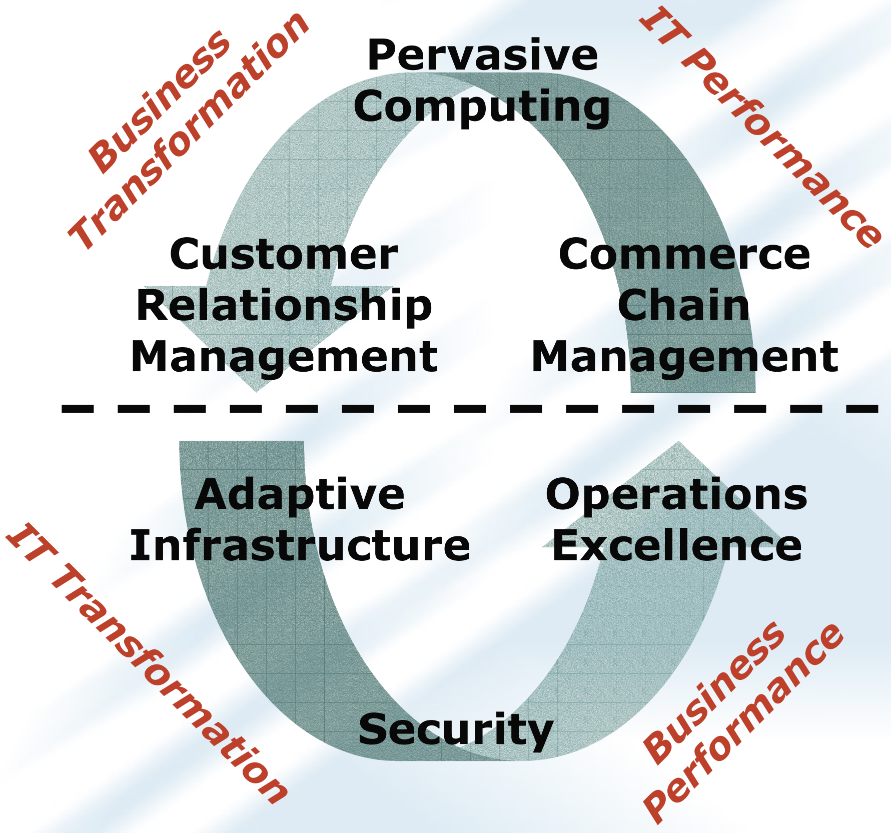
The Changing Nature of Application Testing

METAGROUP

Business and Technology Scenario

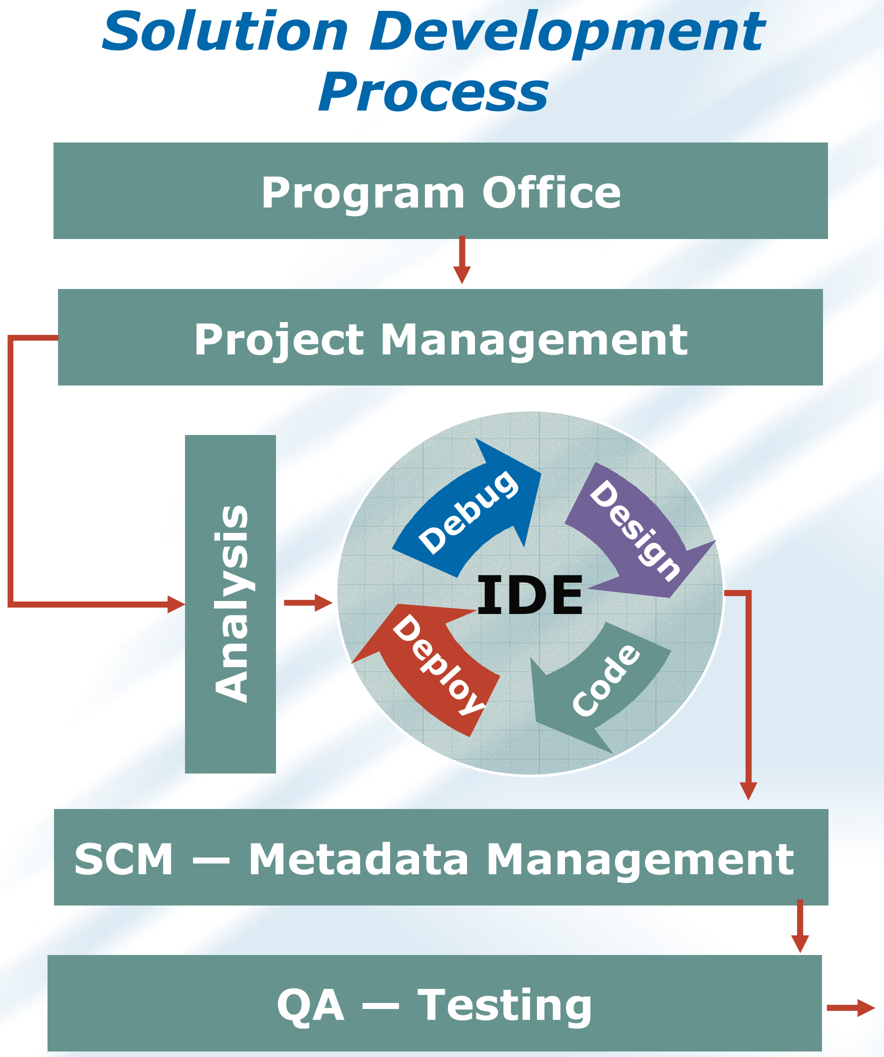
- Corporate value of pervasive computing drives need for control and predictability
- Must incorporate business drivers and customer relevance or lose audience
- Application life cycle in this arena is complex, due to multifacing interactions, multiple devices, and standards

The Evolving Business/IT Decision Cycle



Application Testing Critical Issues

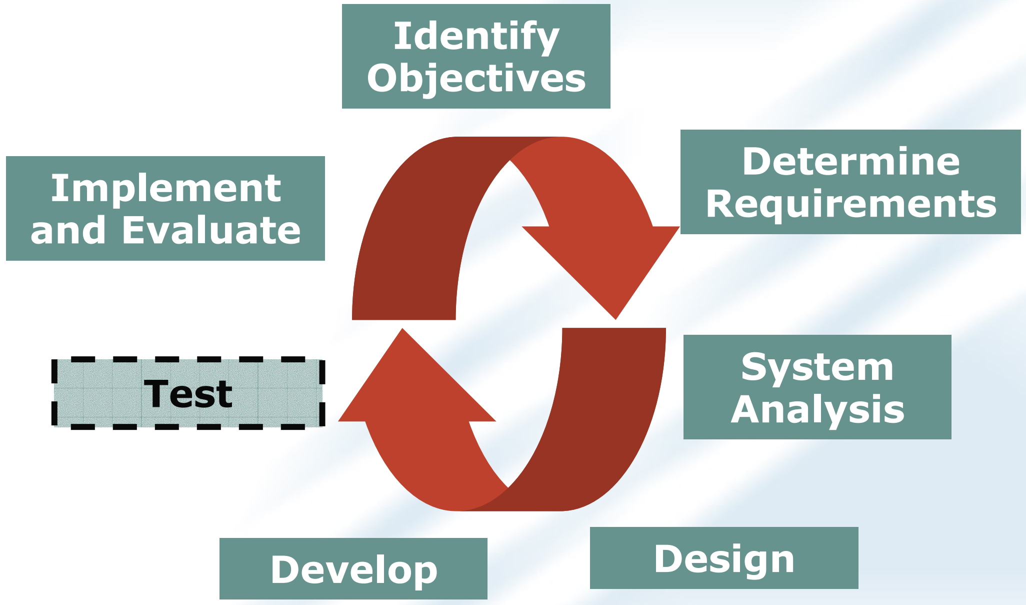
- The role of testing in the application life cycle
- Injecting discipline into the application development process
- Reflecting real-world conditions for successful deployment



Testing in the Application Life Cycle

- Complexity is exploding
- Adopt a life-cycle taxonomy
- Make testing mandatory
- The new application life cycle

Development Without Testing Leaves the Life Cycle Incomplete

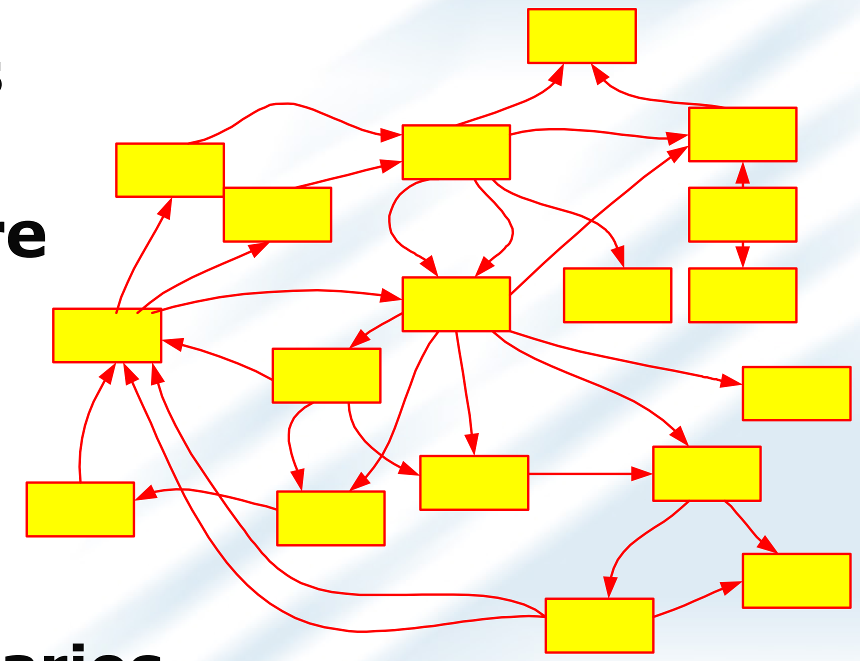


Prepare application development processes for drastically increasing complexity

The Complexity Explosion

- Monolithic applications have given way to distributed components
- Web services will yield highly dynamic structure
  - Components and relationships will change frequently at runtime
  - Testing needs to reflect worst-case scenarios

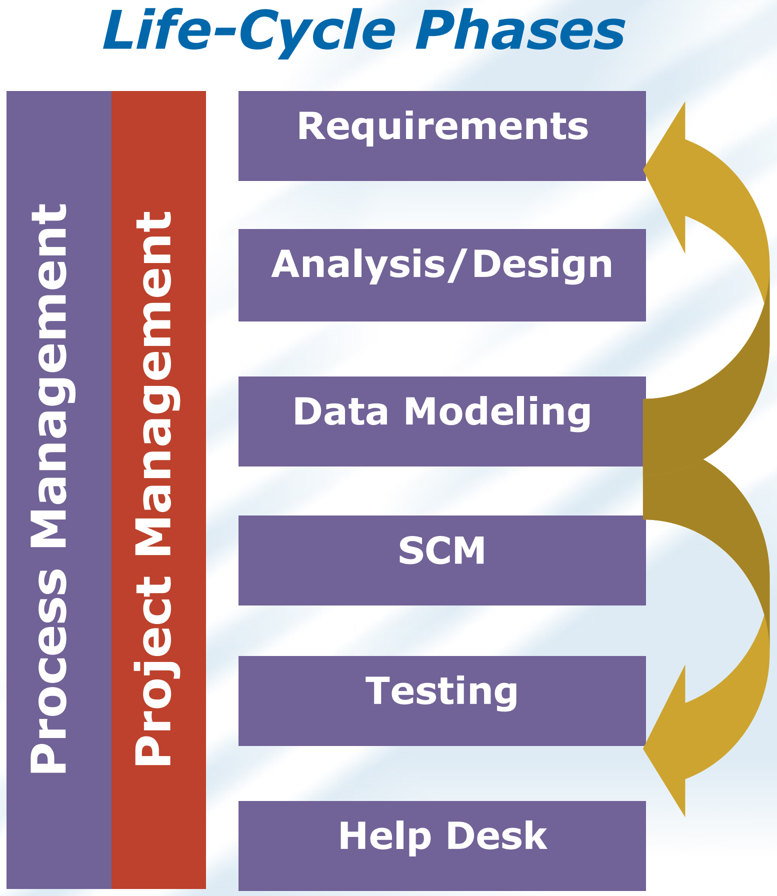
W3C Web Services Concepts and Relationships



To address complexity, adopt a flexible life-cycle structure and development culture

Adopt a Life-Cycle Taxonomy

- Apply iterative development techniques across life-cycle phases
- Evaluate BOB life-cycle integration capabilities
- Establish project and process management across phases
- Address weak links between PM, testing, and SCM



Prohibit application deployment to proceed without adequate testing

Enforce Mandatory Testing

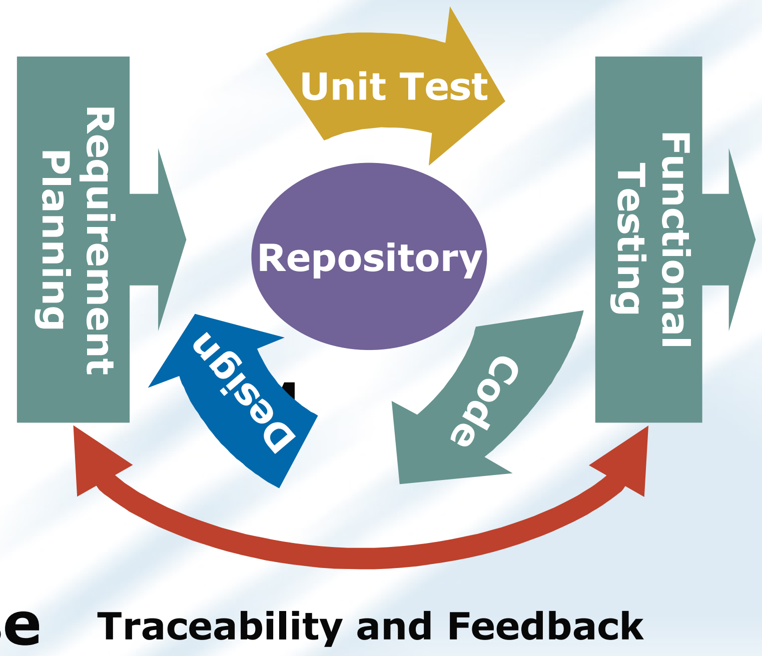
- **No application should go to production without adequate testing**
- **Apathy toward testing results in costly redesign after release and injures IT credibility**
- **ROI can easily exceed 500% within 6 months of application release**
- **Tie individual and organizational compensation to application reliability and accuracy**
  - **A powerful motivator**
  - **Measurement accuracy is critical and includes service-level metrics from operations**

Look to testing as the final gate of any application development life cycle

The New Application Life Cycle

- **Life cycle becomes iterative**
- **Controlled releases**
  - **Not the big bang**
  - **Each release is tested**
- **Create partnership to deliver value**
- **Same principles as old, different implementation**
- **Time is compressed**
  - **Development cycles impose high pressure for delivery**

Iterative Development



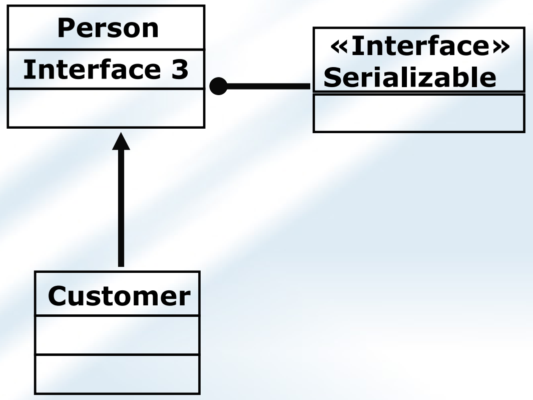
Instill engineering discipline in application development to drive the new life cycle

Create Discipline in Development

- **Challenge current practices**
  - **Demand radical efficiency improvements**
- **Do not just write the application, engineer it**
- **Automate testing whenever possible**
- **Continual re-assessment**

Documenting Design

Class Name	
Responsibilities	Collaborators

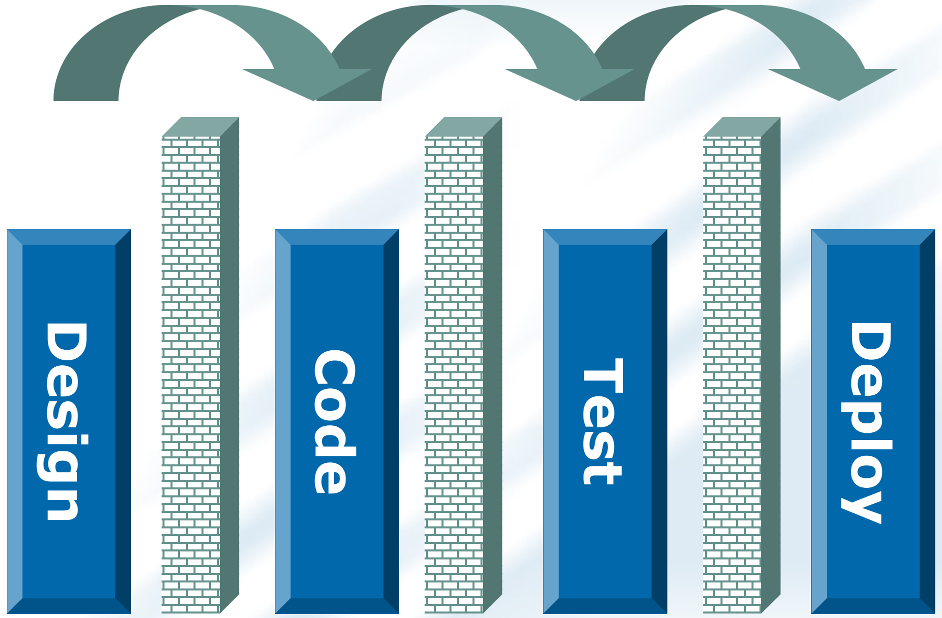


Evaluate current practices to eliminate any haphazard approaches and habits

Challenge Current Practices

- **Current practices cannot easily adapt to coming changes**
  - **Cultural inertia**
  - **Too static**
- **Maverick approaches drive innovation**
  - **Account for emerging technologies**
  - **Resist adherence to obsolete organization**

Traditional Process Isolation

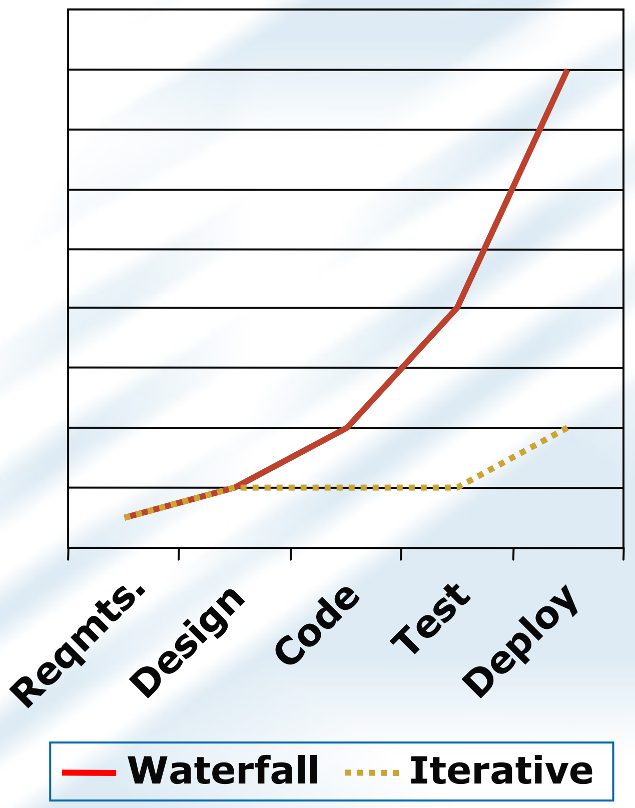


Transform development to a culture of engineering based on proven practices

Engineer Applications

- **Follow proven processes**
- **Obsess about possible failure modes and bottlenecks**
  - **Early detection saves 50%-90% of post-deployment costs**
- **Application errors occur when implementations do not match requirements**
  - **Coding errors**
  - **Requirements gaps**
  - **Change**

Software Change Costs



Follow up engineered processes with automation to accelerate process execution

Automate Testing Appropriately

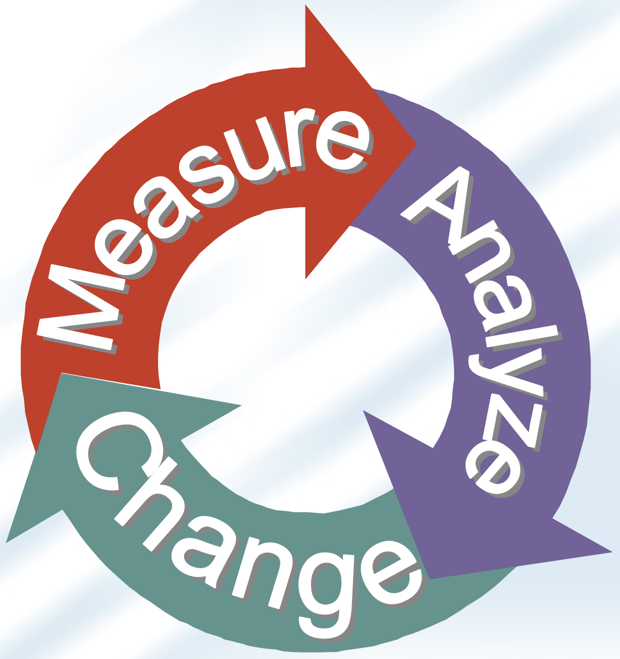
- **Testing products are now highly automated**
  - **Simplified script setup and execution**
  - **Repeatable and parameterized**
  - **Ideal for tedious, repetitive tasks**
- **Next steps will discover application behavior and auto-construct tests**
  - **Reduces human error**
  - **More comprehensive**
- **Tools alone are inadequate**
  - **Processes, training, and staffing are vital**

Continue automated testing through the entire application life cycle

Continual Re-Assessment

- Strive toward perfection
  - Prioritize by business needs
  - Reach a pragmatic plateau
- Develop and adhere to a periodic assessment test
  - Squeeze every ounce of performance and reliability from applications
  - Drive infrastructure change
  - Assimilate changing business requirements

Optimize Performance Through Repetition

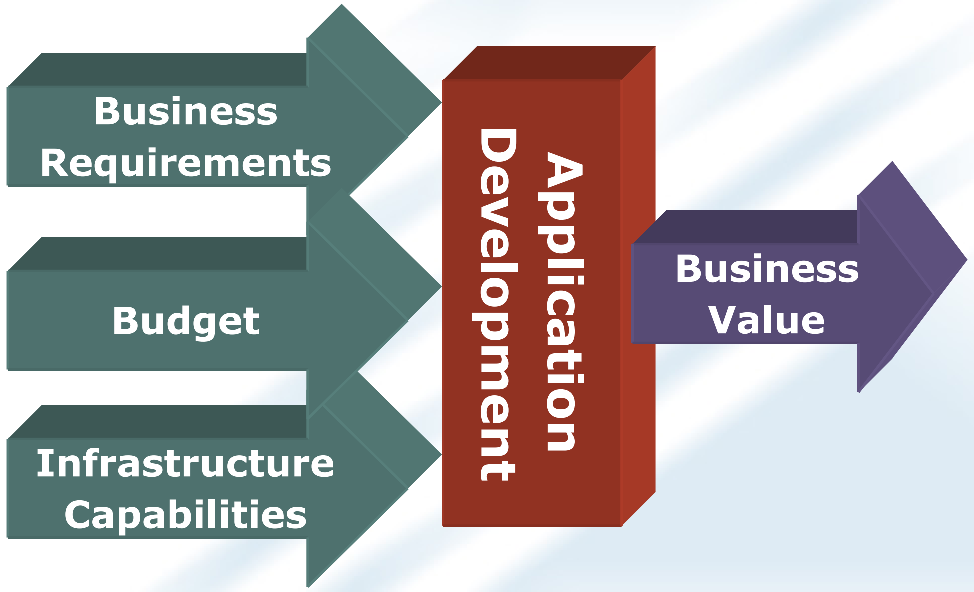


Repeat re-assessments to maintain relevance through constantly changing conditions

Incorporate Real-World Conditions

- Enforce realistic business requirements
- Use modeling and emulation
- Stress-test the actual environment
- Coordinate with other organizations and processes

The Development Process Requires a Rich Dose of Reality to Achieve Business Value



Manage expectations and promises for both IT and business stakeholders

Set Realistic Business Requirements

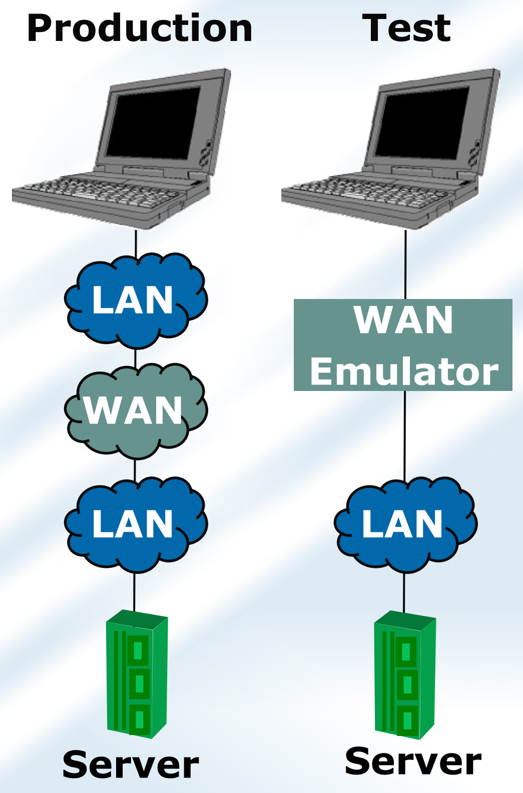
- **Business determines functional requirements**
- **Early consensus by all parties**
  - **Minimize disruptive changes**
  - **All must agree on change process/implications**
- **Determine feasibility**
  - **Beware unrealistic desires**
  - **Cost/benefit analysis**
  - **Communicate iteratively**
- **Involve business relationship management process and staff**

Combine requirements with application and infrastructure behaviors for precise testing

Exploit Modeling and Emulation

- Emulators mimic infrastructure behavior
  - e.g., WAN latency, packet loss, and throughput
  - Controlled, low-risk environment
- Substitute actual systems
  - Test in a simulated model
  - Follow-up with testing in actual environment
  - Compare to improve models

Mimic Reality for Most Effective Testing



Define model attributes based on actual software and infrastructure configurations

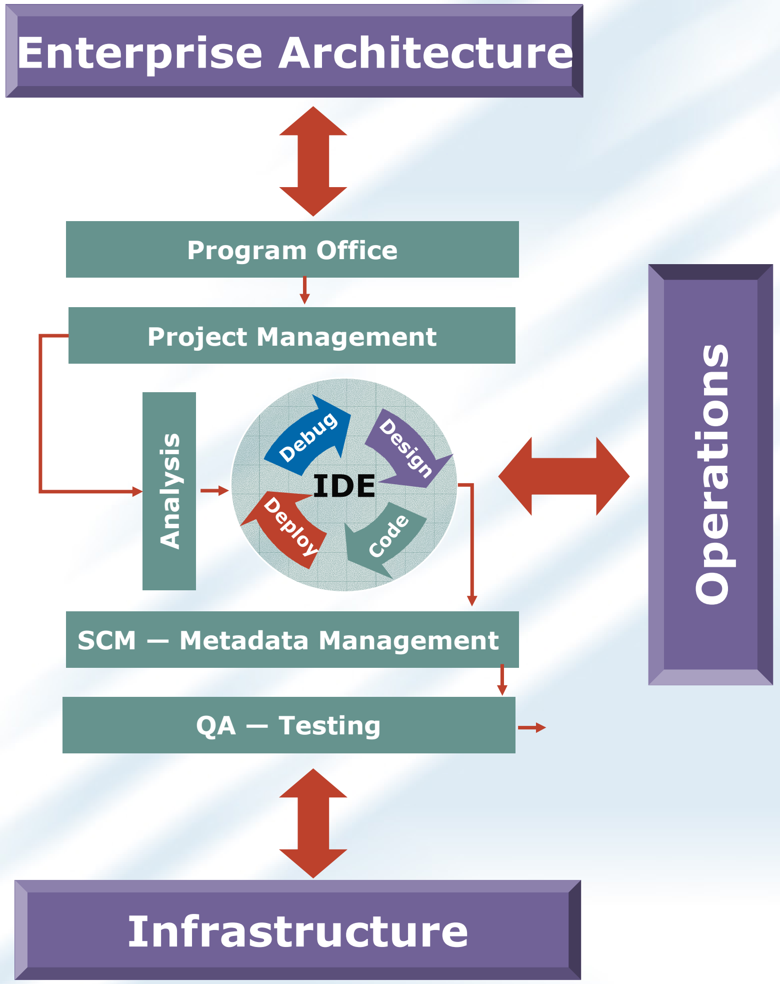
Software Change and Configuration Management

- Understand new SCM requirements for emerging technologies — automated tools lagging in support
- Coordinate with QA/QC to ensure consistent quality
- Identify internal and leverage external SCM best practices — key to handing complexity
- Establish IT/LOB criteria for prioritizing code changes for e-business
- Match and prioritize code changes with business requirements consistently

Coordinate change and configuration management with ongoing operational efforts

Integrate Solution Development

- Architecture process
  - Drives requirements
  - Constrains technology choices
- Infrastructure
  - Developed in parallel to applications
- Operations
  - Design handoff early
  - Requirements feedback loop
- Tests transition well to performance monitoring



Develop cooperative relationships with all stakeholders to protect credibility

Changing Nature of Application Testing

- Fit testing into the application life cycle
  - Make testing a mandatory stage of the SDLC
  - Develop processes and technologies to adapt to increasingly complex applications and systems
- Instill discipline in the development process
  - Discard obsolete methods in lieu of iterative, progressive ideas
  - Impose engineering structure to optimize results
- Account for reality in testing
  - Negotiate and enforce business requirements
  - Coordinate with change and configuration management
  - Cooperate with operations and architecture for realistic conditions