build = (Complexity) (Process) * (Team) * (Tools)

Complexity

- Volume of human generated stuff
 - KSLOC, FPs, UCs
- Quality/performance
- Scope

Process

- Methods
- Maturity
- Agility
- Precedence

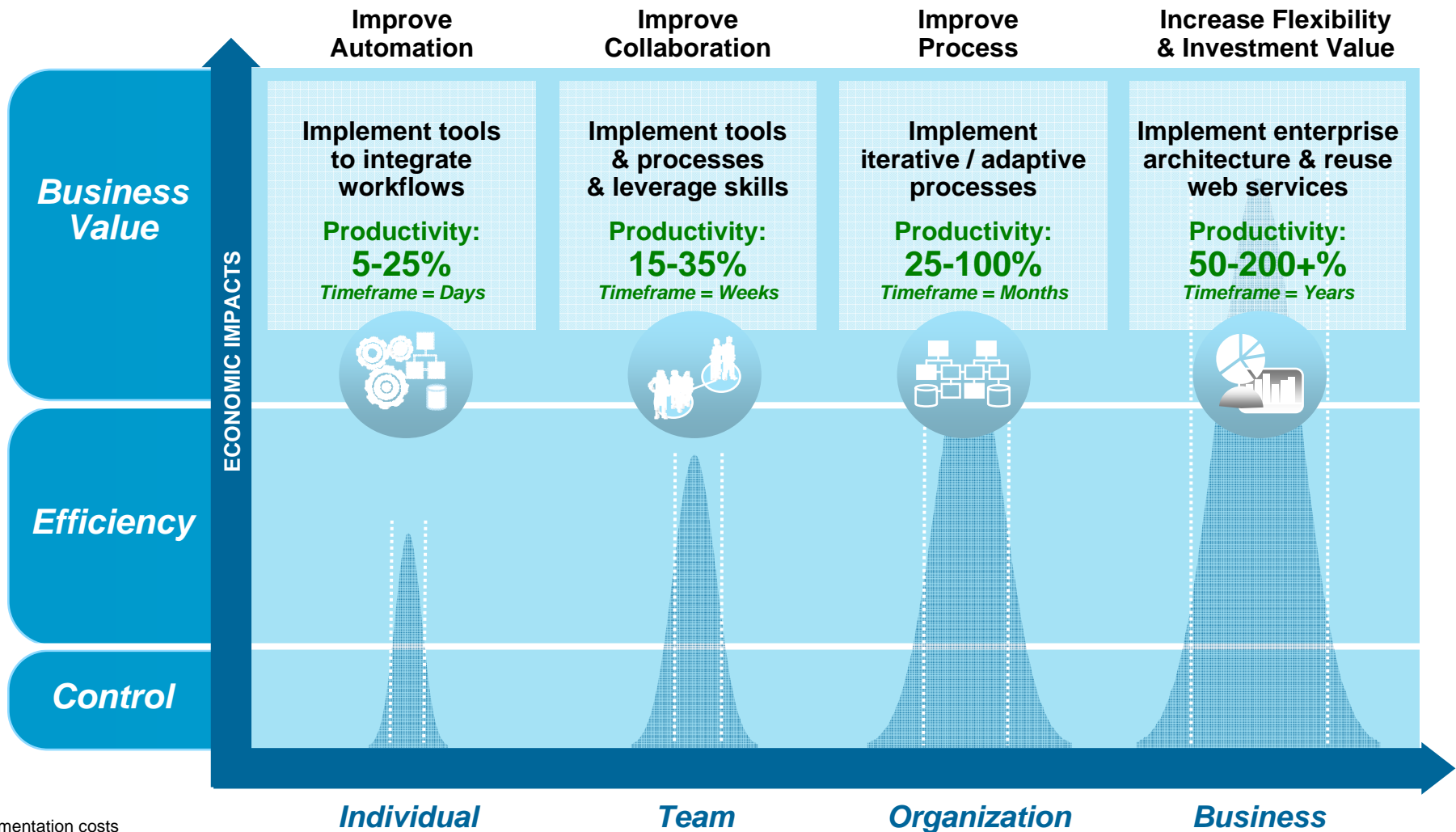
Team

- Skills/Experience
- Collaboration
- Motivation

Tools

- Automation
- Process enactment

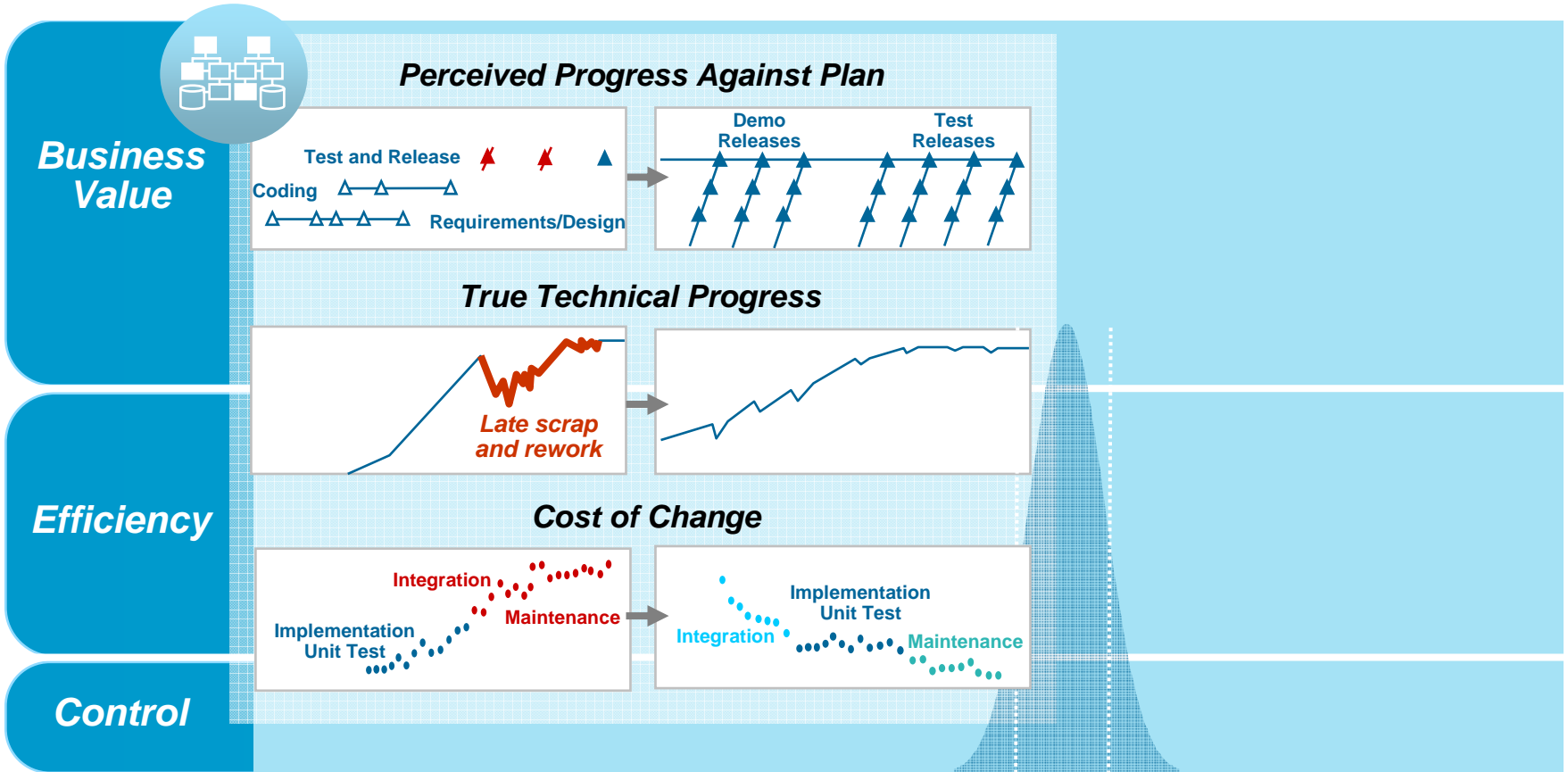
Achieve continuous improvement by measuring cost against business outcomes



Implementation costs are per person per year

Improve process to increase productivity by 25%-100%

Implement iterative / adaptive processes



Improving process and increasing flexibility

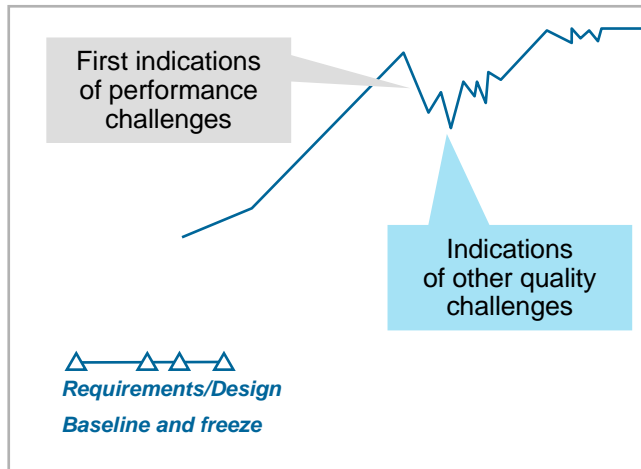
Reducing the significant uncertainties in quality and performance



WATERFALL DEVELOPMENT

Late quality and performance insight constrains flexibility to make tradeoffs

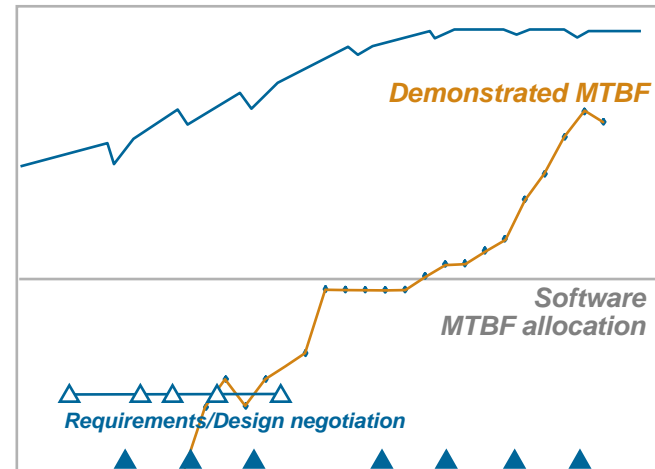
Measured progress and quality



- Speculative quality requirements
- Unpredictable cost/schedule performance
- Late shoehorning of suboptimal changes that impact quality

ITERATIVE DEVELOPMENT AND AGILE DELIVERY

Continuous quality and performance insight allows flexibility in trading off cost, quality, and features



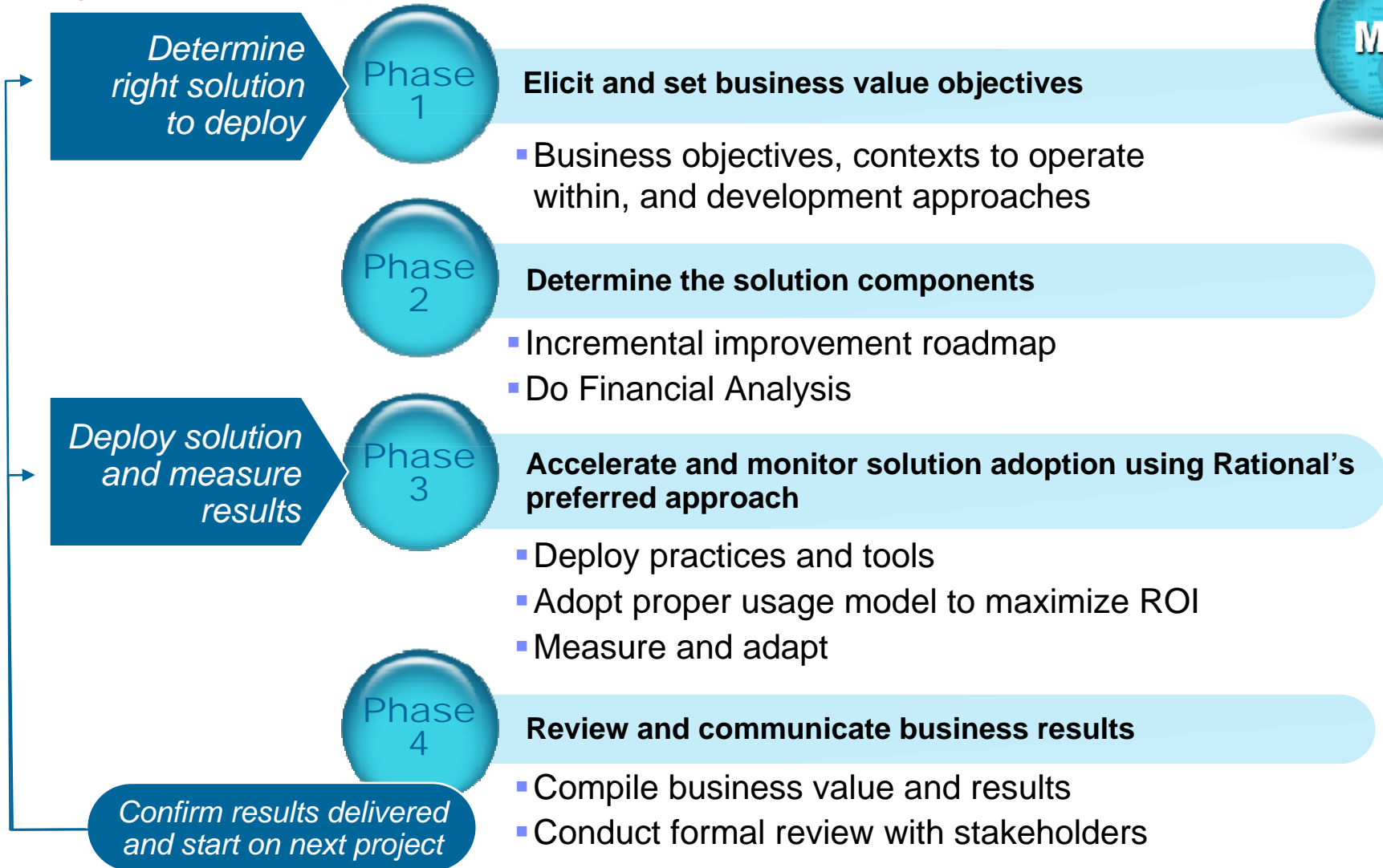
- Release qualities that matter
- Quality progressions/digressions
- Early requirement verification and/or negotiation

A1

4) finally, on slide 4, i changed the heading that was iterative development to both iterative development and agile delivery. not sure if that matters but it matches my paper and danny may prefer agile delivery rather than iterative development.

Author, 2009-4-29

Measured Capability Improvement Framework (MCIF): A systematic approach to software excellence



Some final thoughts

***Software delivery is a discipline of software economics
balancing risks and opportunities***

***Process enactment and measurement are imperatives
to achieving agility at scale***

***Software delivery requires a platform that is architected
for automation, collaboration and reporting***

Thank You

© Copyright IBM Corporation 2009. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, Rational, the Rational logo, Telelogic, the Telelogic logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.