

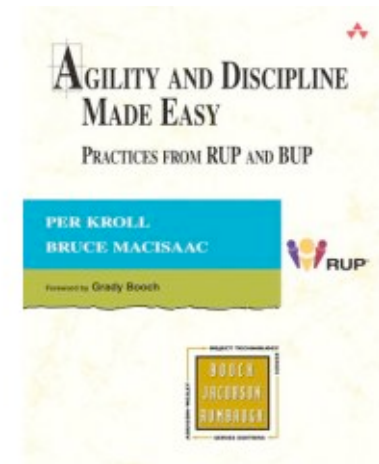
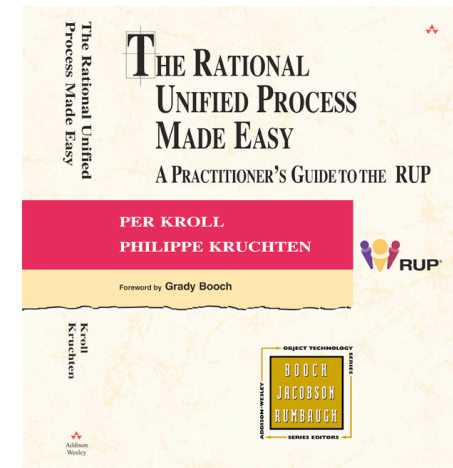
# Real World Trends Impacting Software Best Practices

***Per Kroll***



## Per Kroll - Background

- Development Manager - Commercial Methods
- Technology Strategist – IBM Rational
- Project lead – Eclipse Process Framework
- Author of
  - ▶ The Rational Unified Process Made Easy – A Practitioner's Guide to RUP
  - ▶ Agility and Discipline Made Easy – Practices from RUP and BUP





# Agenda

- Trends impacting software development
- Key principles for business-driven development
- An End-to-End Solution for SOA
- Summary and conclusions



# Agenda

- Trends impacting software development
- Key principles for business-driven development
- An End-to-End Solution for SOA
- Summary and conclusions

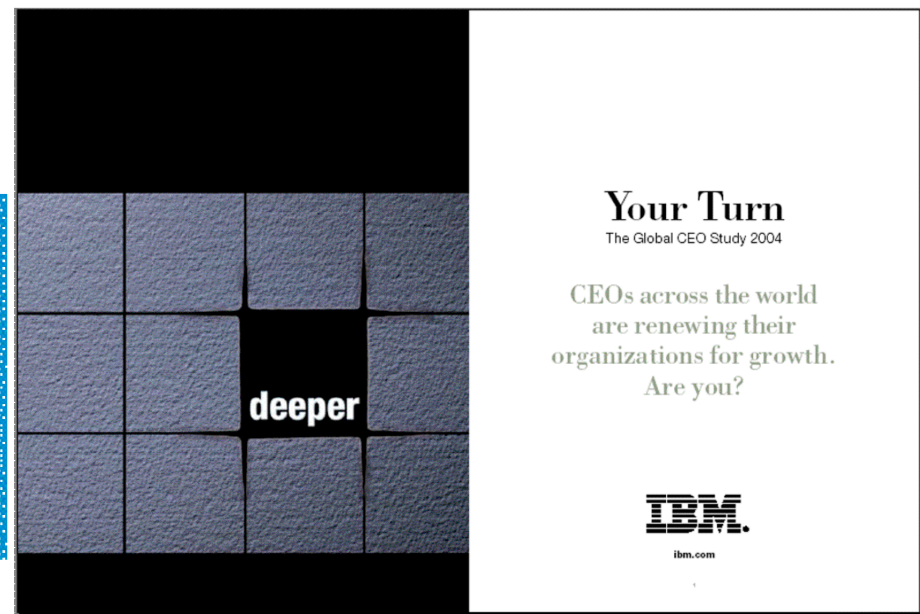


## IBM 2004 Global CEO Study

*Growth is the Number One Priority*

**Survey of  
456 CEOs worldwide**

**Top business priority:**  
*Growth driven by innovation*



[http://www-1.ibm.com/services/us/bcs/html/2004\\_global\\_ceo\\_study\\_gen.html](http://www-1.ibm.com/services/us/bcs/html/2004_global_ceo_study_gen.html)



# Trends Impacting Software Development

- **Efficiency**

- ▶ Business growth pressures to efficiently deliver quality software

- **Compliance**

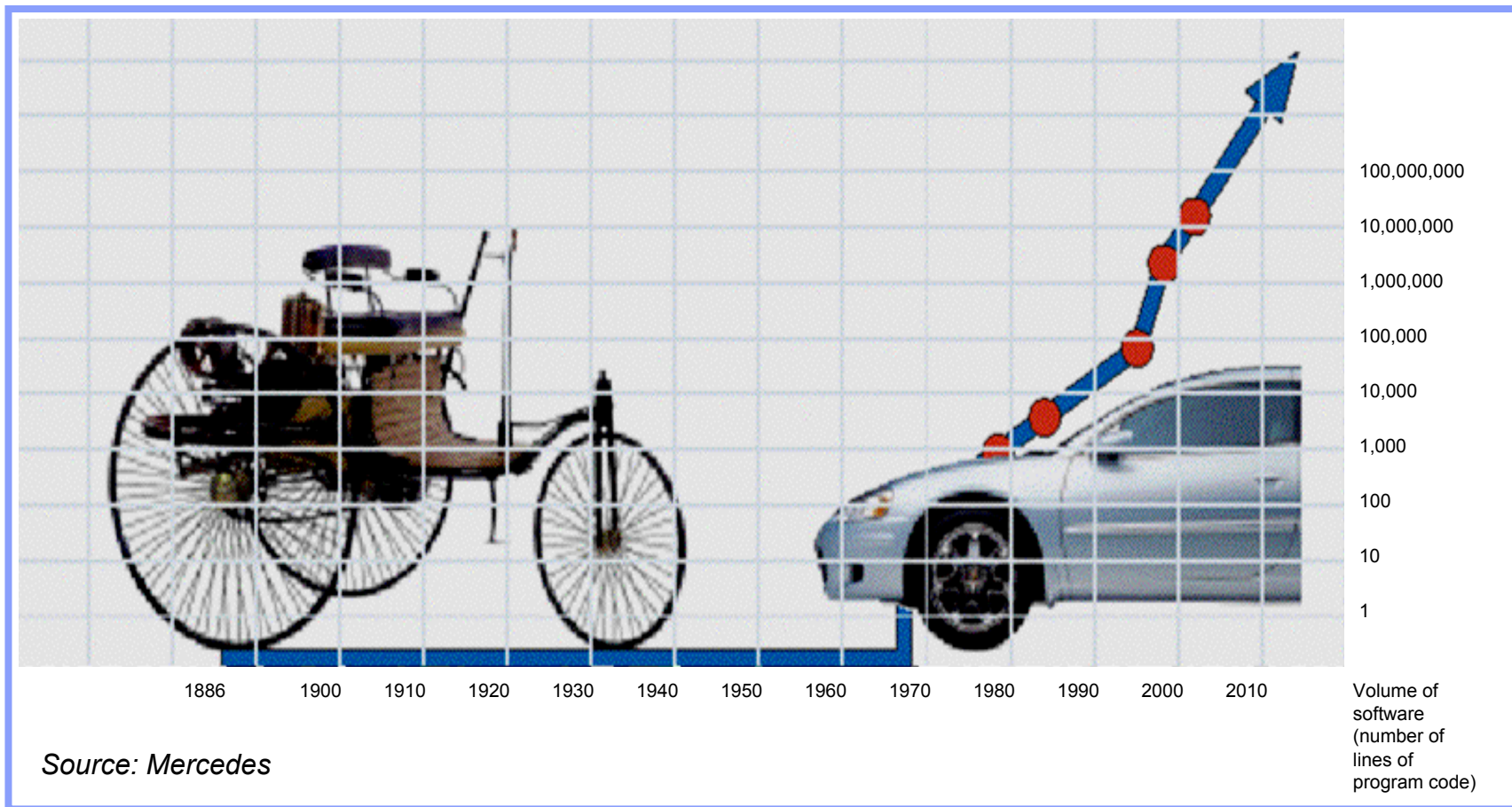
- ▶ Internal quality standards and regulatory pressure

- **Flexibility and responsiveness**

- ▶ Open source, open standards, open architectures



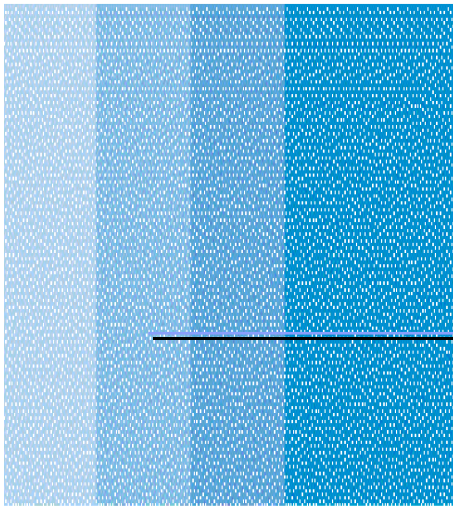
## Efficiency: Software is everywhere



Source: Gartner, April 2003, *Embedded Software Development and Management - Automotive Industry*



## Efficiency: Software is the business



***“It is impossible to separate  
IT and business strategy.  
IT doesn’t support the  
business, it is the business.”***

Asiff Hirji  
CIO Ameritrade

## Efficiency: Improving Software Economics

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

- Complexity** → Volume of human-generated code
- Process** → Methods, notations, maturity
- Team** → Skill set, experience, motivation
- Tools** → Process automation



## Compliance failure has significant, negative cost consequences

- Companies paid of **\$2.4m more for audits** last year than they anticipated.<sup>1</sup>
- Large firms, on average, spent **70,000 additional man-hours** complying with Sarbanes-Oxley.<sup>1</sup>
- 49% of firms surveyed report that **IT issues proved to be a larger part of overall compliance efforts** than companies expected.<sup>2</sup>

<sup>1</sup>The Economist, "Sarbanes Oxley: A Price Worth Paying?", May 19, 2005

<sup>2</sup>CFO Magazine, "Sarbox and IT: How Bad Can Things Get?", June 22, 2005





## Flexibility and responsiveness: Open Computing

### Open standards:

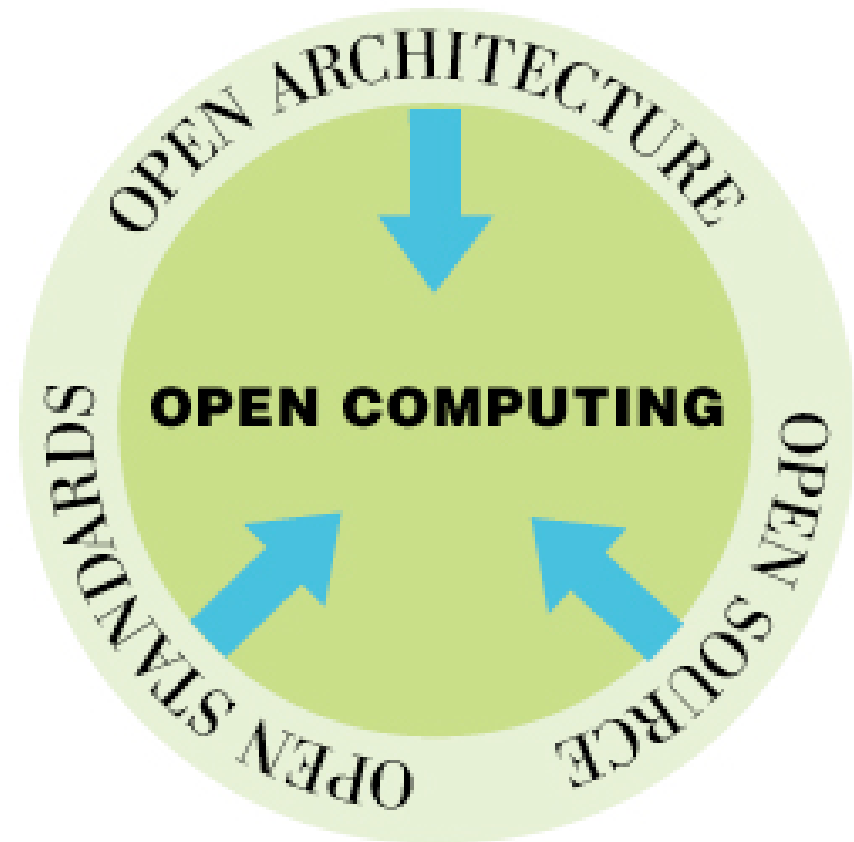
- Promoting interoperability by using open published specifications for API's, protocols and data and file formats

### Open architecture:

- Building loosely coupled, flexible, reconfigurable solutions
- Coupling business processes to software capabilities

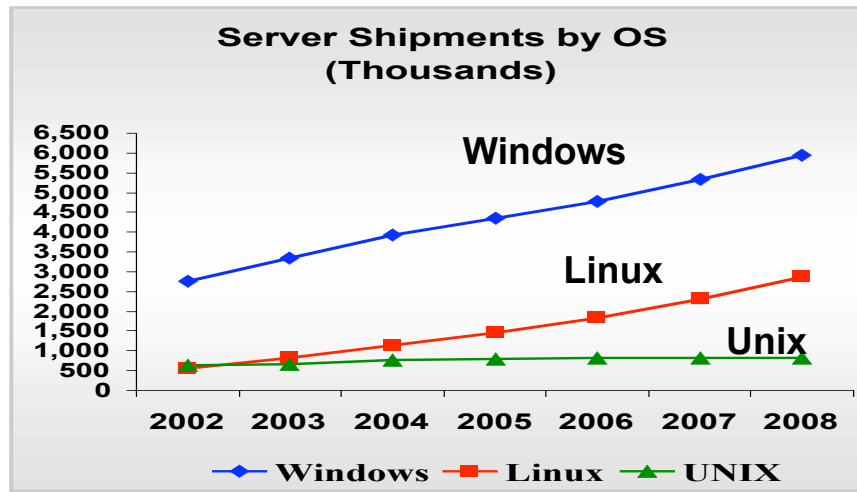
### Open source software:

- Promotes standards
- Leverages community development and collaborative innovation



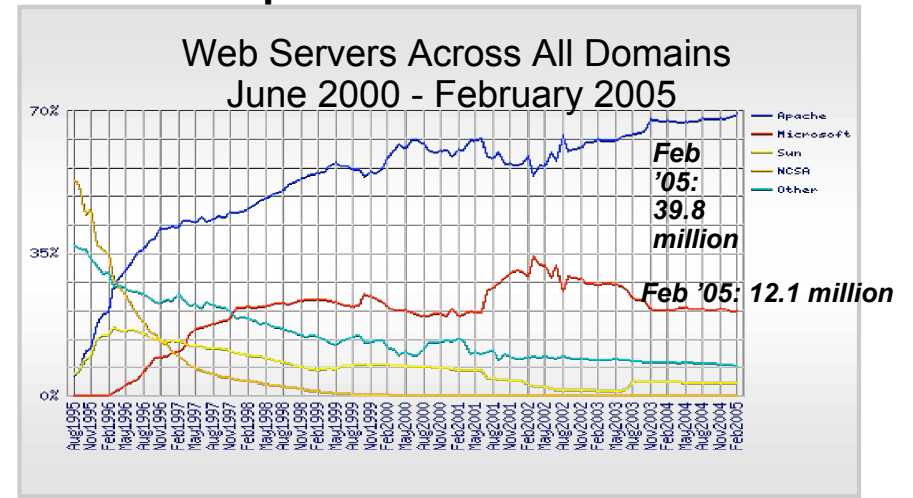
# Flexibility and responsiveness: Open Source Usage Growth

## Linux



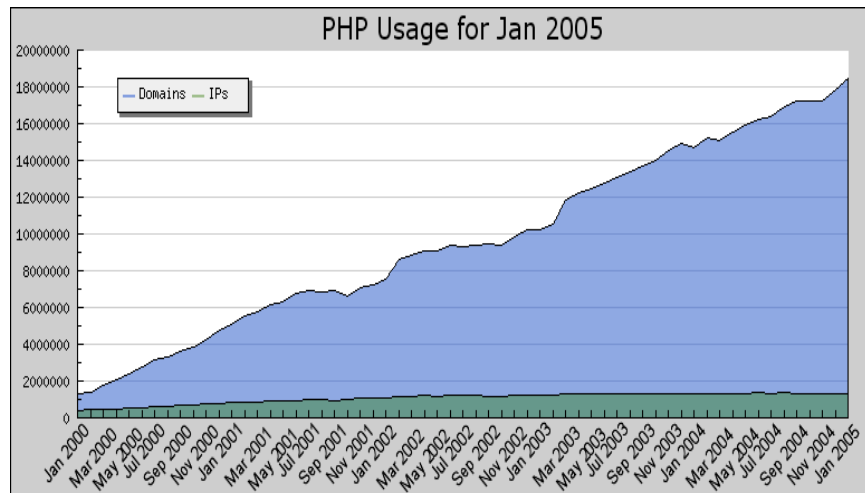
Source: IDC Server Market Quarterly Forecaster, 1Q04 2004

## Apache Web Server



Source: Netcraft Web Server Study, Feb. 2005

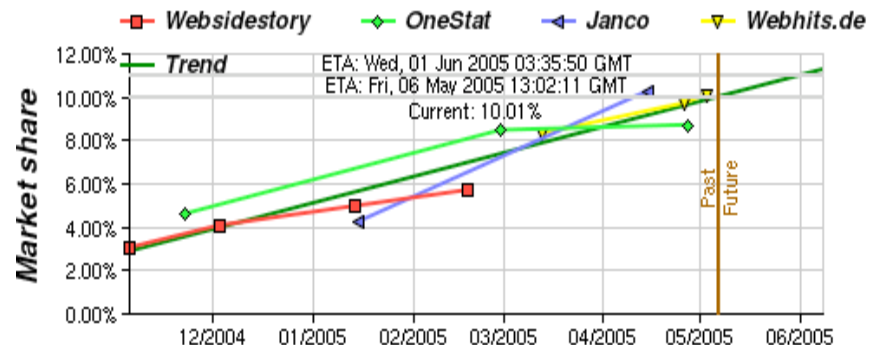
## PHP



Source: Netcraft

## Firefox

### Firefox market share



Source: <http://ff.asbjorn.it>

## Flexibility and responsiveness

***“Open Source is not about Free. It's about Freedom. The freedom to collaborate. The freedom to innovate.”***

*“Open source gives more people access to the building blocks of innovation, enabling diverse perspectives and influences to be integrated into the creative process.”*



Nick Donofrio  
Sr. VP, Technology & Manufacturing, IBM  
At LinuxWorld August, 2004



## Trends Impacting Software Development: Conclusion

- Efficiency - business pressure, global pressure
  - ▶ Business, Development and Operations needs to be tightly integrated
  - ▶ Lack of quality has an immediate business impact
  - ▶ Improve software economics across all 4 dimensions; Complexity, Process, Teams, Tools
- Compliance
  - ▶ Increased pressure to make best practice daily practice
  - ▶ Minimize overhead for teams while ensuring regulatory compliance
- Innovation and flexibility through open computing
  - ▶ Open standards for interoperability
  - ▶ Open architectures and SOA for flexibility and responsiveness
  - ▶ Open source software to drive innovation and transparency



# Agenda

- Trends impacting software development
- **Key principles for business-driven development**
- An End-to-End Solution for SOA
- Summary and conclusions



## A way to transform and simplify development

### Business-driven development

An integrated approach to software development that aligns line-of-business, development and operations teams to improve business performance



### *Development as a business process*

- Align Technology and Business priorities
- Improve efficiency and responsiveness
- Create innovative products

**Software development  
becomes a driver  
of competitive advantage**



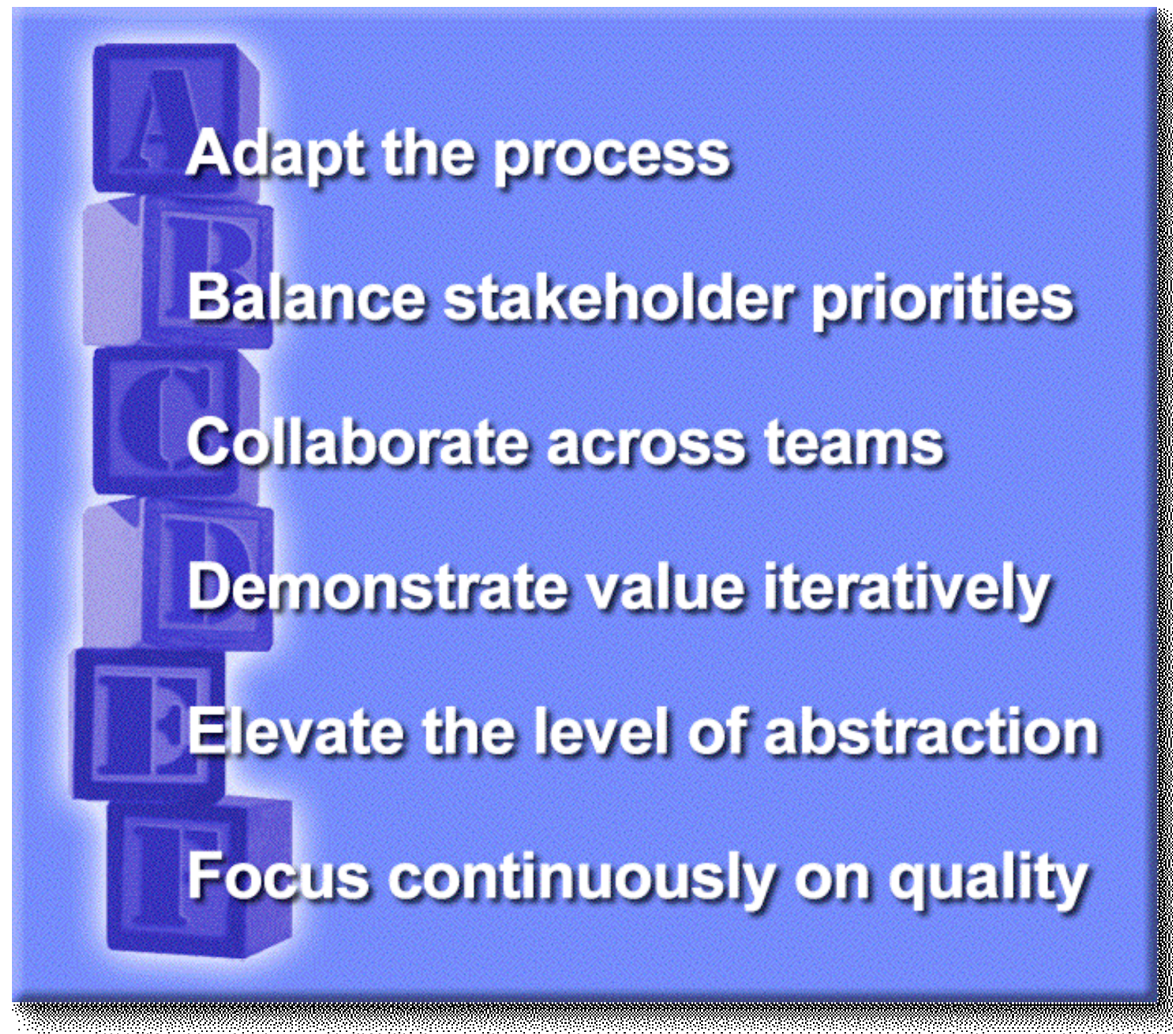
# Defining Principles for Business-Driven Development

- Input
  - ▶ Current 6 best practices for IBM Rational
  - ▶ Ongoing workshops with +1,000 development managers and executives in 1999-2005
  - ▶ Input from +80 key technical leaders within IBM
- How did we develop these principles?
  - ▶ 10 months of collaborative effort with various workgroups
  - ▶ Many rounds of feedback from technical communities within IBM
- Guiding focus
  - ▶ Based on real world experiences
  - ▶ Applies to a broad set of contexts
  - ▶ Time resistant





## Key Principles for Business-Driven Development





## Principle: Adapt the Process

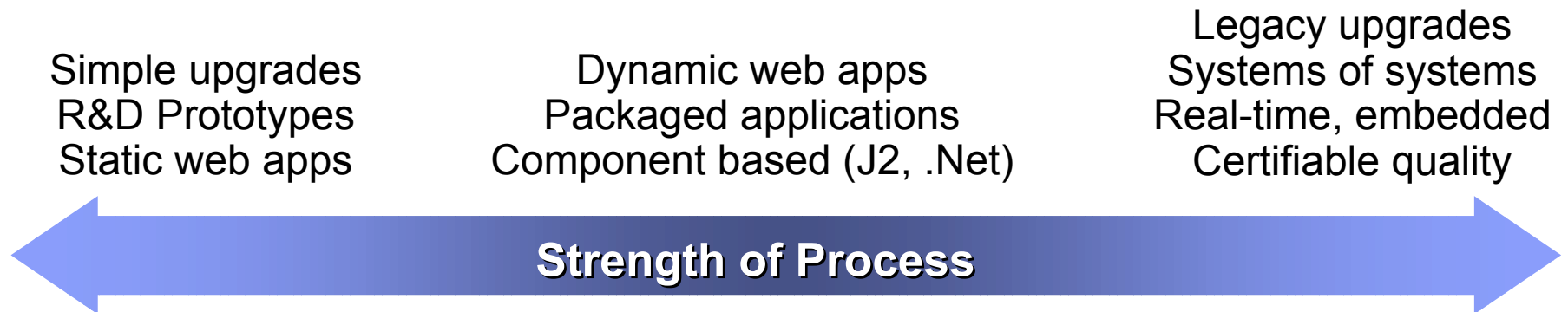
- Benefits
  - ▶ Lifecycle efficiency, open/honest communication of risks
- Pattern
  - ▶ Adapt the process to the size and distribution of the project team, to the complexity of the application, and to the need for compliance
  - ▶ Precision and formality evolve from light to heavy over the project lifecycle as uncertainties are resolved
  - ▶ Improve your process continuously



- Anti-patterns
  - ▶ More process is better
  - ▶ Always using the same amount of process throughout the lifecycle



## How Much Process is Necessary?



### When is Less Appropriate?

- Co-located teams
- Smaller, simpler projects
- Few stakeholders
- *Early life-cycle phases*
- Internally imposed constraints

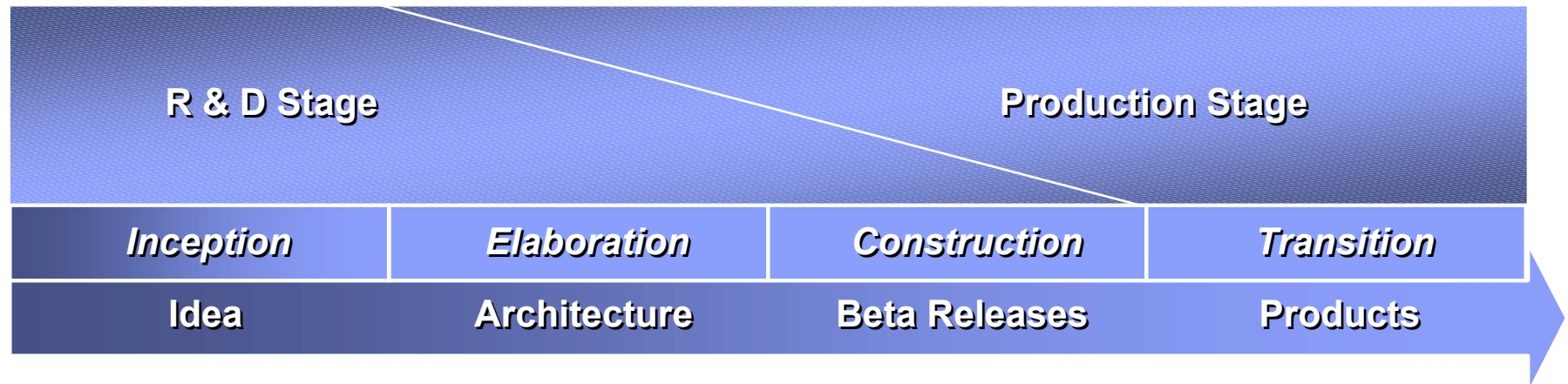
### When is More Appropriate?

- Distributed teams
- Large projects (teams of teams)
- Many stakeholders
- *Later life-cycle phases*
- Externally imposed constraints
  - Standards
  - Contractual requirements
  - Legal requirements




## Process Evolution Over the Life Cycle

- Prototypes
- Major risk items
- Creative, judgment
- Maneuverable processes
- Change managed baselines
- Low risk Items
- Engineering, reasoned
- Well-instrumented processes



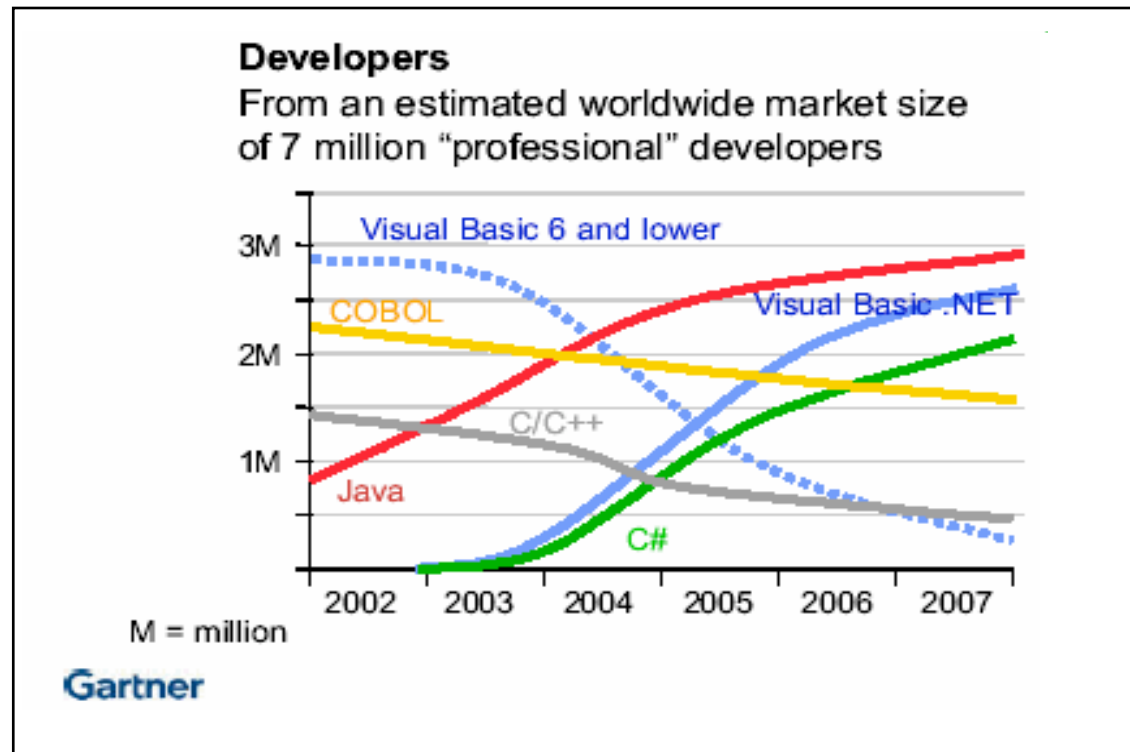
## Principle: Balance Stakeholder Priorities

- Benefit
  - ▶ Align applications with business and user needs
  - ▶ Reduce custom development, and optimize business value
- Pattern
  - ▶ Understand what assets you can leverage; and balance user needs and reuse of assets
  - ▶ Define and prioritize business processes and user needs, and couple user needs to software capabilities
  - ▶ Center development activities around user needs
-  ■ Anti-pattern
  - ▶ Requirements focus drives towards custom solution
  - ▶ Achieve precise and thorough requirements before any project work begins



# Decades of Existing Assets Must be Leveraged

*Rewriting all existing applications and moving them to new platforms is not a viable option*

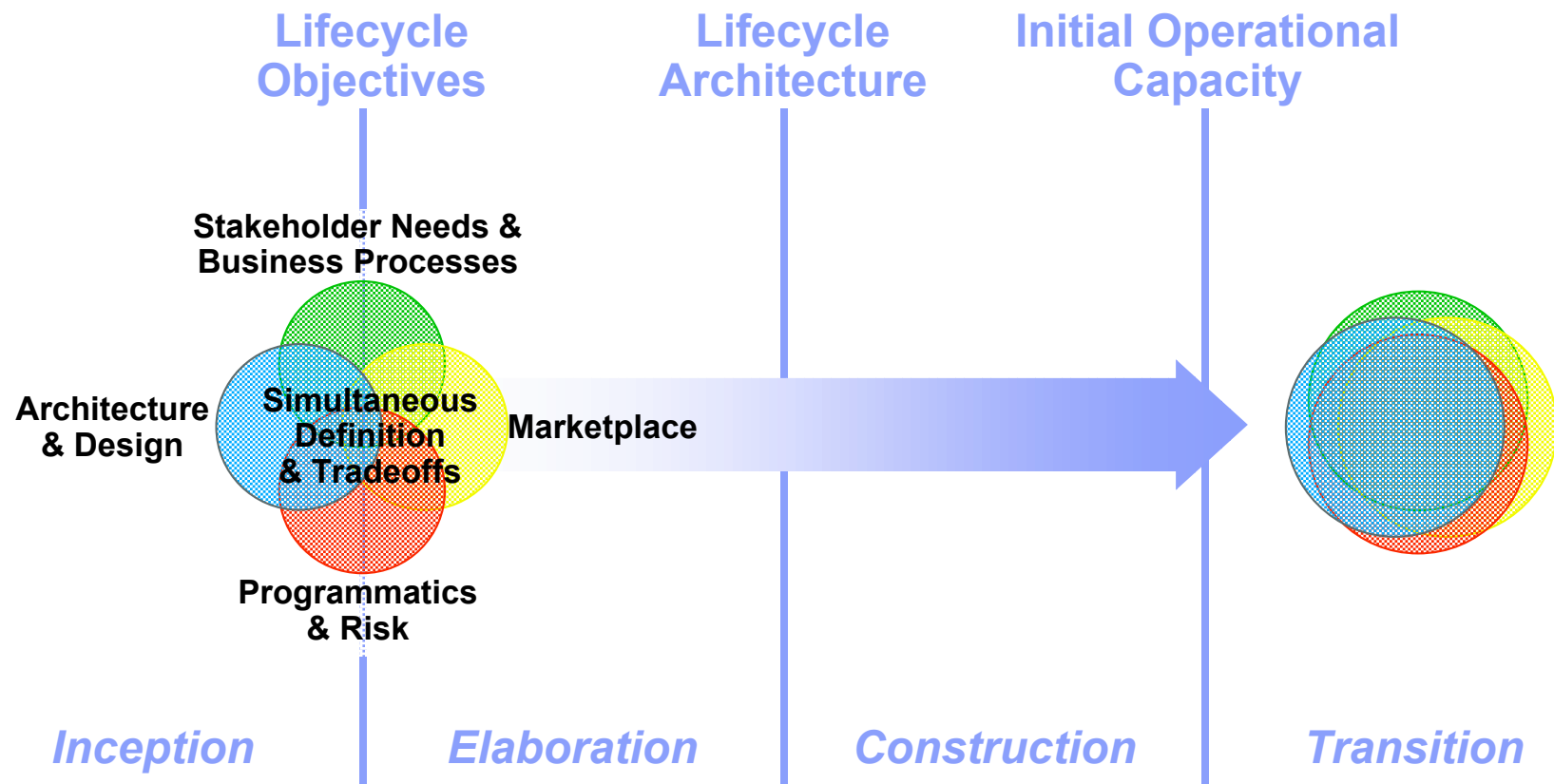


- ★ New code **cost 5X** than **reusing** existing code, *Software Productivity Research (SPR)*
- ★ **200 Billion lines of COBOL** code in existence, *eWeek*
- ★ **5 Billion lines of COBOL** code added yearly, *Bill Ulrich, TSG Inc.*
- ★ Between **850K and 1.3 Million COBOL** developers with **12,000 per year attrition**, *IDC*



## Balance User Needs and Reuse of Assets

*Delivering the right application to the right price is a continuous balance act between sometimes competing priorities*



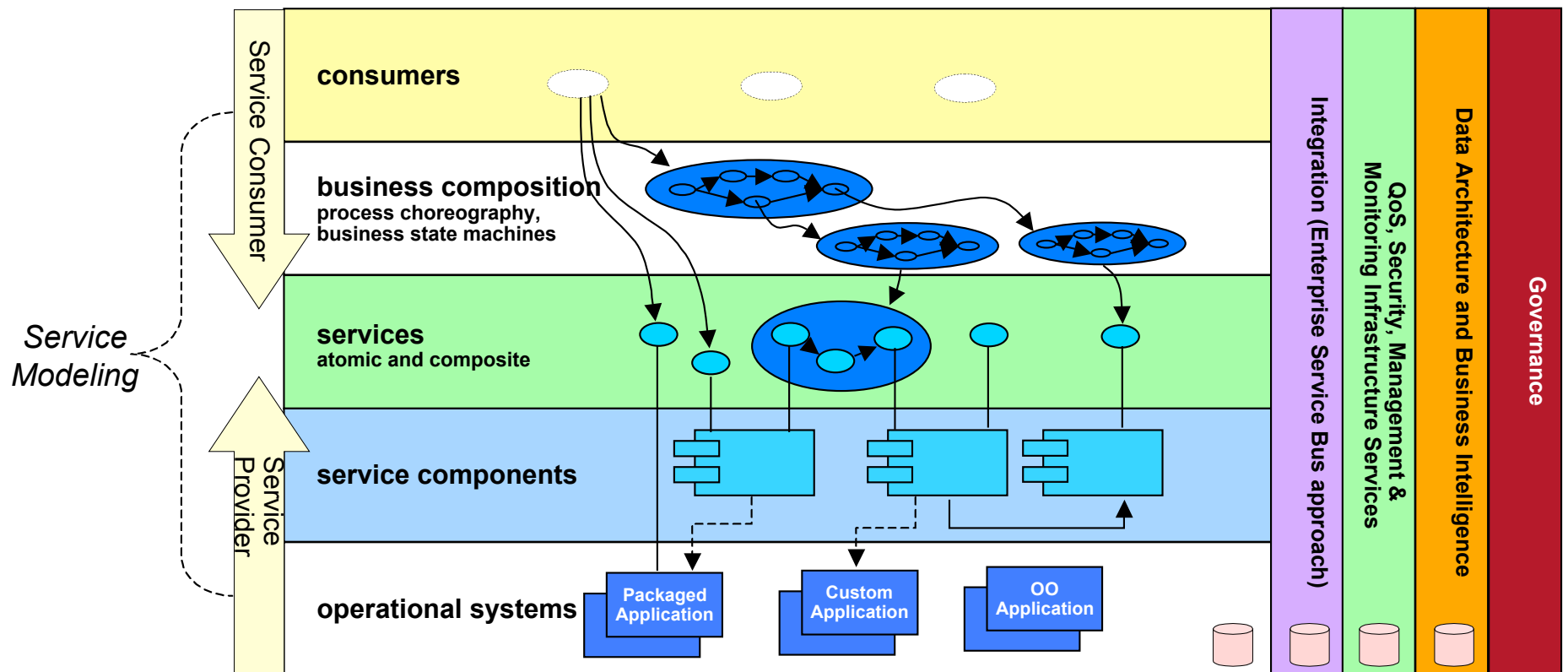
Source: SEI EPIC



# Service-Oriented Architecture – Business Processes to Software Capabilities

An SOA is composed of multiple layers.

At the heart of the SOA are services, components that realize services, and service flows.



## Principle: Collaborate Across Teams

- Benefits
  - ▶ Team productivity, fewer meetings
  - ▶ Better coupling between business needs, and the development and operations of software systems
- Pattern
  - ▶ Motivate people to perform at their best
  - ▶ Encourage cross-functional collaboration
  - ▶ Provide effective collaborative environments
  - ▶ Integrate business, software, and operation teams
- Anti-pattern
  - ▶ Nurture heroic individuals and arm them with power tools



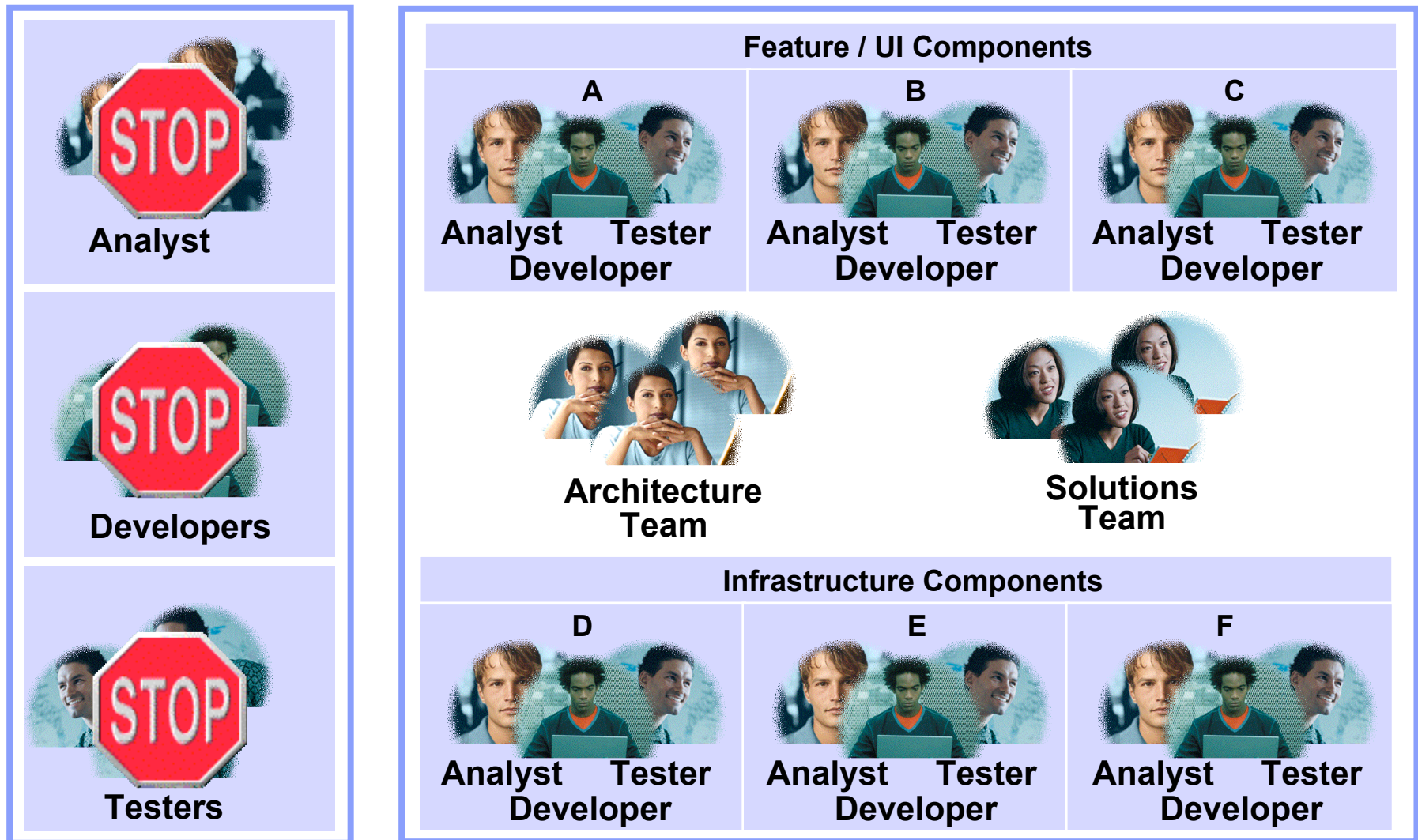


## Motivate people to perform at their best

- Make sure the team understand and buys in to the vision and mission
- Make the team responsible for the end result
- Reconcile top down estimates with bottom up estimates
- Build skills to enable the team
- Self-managed teams when appropriate
- High-trust environment
- Provide a supporting environment that enables people to succeed

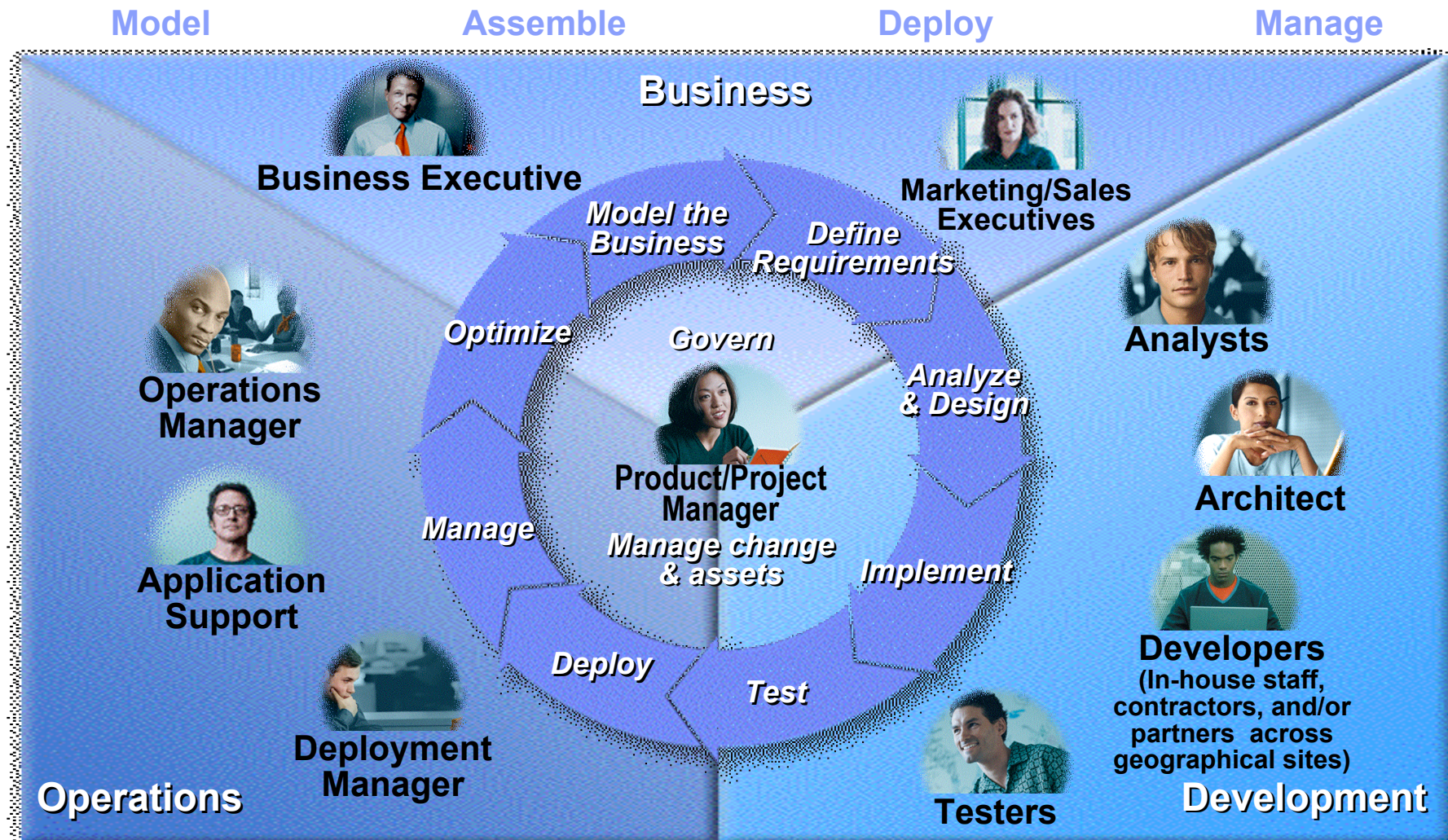


# Encourage cross-functional collaboration





# Integrate business, software, and operation teams





## Multiple Languages = Communication Barriers




**Business Analyst**

**Requirements  
and  
Business Models**




**HTML  
CGI  
XML  
JavaScript**

**Web Content  
Creator**



**Software Engineer**

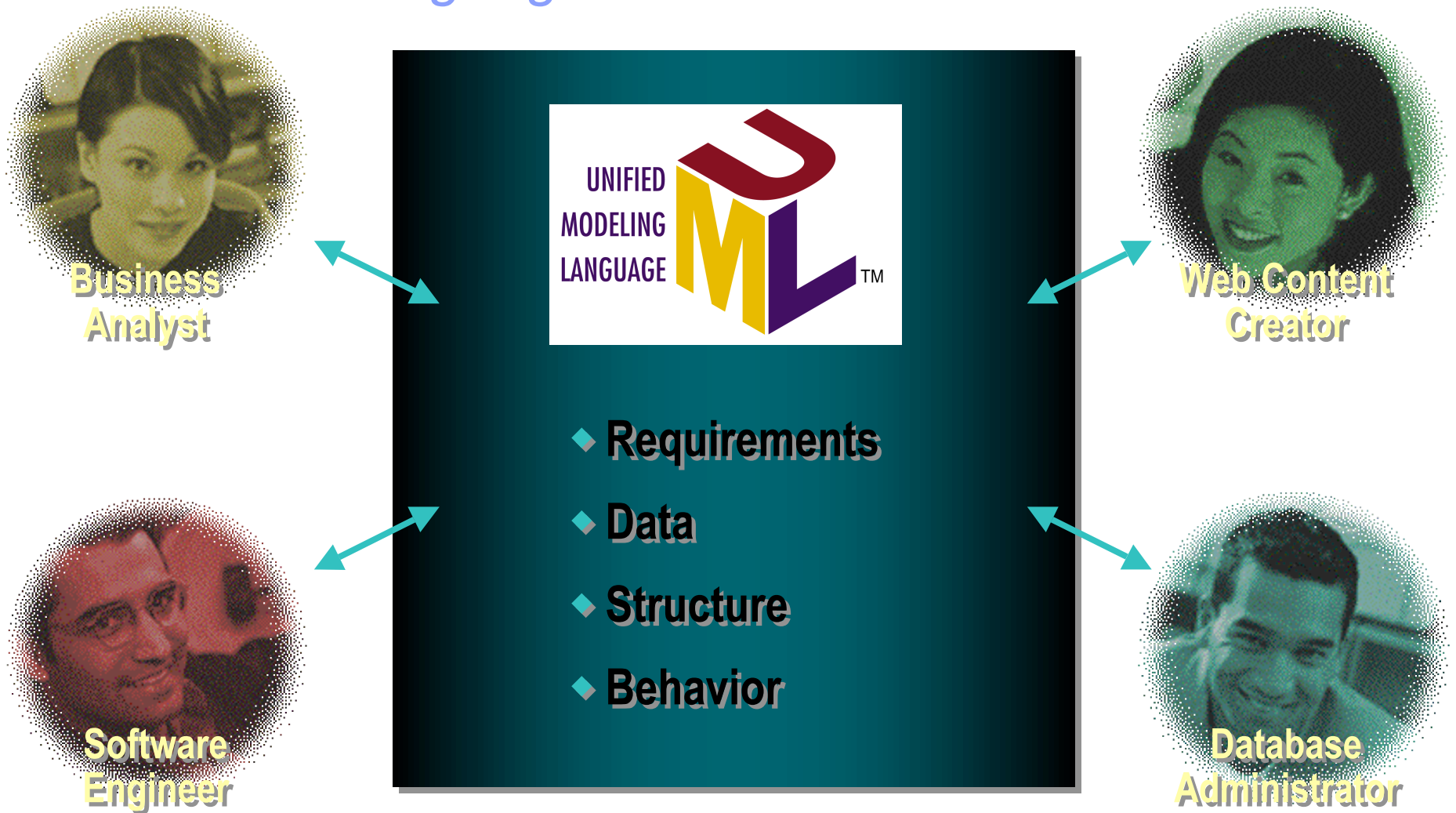
**C++  
Java  
SW Models**



**SQL  
ER Models**

**Database Administrator**

## UML: *One Language for All Practitioners*



## Principle: Demonstrate Value Iteratively

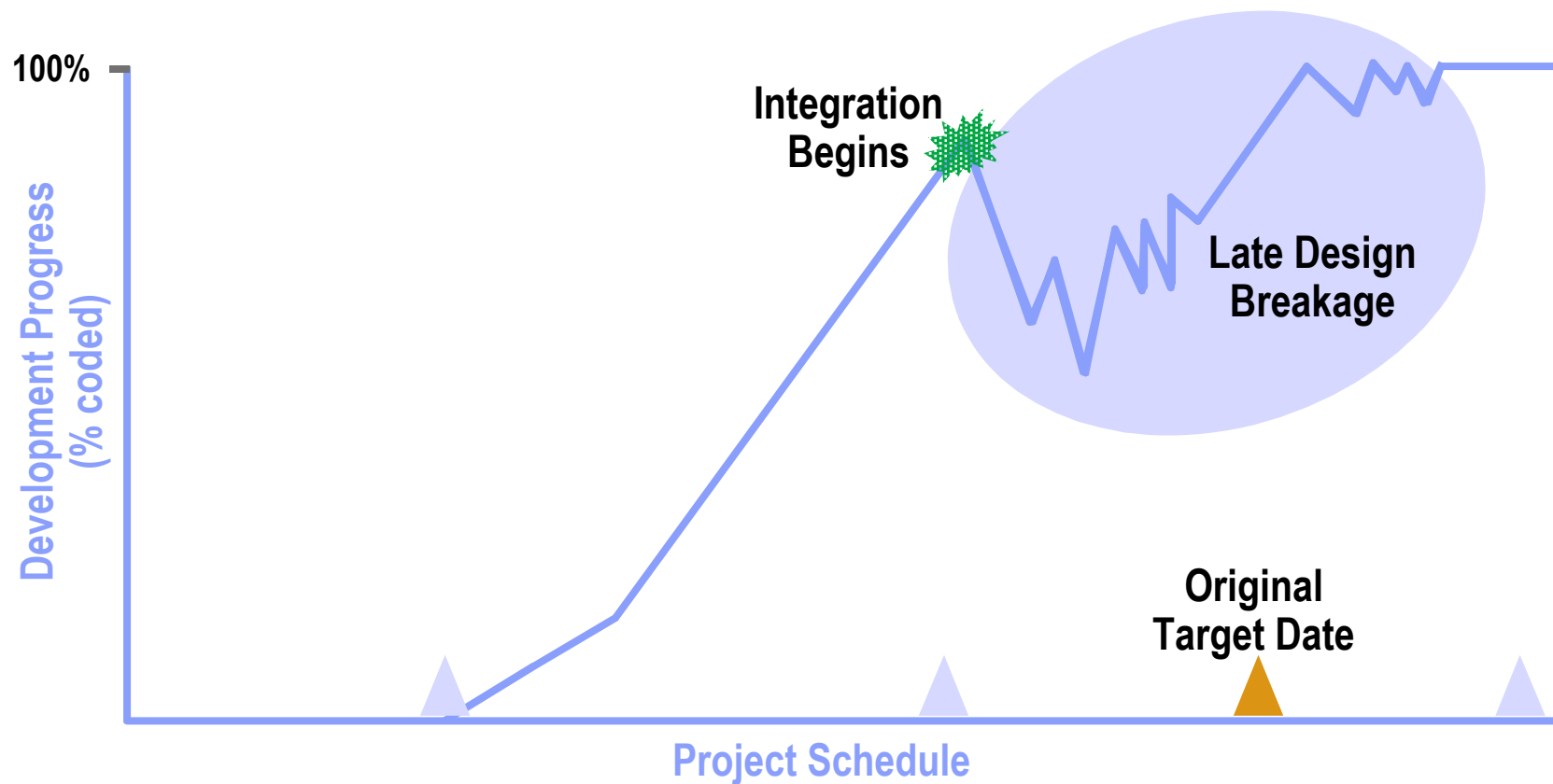
- Benefits
  - ▶ Early risk reduction
  - ▶ Higher predictability
  - ▶ Trust among stakeholders
- Pattern
  - ▶ Attack major technical, business and programmatic risks first
  - ▶ Enable feedback by delivering incremental user value in each iteration
  - ▶ Demonstrations provide more tangible insight into progress/quality
  - ▶ Embrace and manage change
  - ▶ Adaptive management using an iterative process
- Anti-pattern
  - ▶ Plan the whole lifecycle in detail, track variances against plan
  - ▶ Less reliance on expensive and error prone human inspection
  - ▶ Assess status by reviewing specifications



# Waterfall Development - What Happens in Practice

*Sequential activities:*

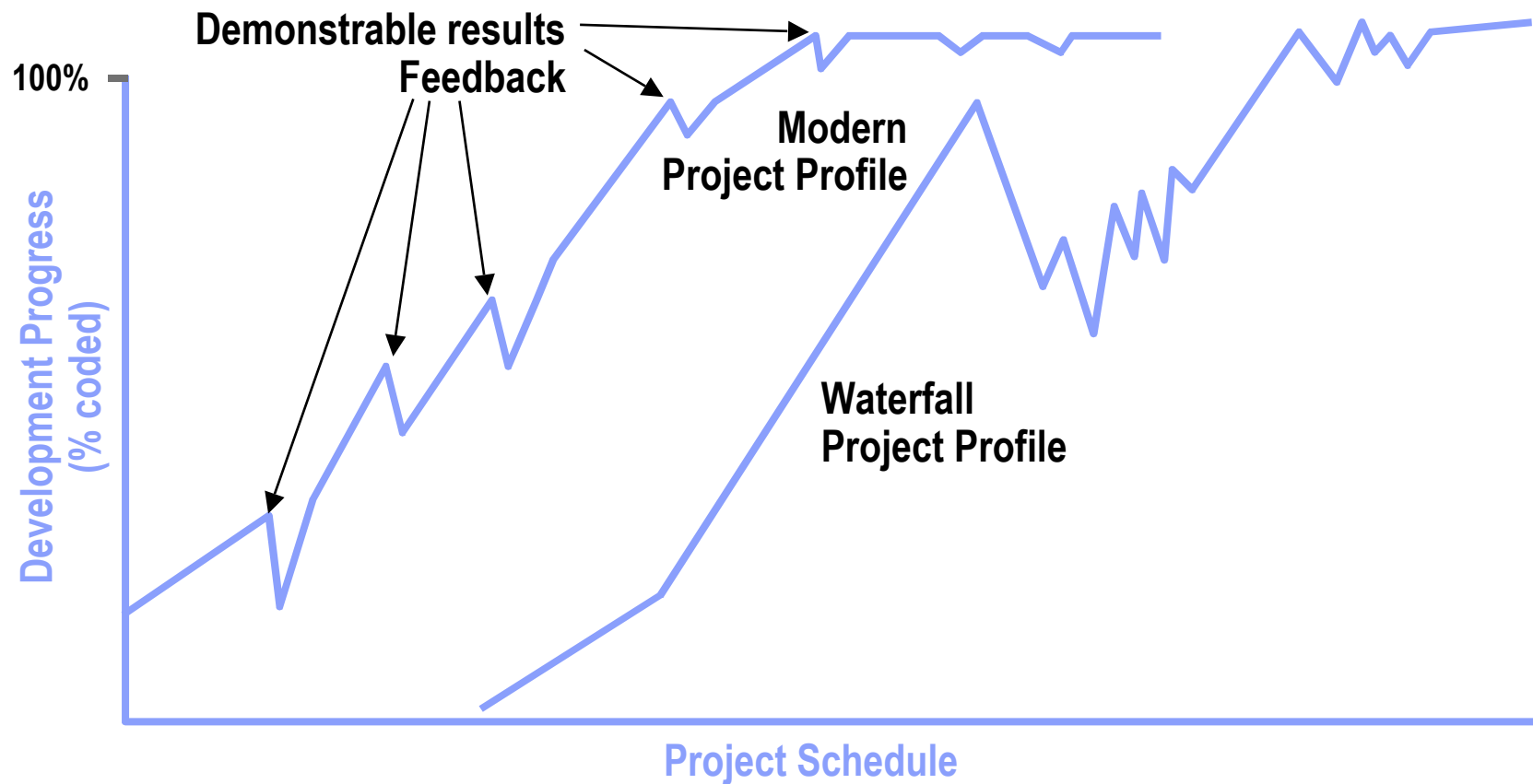
Requirements ➡ Design ➡ Code ➡ Integration ➡ Test



## Better Progress Profile

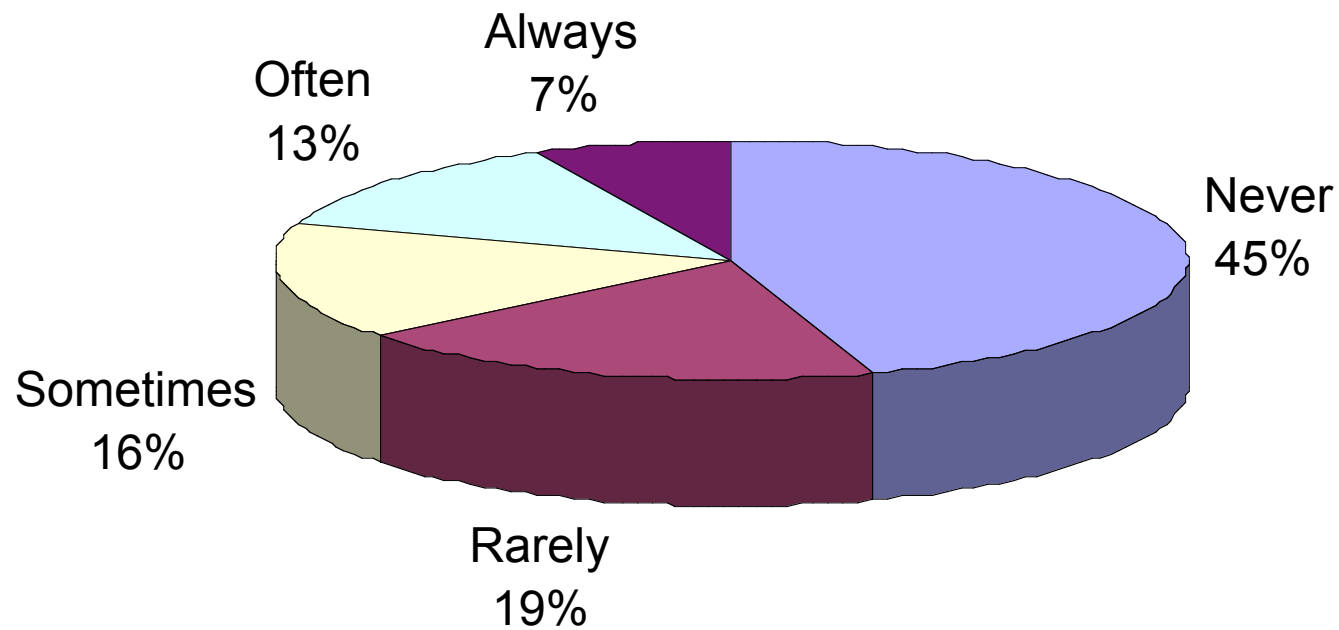
Sequential phases, but iterative activities

Prototypes → Architecture → Functional Releases → Product Release





## Feature and Function Usage



(Standish Group's Chaos Report 2003)




## Embrace and Manage Change

- Embrace change
  - ▶ If we do not change, we build the wrong solution
  - ▶ Ability to efficiently incorporate changes throughout lifecycle provides a competitive advantage
- Manage change
  - ▶ If you never stop changing, you will never complete the project
- The ability to do life-cycle changes depend on
  - ▶ your process
  - ▶ your life-cycle tooling
  - ▶ the complexity of your project / organization
  - ▶ the impact of delivering with defects
- Iterative development and other agile practices aim at maximizing change freedom



## Principle: Elevate the Level of Abstraction

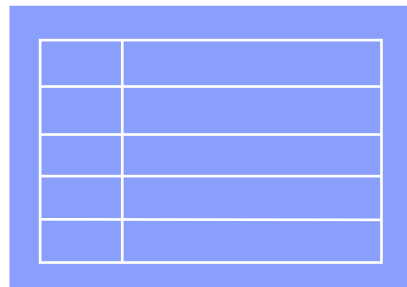
- Benefits
  - ▶ Productivity
  - ▶ Reduced complexity
- Pattern
  - ▶ Plan with evolving levels of details
  - ▶ Reuse existing assets
  - ▶ Reduce the amount of human generated stuff through higher-level tools and languages
  - ▶ Architect for resilience, quality, understandability, and complexity control (Focus on architecture first)
-  ■ Anti-pattern
  - ▶ Go directly from vague high-level requirements to custom-crafted code



## Plan with Evolving Levels of Details

### Coarse-grained Plan: *Project Plan*

One For Entire Project



#### Phases and major milestones

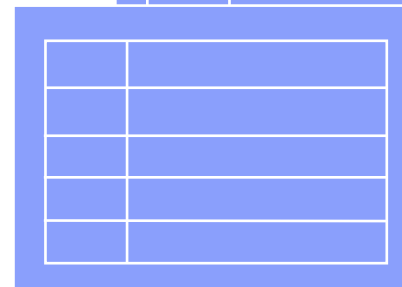
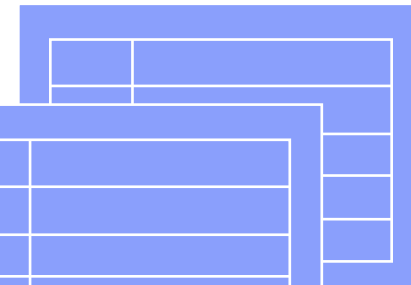
- What and when

#### Iterations for each phase

- Number of iterations
- Objectives and Duration

### Fine-grained Plans: *Iteration Plans*

Next Iteration



Current Iteration

#### To-Do List / Gantt chart

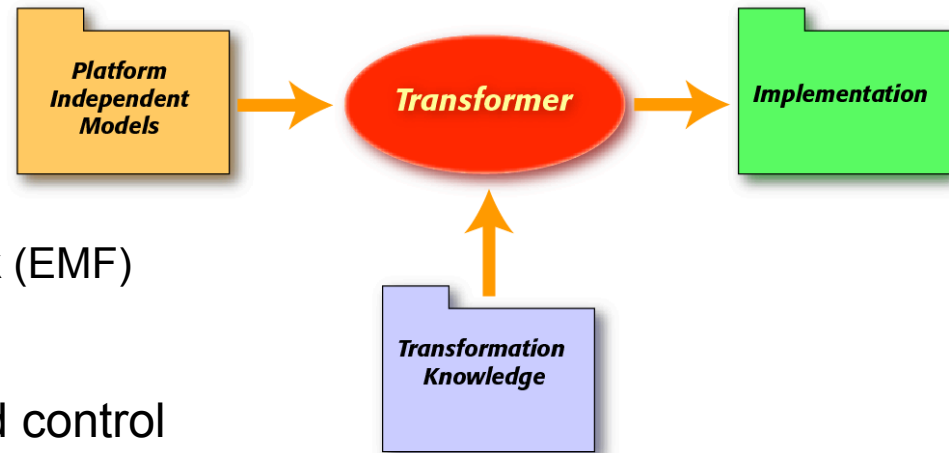
- Who does what when



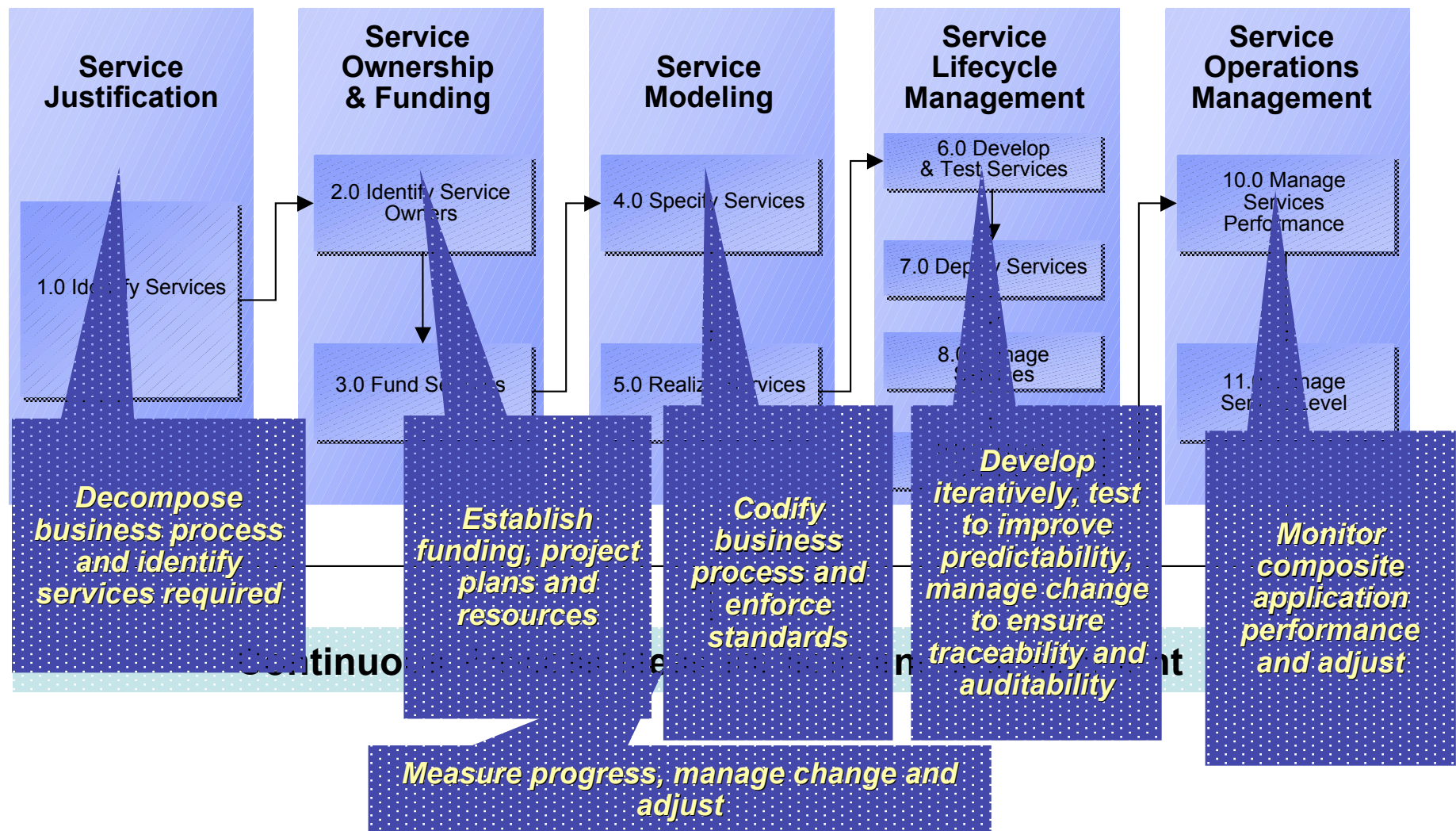
# Reduce the Amount of Human Generated Stuff

## Model-driven architecture

- Model-to-model transformations
- Model-to-code transformations
  - ▶ Compare Eclipse Modeling Framework (EMF)
- Visualization
- Architectural analysis, discovery, and control
- Assets, patterns, templates
  - ▶ Parameterized asset
- Recipes
  - ▶ Composite assets aiding effective usage of an asset
- Solutions guide
  - ▶ Assets, guidance, wizards or cheat sheets



# Transforming to an SOA environment



# Why Will SOA Change the Industry?

## Standards

- Broadly adopted Web services ensure well-defined interfaces
- Before, proprietary standards limited interoperability

*"We are taking apart each task and sending it... to whomever can do it best, ...and then we are reassembling all the pieces."*

*from Thomas Friedman's 'The World is Flat'*

## Organizational Commitment

- SOA unites Business and IT (66% of projects today are driven by line of business)
- Before, IT alone defined the design

## Degree of Focus

- SOA services focus on business-level activities & interactions
- Before, focus was on narrow, technical sub-tasks

## Connections

- SOA services are linked dynamically and flexibly
- Before, service interactions were hard-coded and dependent on the application

## Level of Reuse

- SOA services can be extensively re-used to leverage existing IT assets
- Before, any reuse was within silo'd applications

\*Source: Cutter Benchmark Survey



## Focus Continuously on Quality

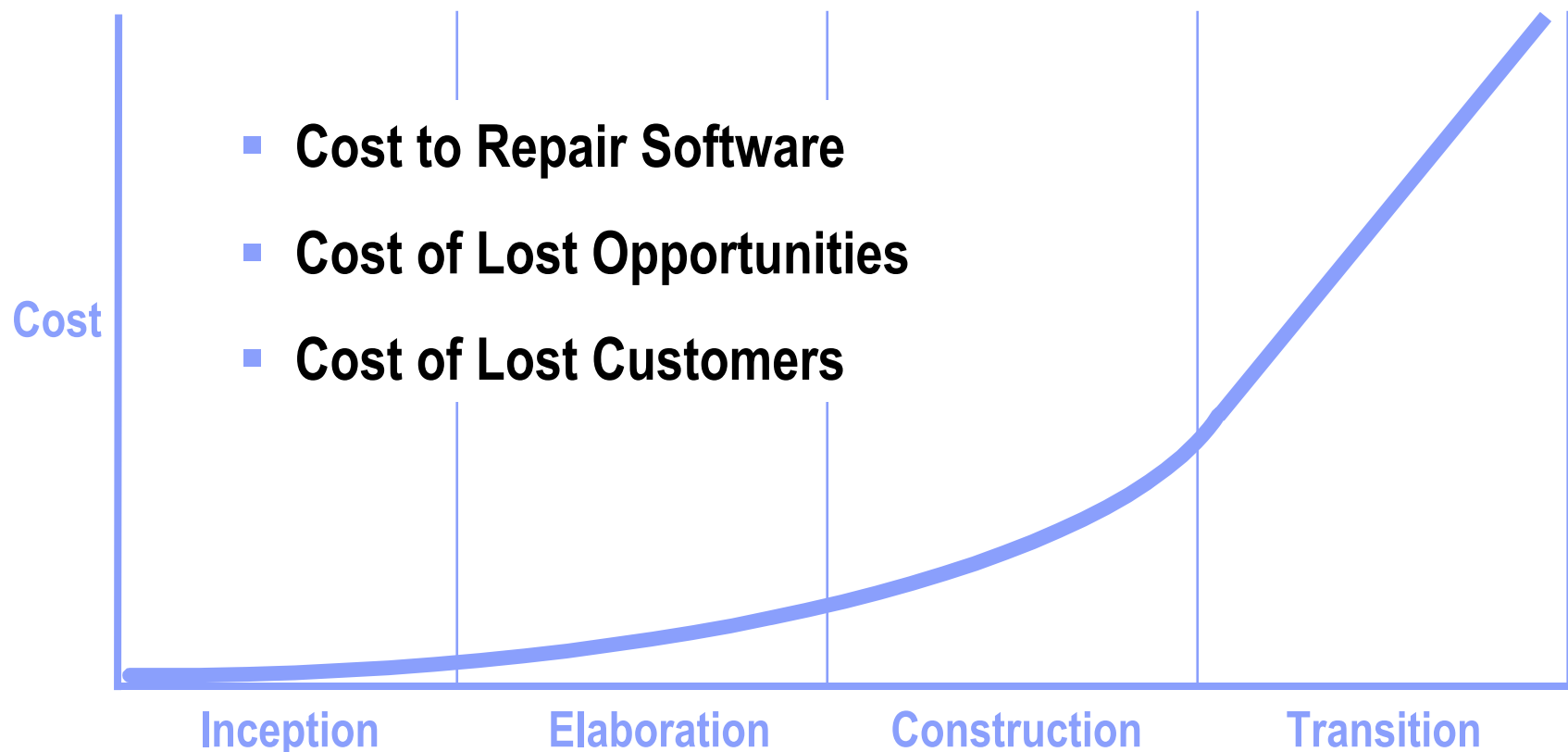
- Benefits
  - ▶ Higher quality and earlier progress/quality insight
- Pattern
  - ▶ Team responsibility for end product
  - ▶ Test early and continuously
  - ▶ Incrementally build test automation
- Anti-pattern
  - ▶ Peer-review all artifacts, rather than also driving partial implementation and testing to discover issues
  - ▶ Complete and unit test all code before integration testing
  - ▶ System level behaviors tested late in the lifecycle





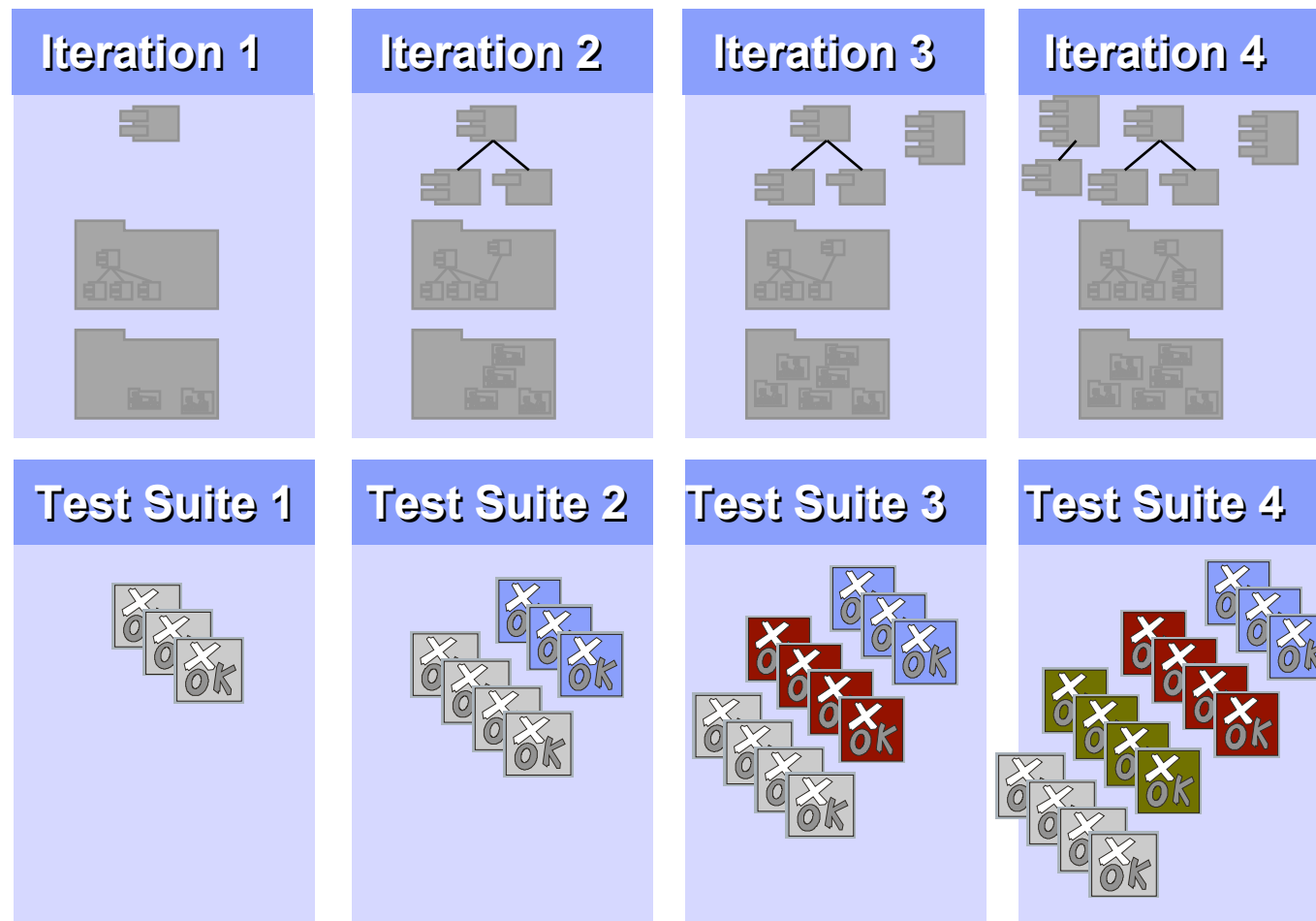
## Make Quality a Way of Life, Not an Afterthought

*Software problems are 100 to 1000 times more costly to find and repair after deployment*



# Incrementally Build Test Automation

**UML Model  
and  
Implementation**

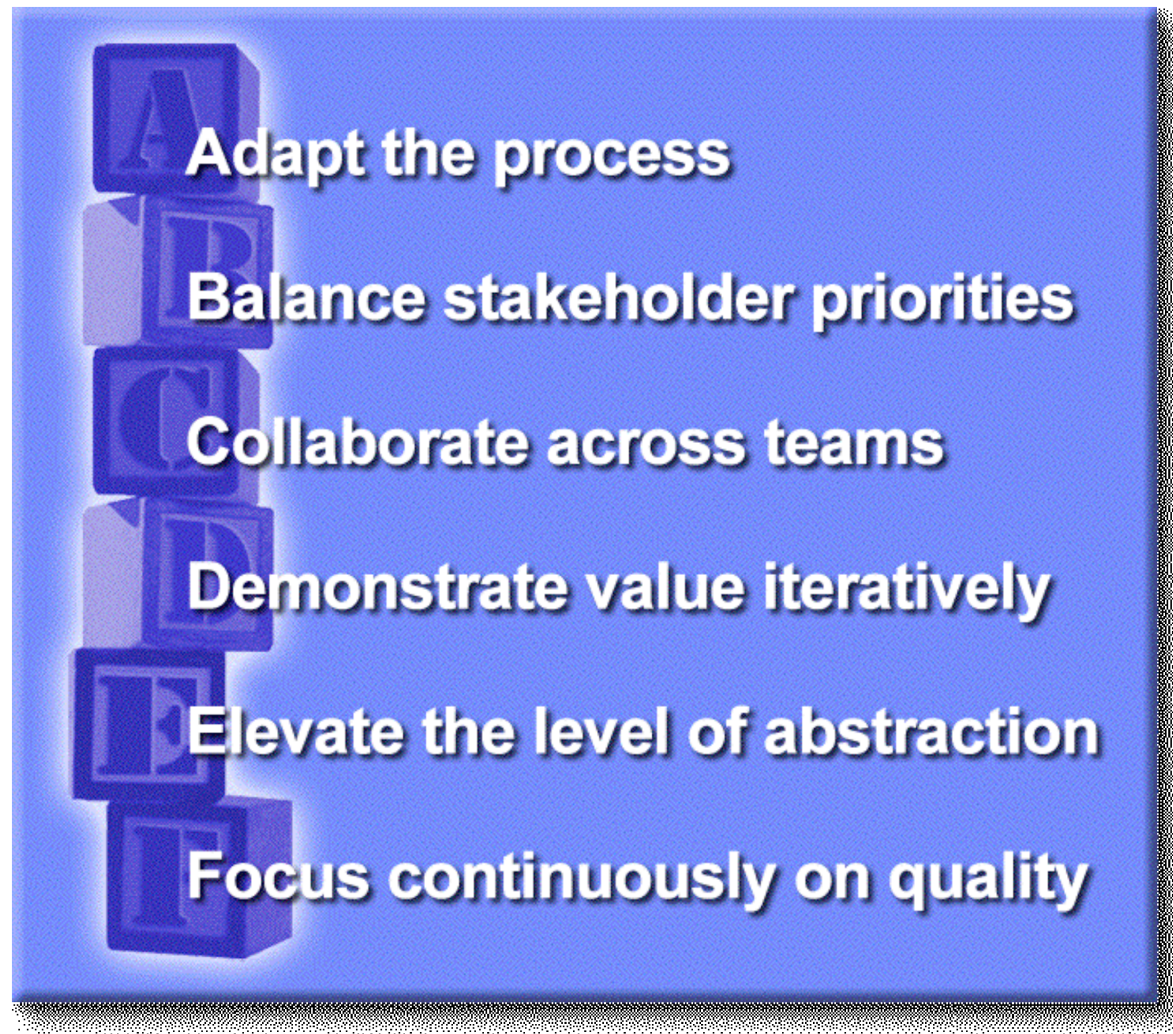


**Tests**

It is often cheaper to find a problem through early implementation and testing, than through detailed design review.



## Key Principles for Business-Driven Development

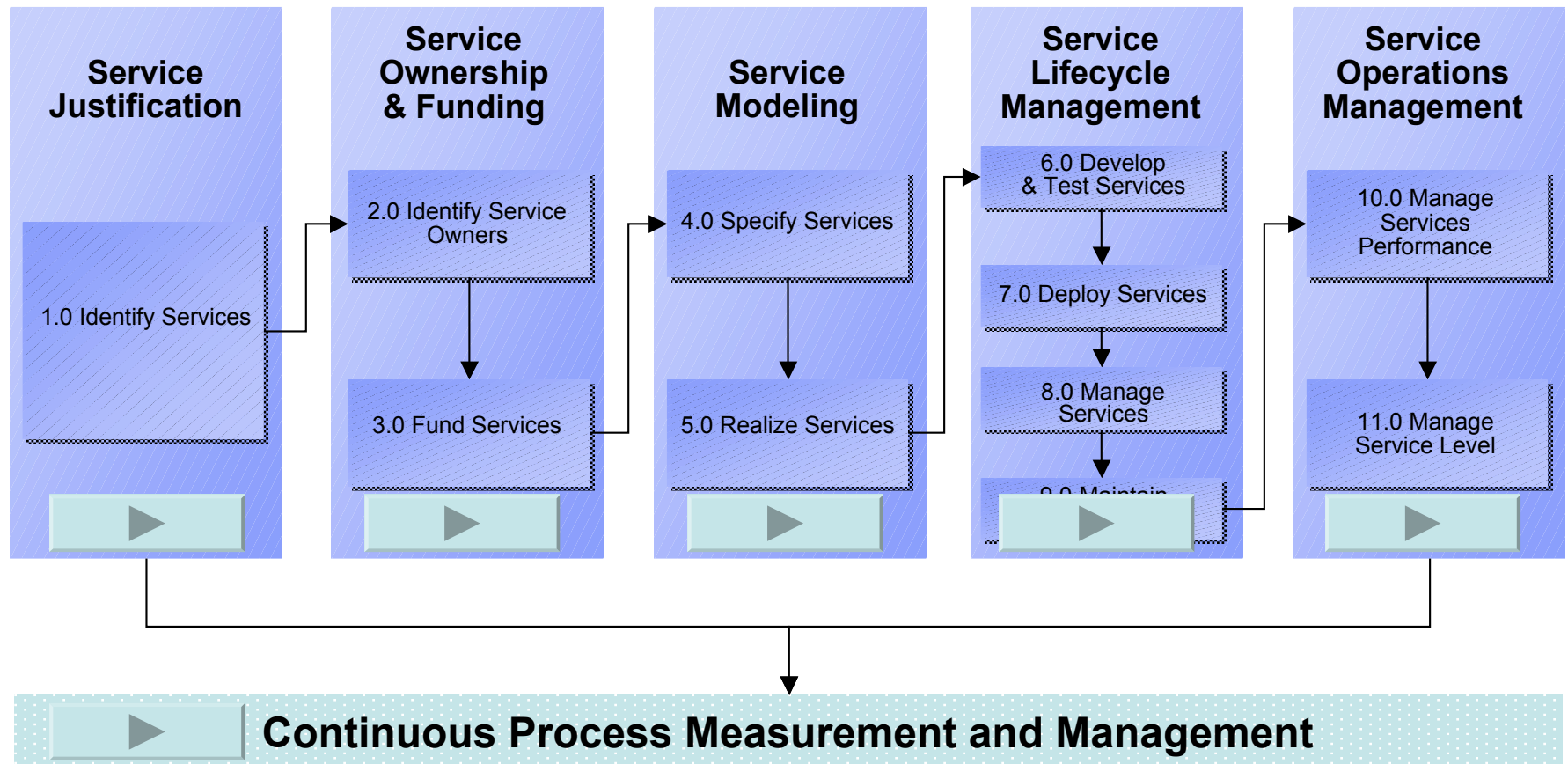


# Agenda

- Trends impacting software development
- Key principles for business-driven development
- **An End-to-End Solution for SOA**
- Summary and conclusions

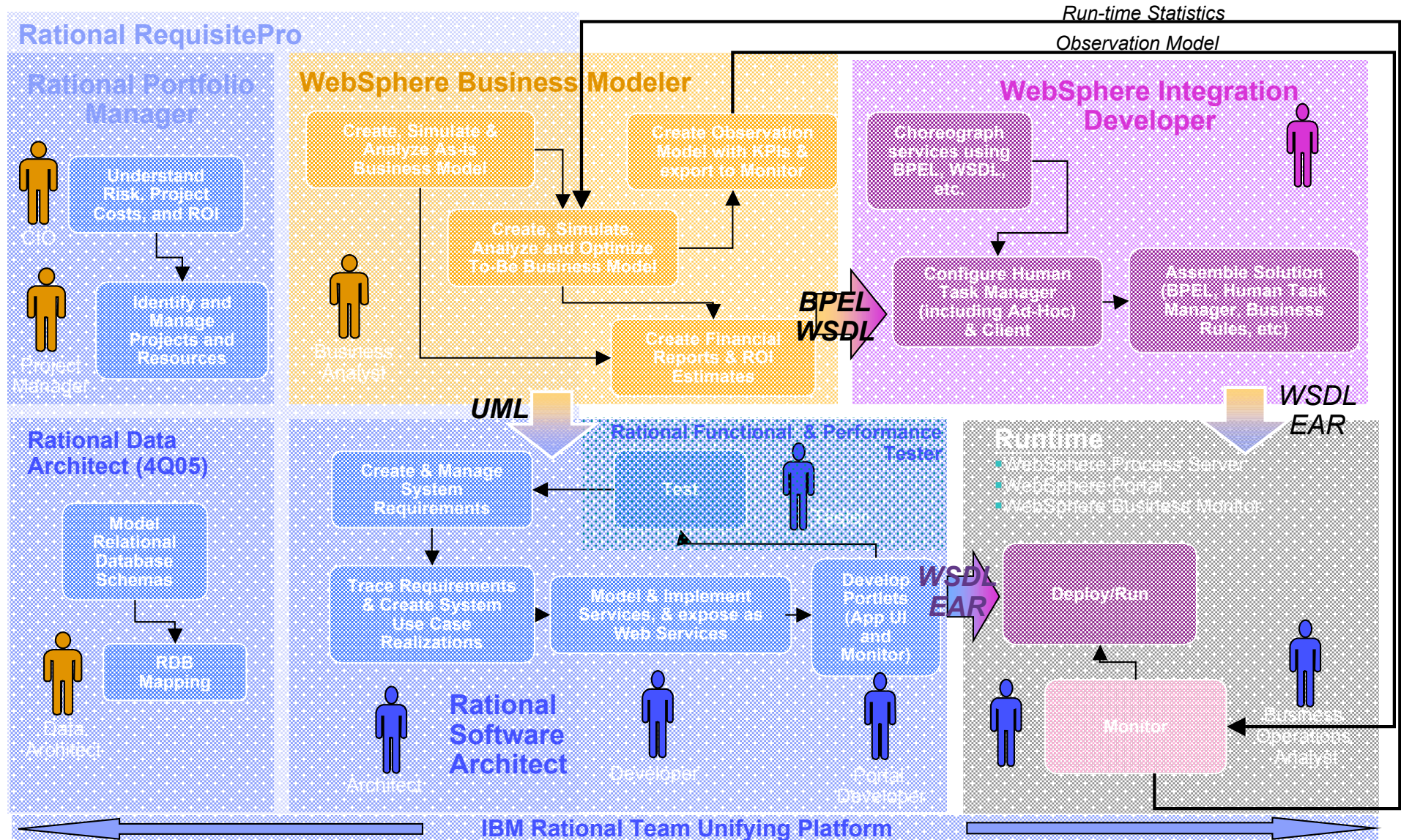


# How Do IBM Rational Technologies Support SOA





# Business Driven Development in the Larger Context

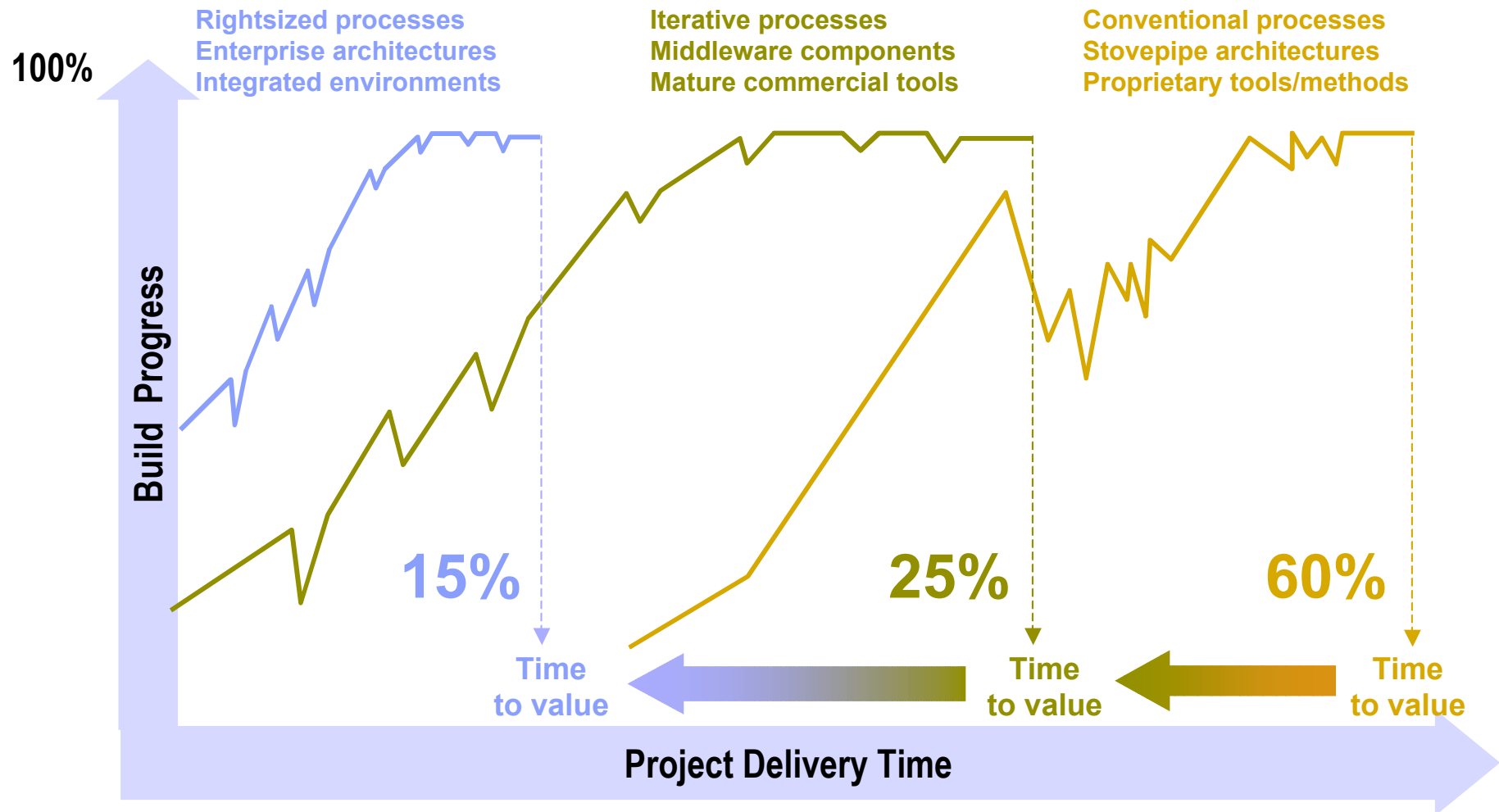


# Agenda

- Trends impacting software development
- Key principles for business-driven development
- An End-to-End Solution for SOA
- **Summary and conclusions**

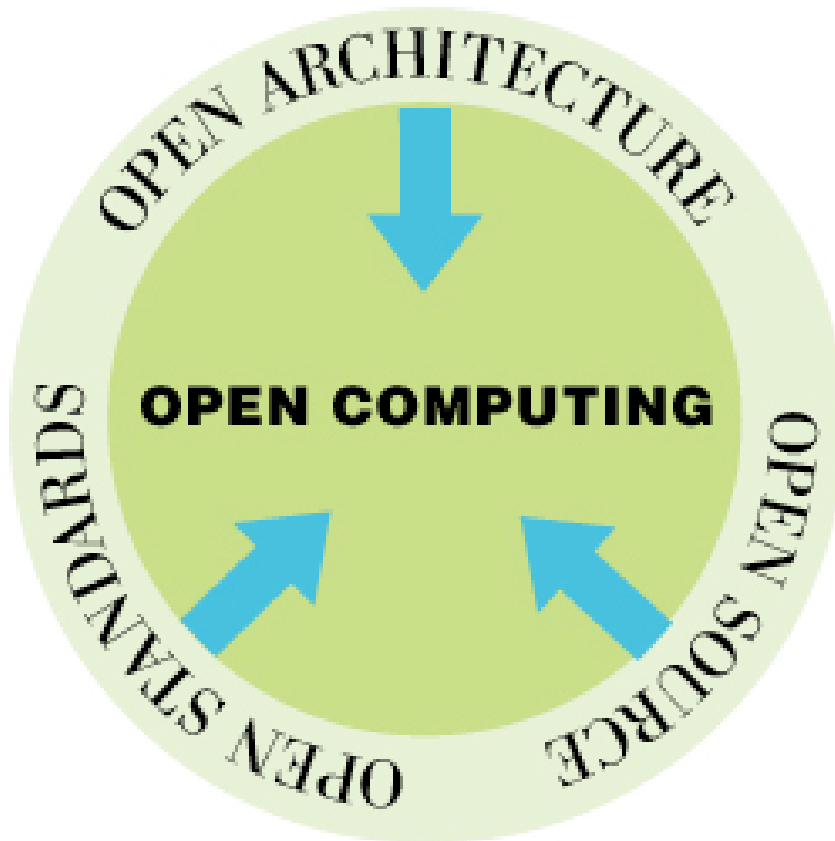


# Improving Time to Value





## Business-Driven Development



### Key Principles

**Adapt the process**

**Balance stakeholder priorities**

**Collaborate across teams**

**Demonstrate value iteratively**

**Elevate the level of abstraction**

**Focus continuously on quality**

## Questions???



Thank  
you