Achieving Agility at Scale
Improving Software Economics

Monvorath Phongpaibul, Ph.D.
Solution Architects

Rational. software

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	Software Delivery
Distinct development phase	***Continuously evolving systems***
Distinct handoff to maintenance	***No distinct boundary between development and maintenance***
Requirements-design-code-test sequence	***Sequence of released capabilities with ever increasing value***
Phase and role specific tools	***Common platform of integrated process / tooling***
Collocated teams	***Distributed, web based collaboration***
Standard engineering governance	***Economic governance tailored to risk / reward profiles***
Engineering practitioner led	***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

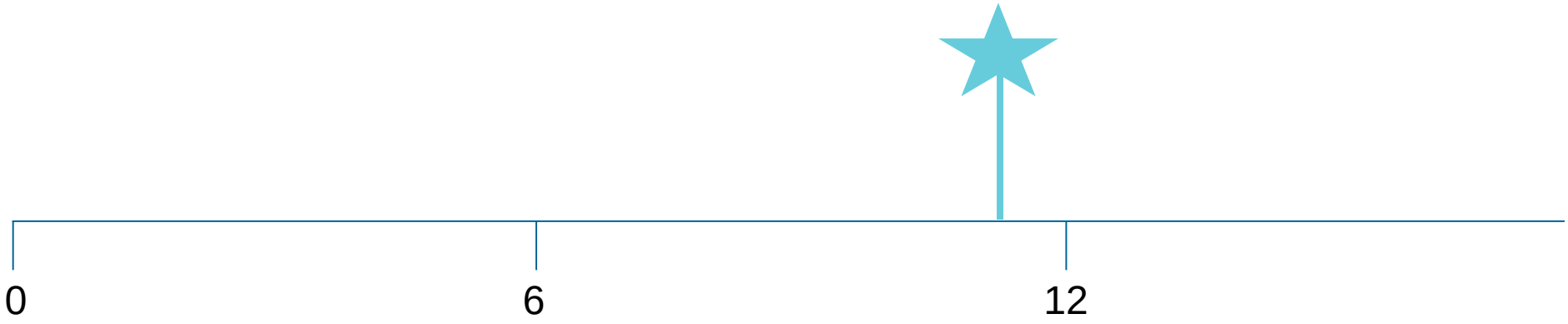
Evolving artifacts
- Scope (Problem specs)
- Design (Solution specs)
- Constraints (Planning specs)

Right-size the process
- Desired results drive process
- Manage variances

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

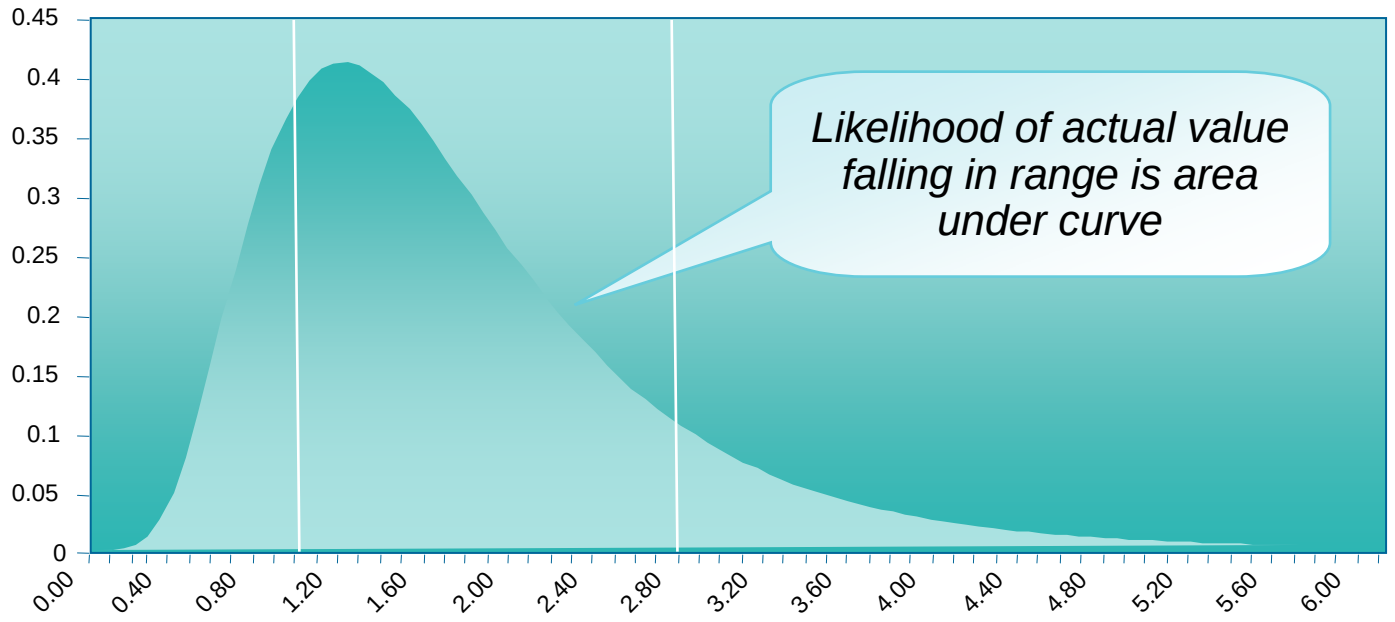
- Your estimators tell you it will be done in 11 months

- What do you do with the information?
  - Rest easy, believing there is no risk?



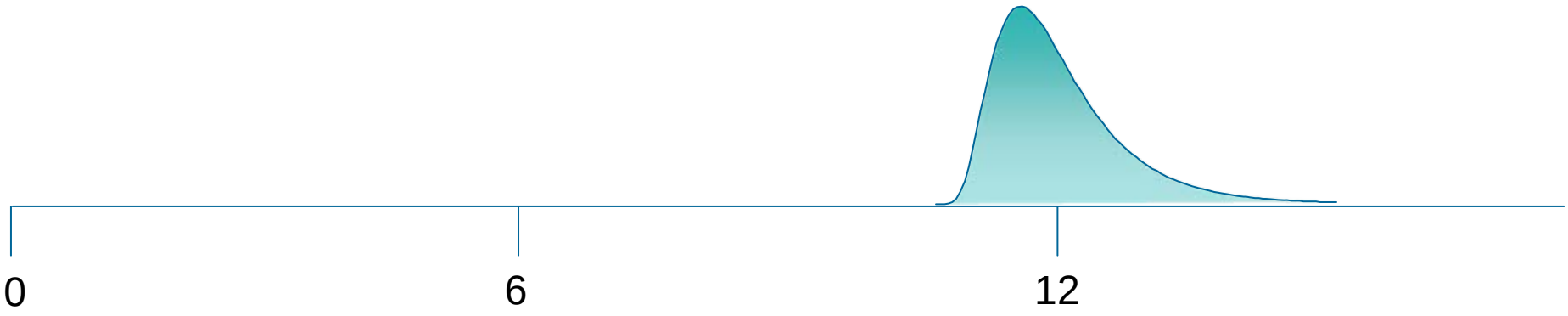
Maybe you realize that program parameters (cost, schedule, effort, quality, …) are random variables

- Area under curve describes probability of measurement falling in range



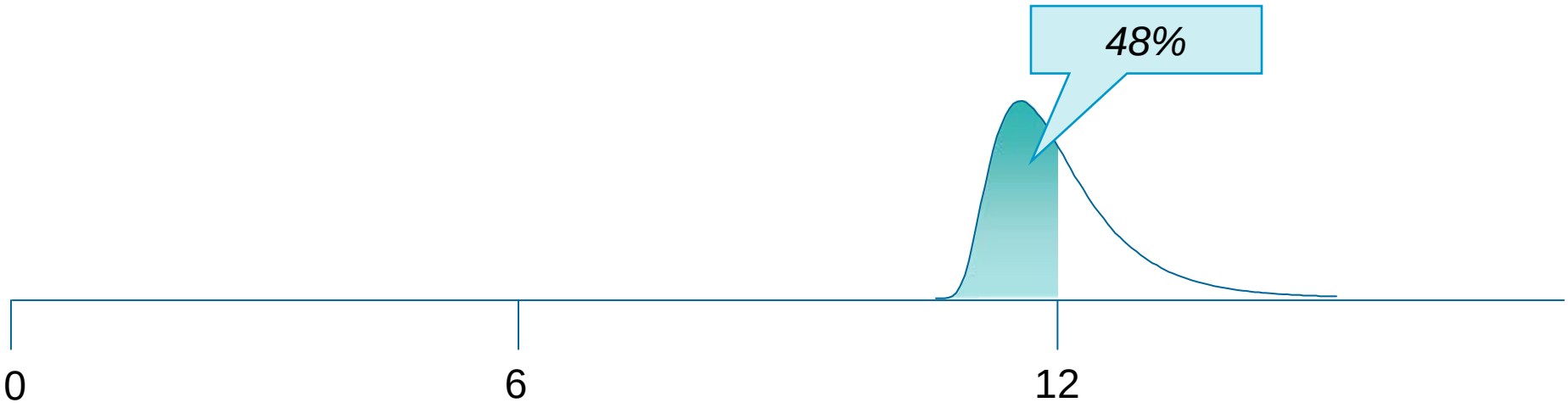
Imagine you have 12 months to deliver a business critical systems

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



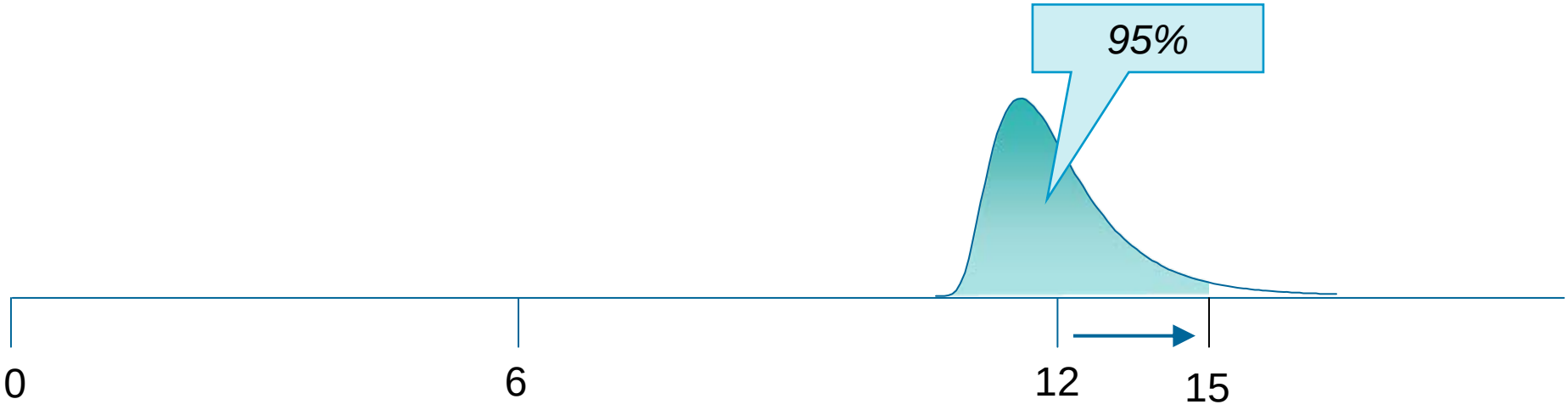
Imagine you have 12 months to deliver a business critical systems

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



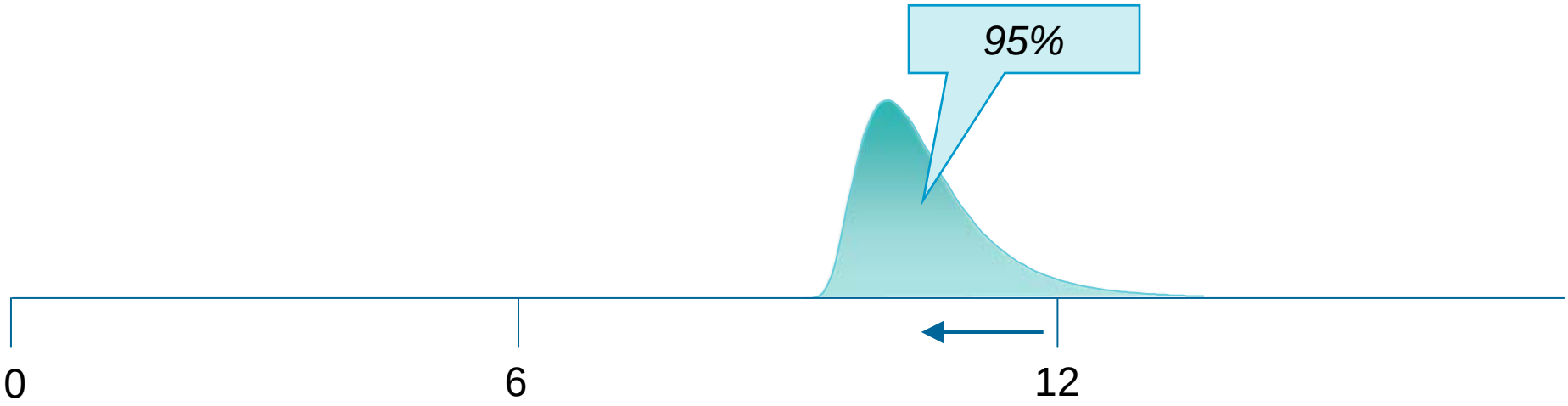
Then what?

- Move out the date to improve likelihood of shipping?



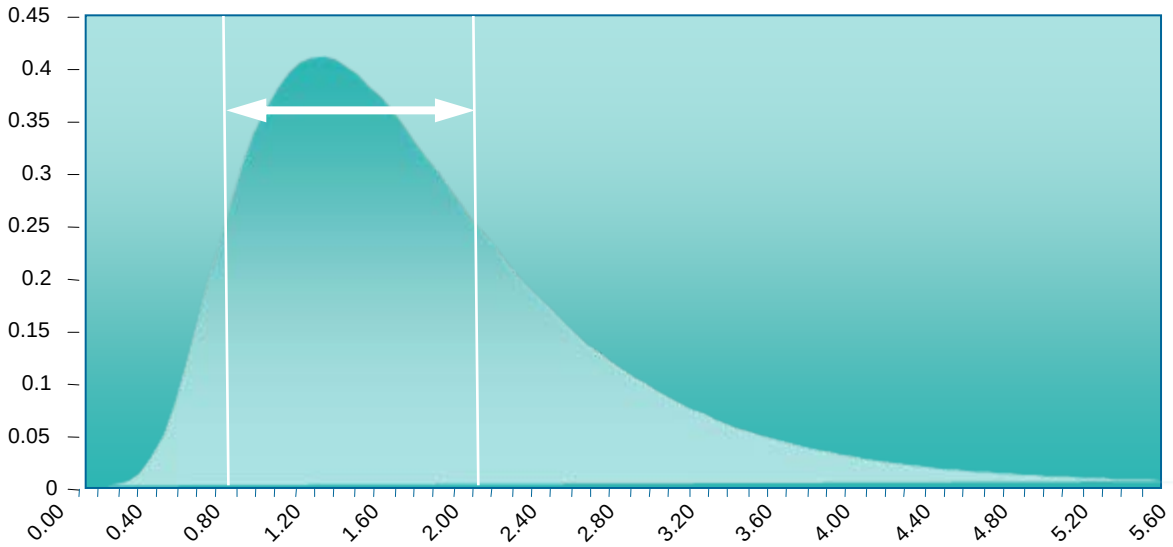
Then what?

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



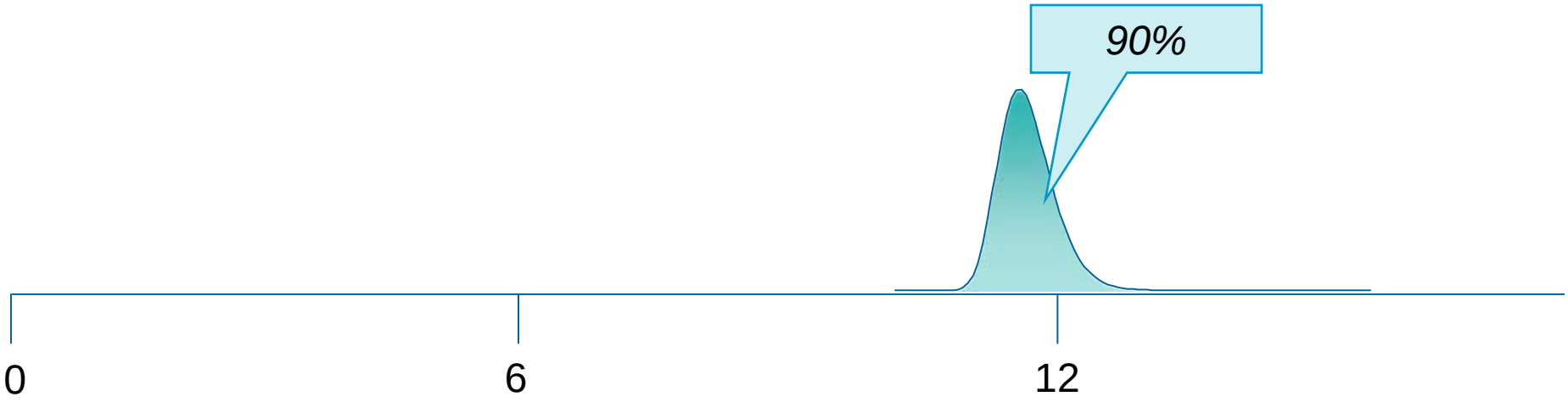
Managing variances in scope, solution, plans: The real key to improving software economics

- 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

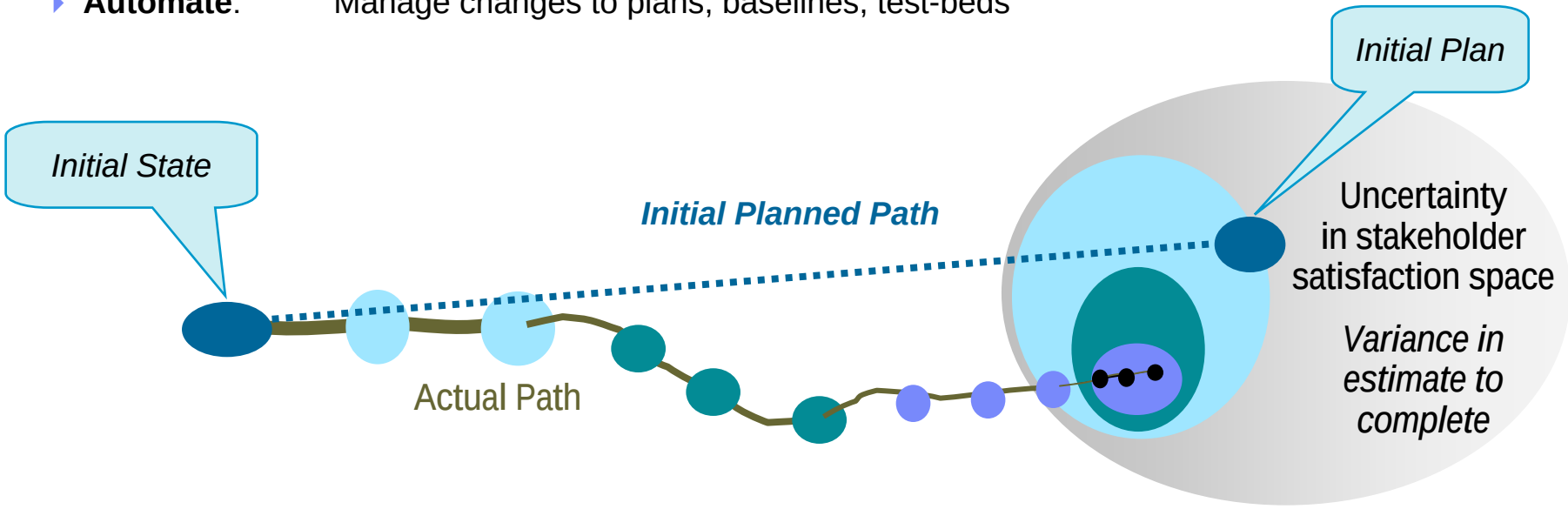


Then what?

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

Measure and steer

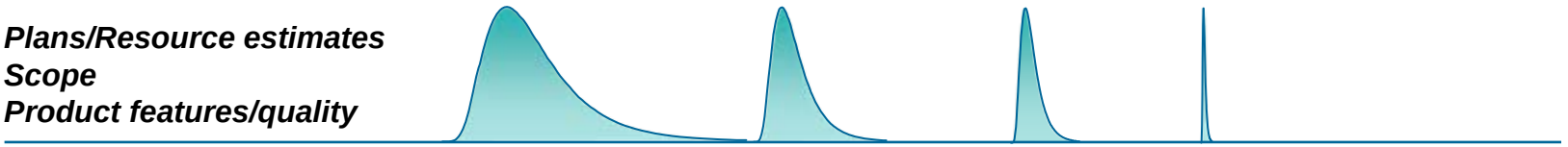
- At onset of program
  - **Report:** Establish estimates/variances of effort, cost, establish initial plan
  - **Collaborate**: Set initial scope and expectations with stakeholders
  - **Automate**: Establish a collaborative development environment
- At each iteration, improve estimates and report
  - **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



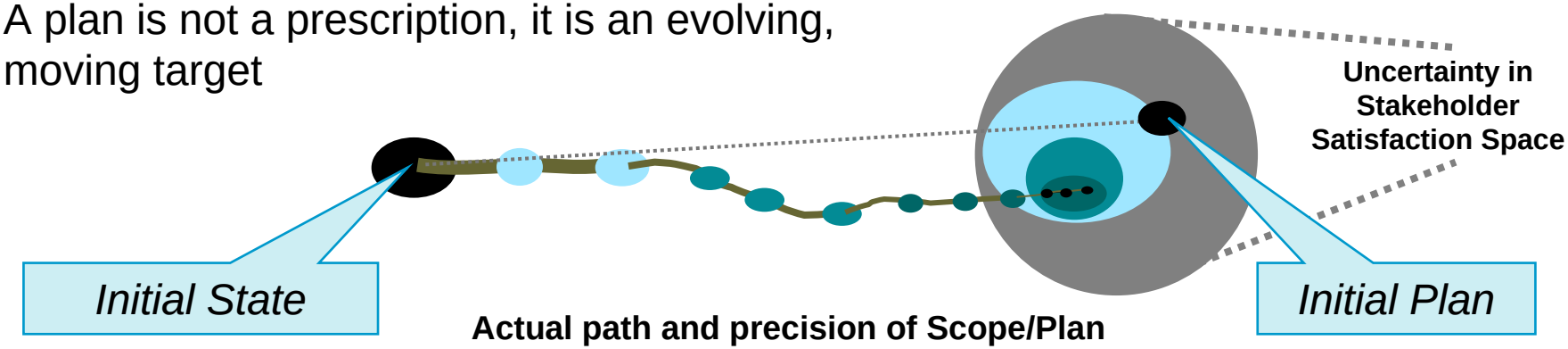
Agile Governance = Managing Uncertainty = Managing Variance

- 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

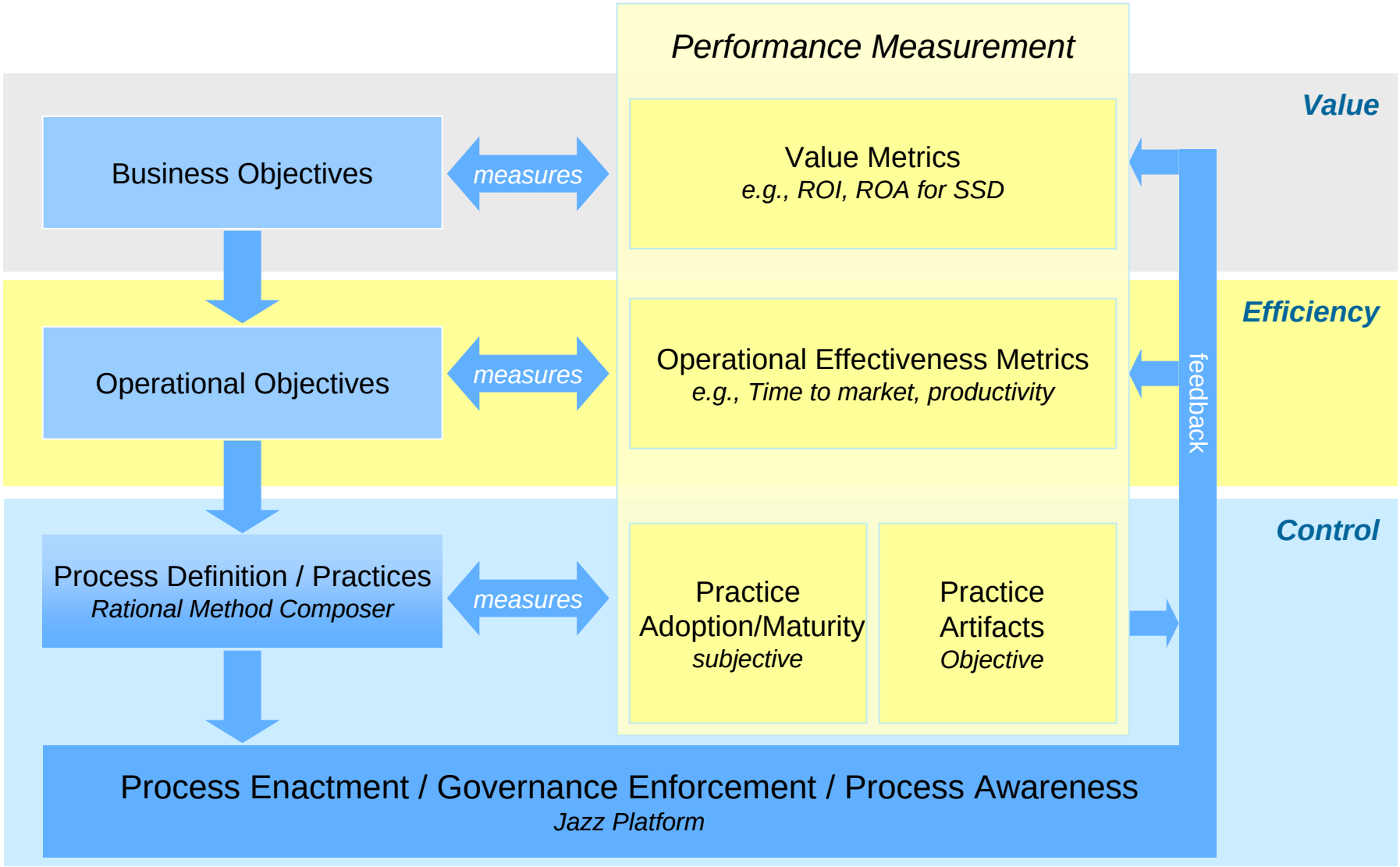


Practices included as part of Rational Method Composer

Agenda

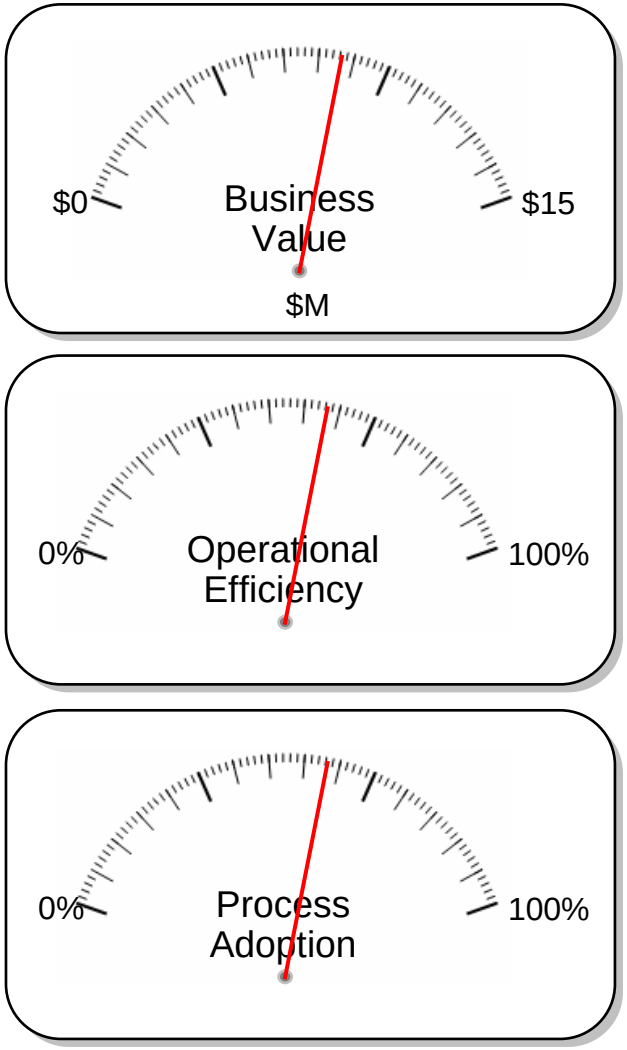
- Transitioning to Agile Software Delivery
  - Economic governance
  - Steering and managing uncertainty
- **Metrics and Measures**
  - **Instrumentation for gaining control, improving efficiency and optimizing value**
- Improving Software Economics
  - A framework for improvement priorities

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

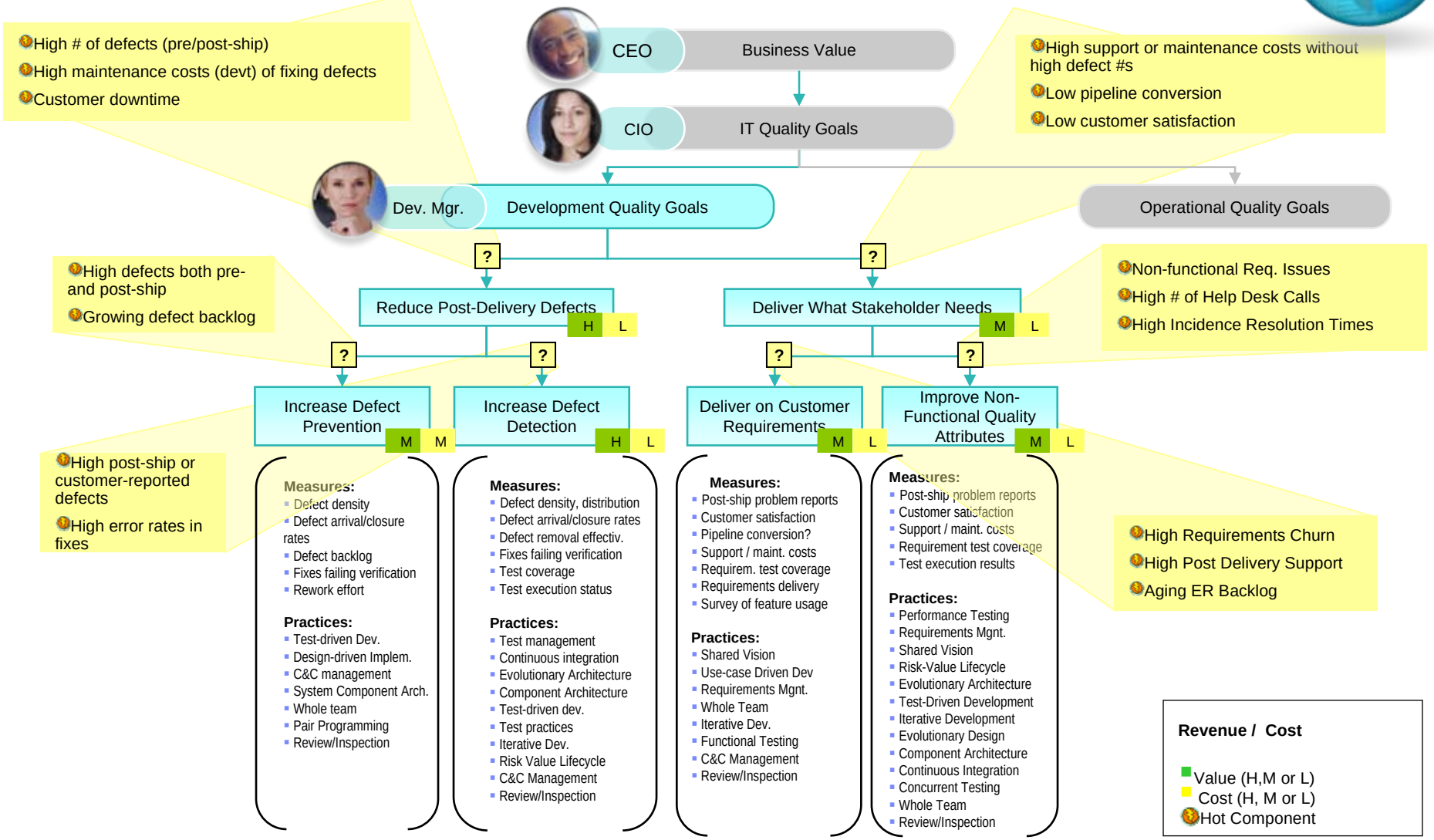


Tailor to organizational and project context

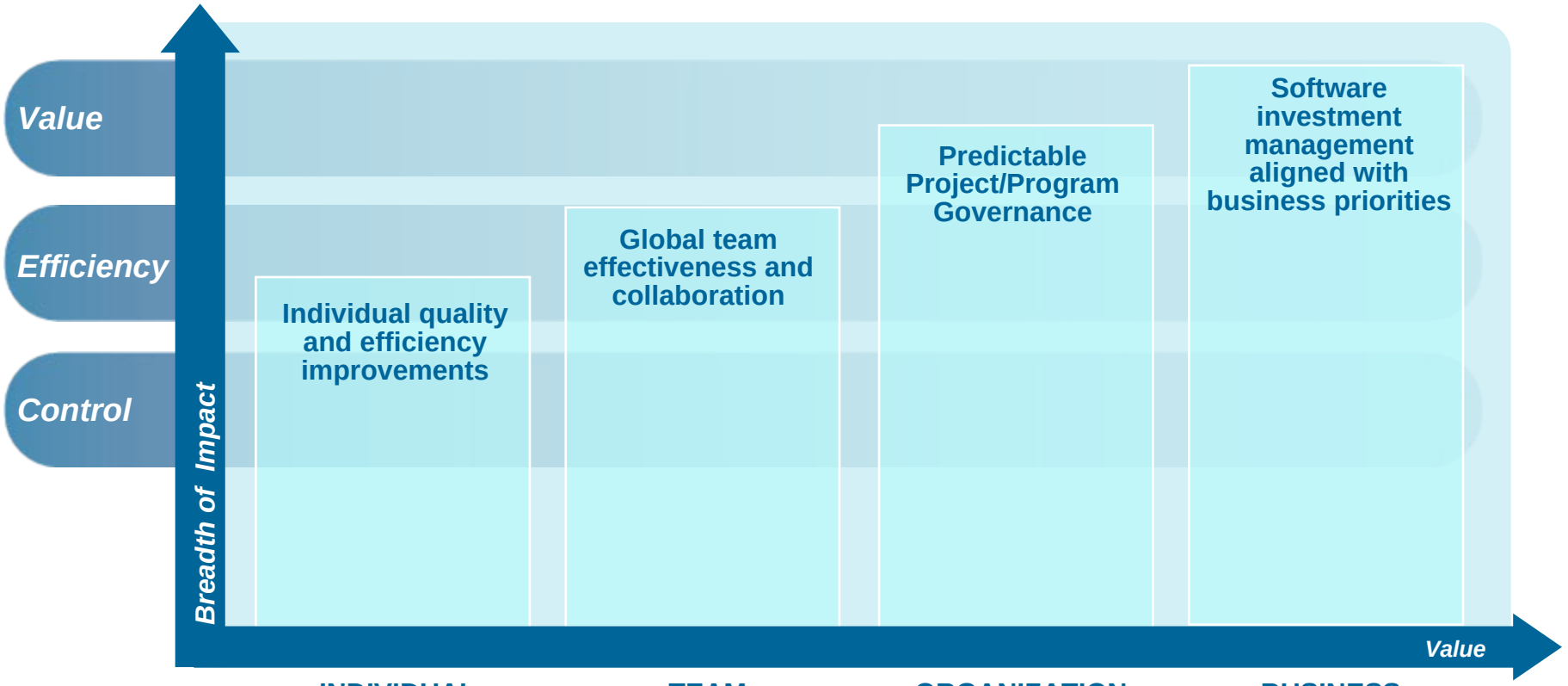
- Agree on business value measures: Cost, profit, return on assets, market share, etc.
- Determine project mix type
  - Choose appropriate operational measures
  - Choose practices to achieve measures for project mix
  - Establish measures and feedback channels for closed loop control

	Variance Examples		
	Low	***Medium***	***High***
Value (Business Measures)	▪ Cost of operations	▪ Market share growth ▪ Time to market for new features	▪ Profitability of one-of-a-kind system
Efficiency (Operational Measure)	▪ Cost per change request ▪ Individual productivity	▪ Cost per change request ▪ Team Productivity	▪ Architectural stability ▪ Organizational productivity
Controls	▪ Self check for practices	▪ Beta releases ▪ Defect densities, removal rates	▪ Stakeholder demonstrations
Practices	▪ Requirements management ▪ Change management ▪ Iterative development	▪ Agile planning ▪ Test driven development	▪ 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



	INDIVIDUAL	TEAM	ORGANIZATION	BUSINESS
Measures of increasing value	▪ More creative time, less overhead time ▪ Painless governance ▪ More automation support	▪ Fewer meetings ▪ Less scrap/rework ▪ Earlier defect detection ▪ Honest metrics	▪ More reusable assets, services, skills, practices and measures ▪ More predictable outcomes ▪ Higher ROI	▪ 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

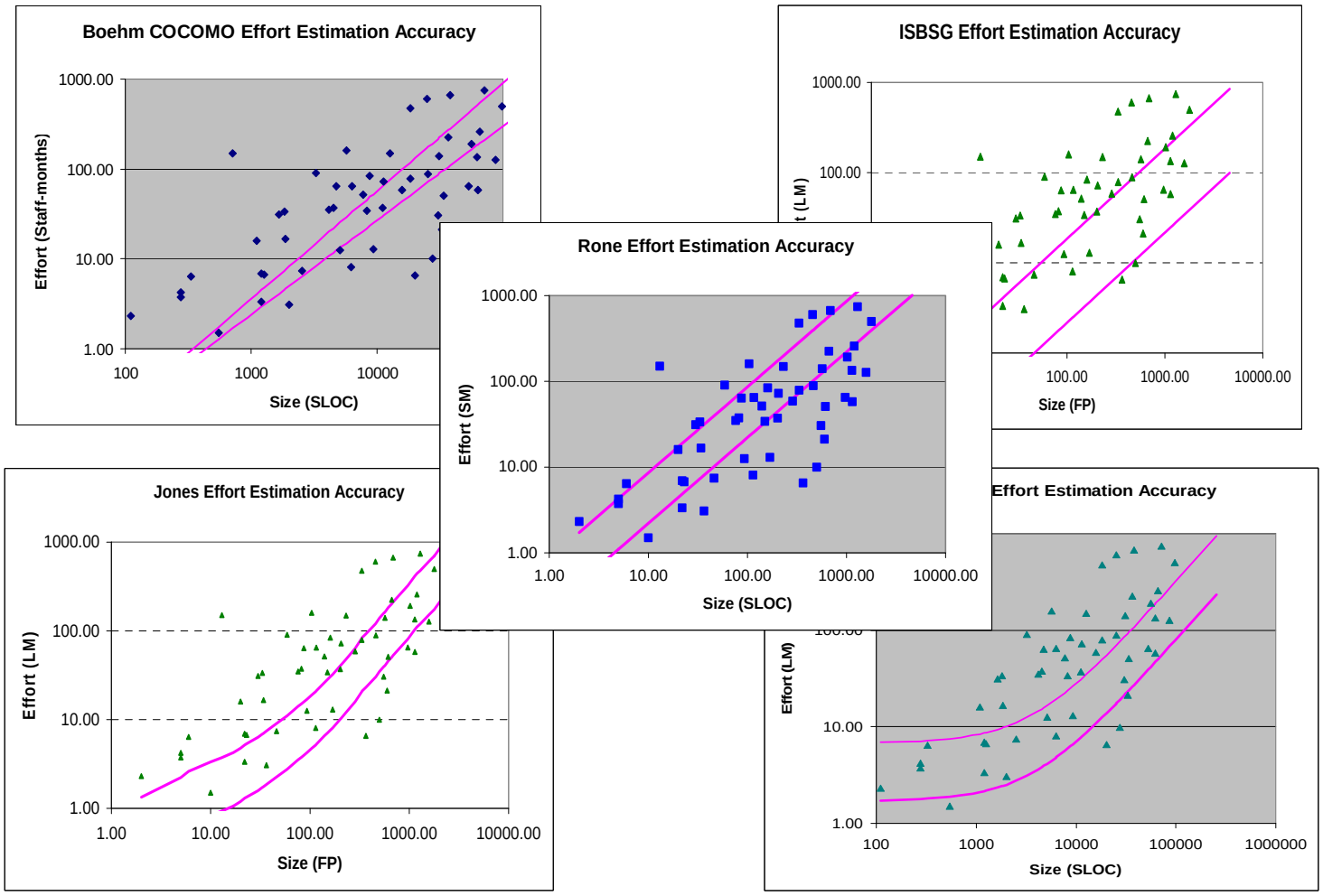
1. **Scope management ➔ *Asset based development***
   Solutions evolve from requirements AND requirements evolve from available assets
   As opposed to getting all the requirements right up front

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

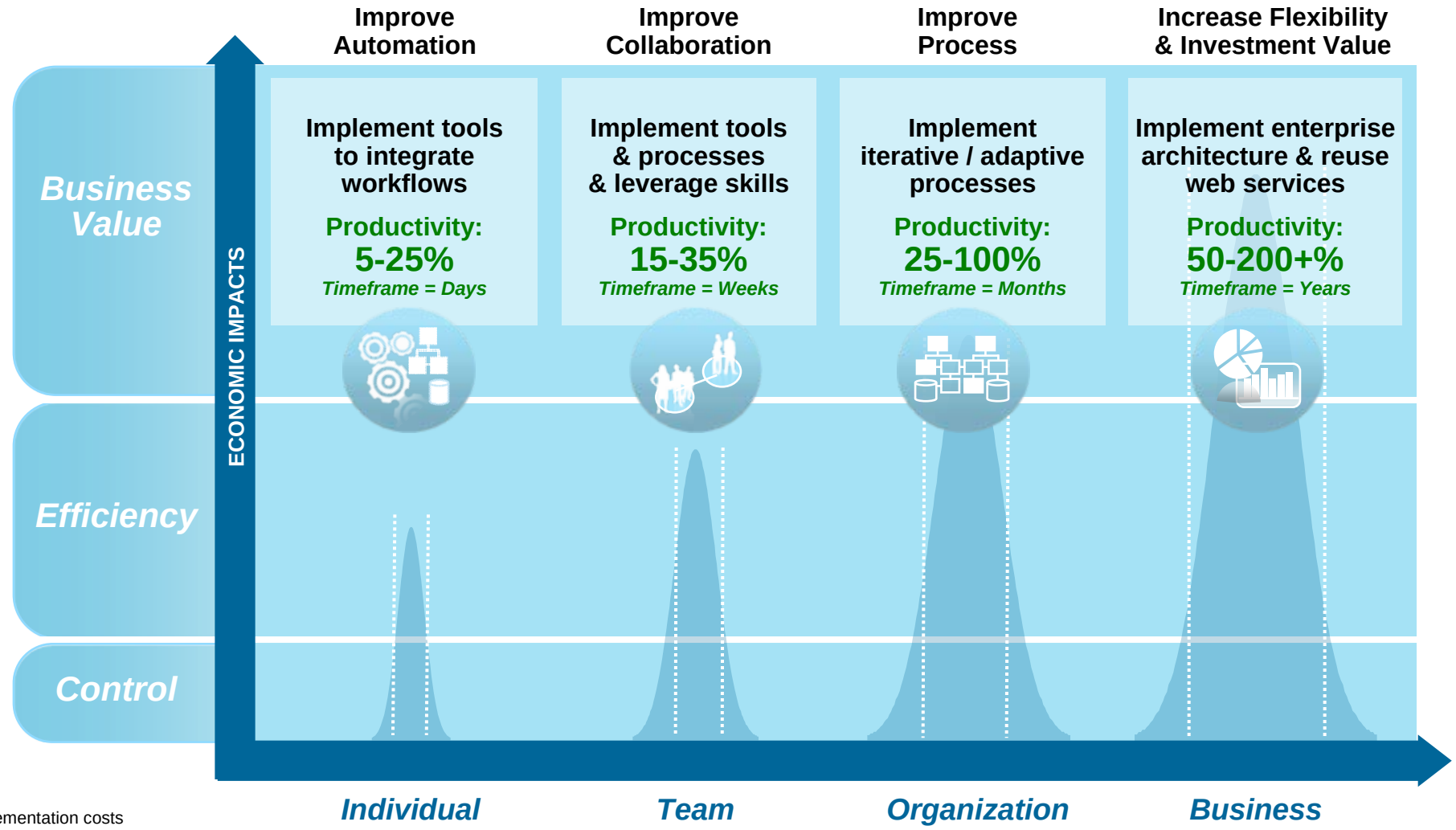
Improving software economics

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

$$\text{Time or Cost To Build} = (\text{Complexity})^{(\text{Process})} * (\text{Team}) * (\text{Tools})$$

Complexity	Process	Team	Tools
▪ Volume of human generated stuff - KSLOC, FPs, UCs ▪ Quality/performance ▪ Scope	▪ Methods ▪ Maturity ▪ Agility ▪ Precedence	▪ Skills/Experience ▪ Collaboration ▪ Motivation	▪ Automation ▪ Process enactment

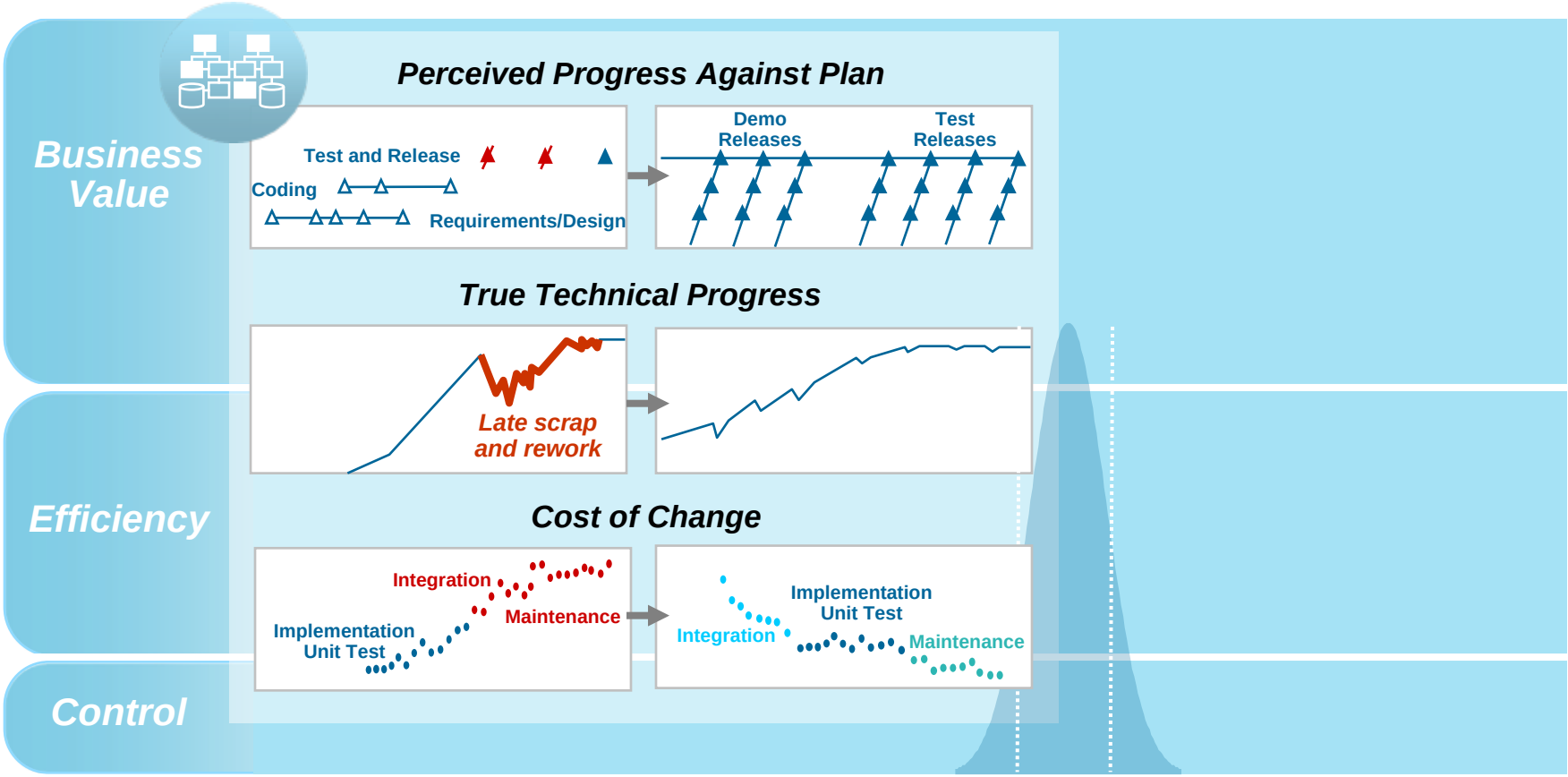
Achieve continuous improvement by measuring cost against business outcomes



Implementation costs are per person per year

Improve process to increase productivity by 25%-100%

Implement iterative / adaptive processes



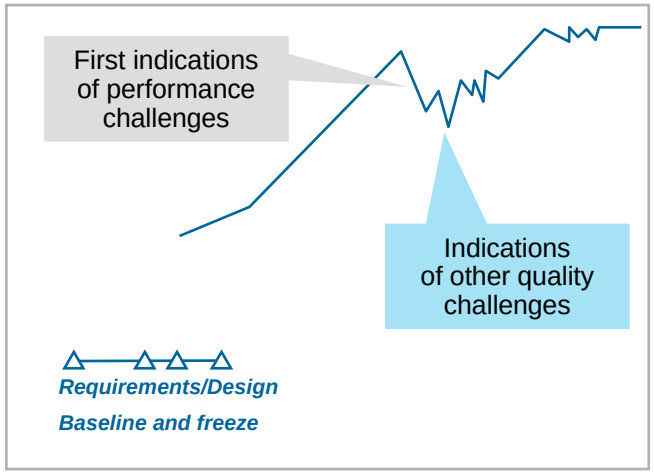
Improving process and increasing flexibility

Reducing the significant uncertainties in quality and performance

Improve Process and Increase Flexibility

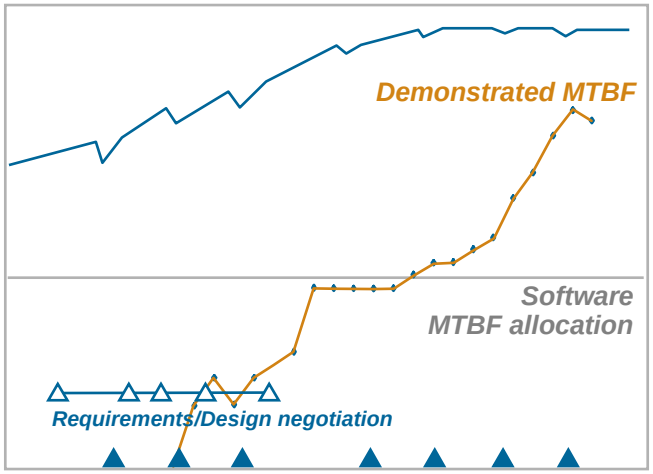
WATERFALL DEVELOPMENT → **ITERATIVE DEVELOPMENT AND AGILE DELIVERY**

Late quality and performance insight constrains flexibility to make tradeoffs



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

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

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

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