Software Quality

Jon C. Arce (jonarce@microsoft.com)
Architect Advisor
Microsoft – CCA Region

Microsoft

Current Status

- Software is the only product where large numbers of defects seem acceptable
  - U.S. Average Defect Rate – 5.9 to 7 defects per thousand lines of code (Software Assessments, Benchmarks, and Best Practices by Capers Jones)
- Software defects rates have increased 15% in 1999-2004 compared to 1995-1998 (Meta Group January 2004)
- Average computer user experiences a crash 2.5 times a week (InfoWorld 9/17/01)

Current Status

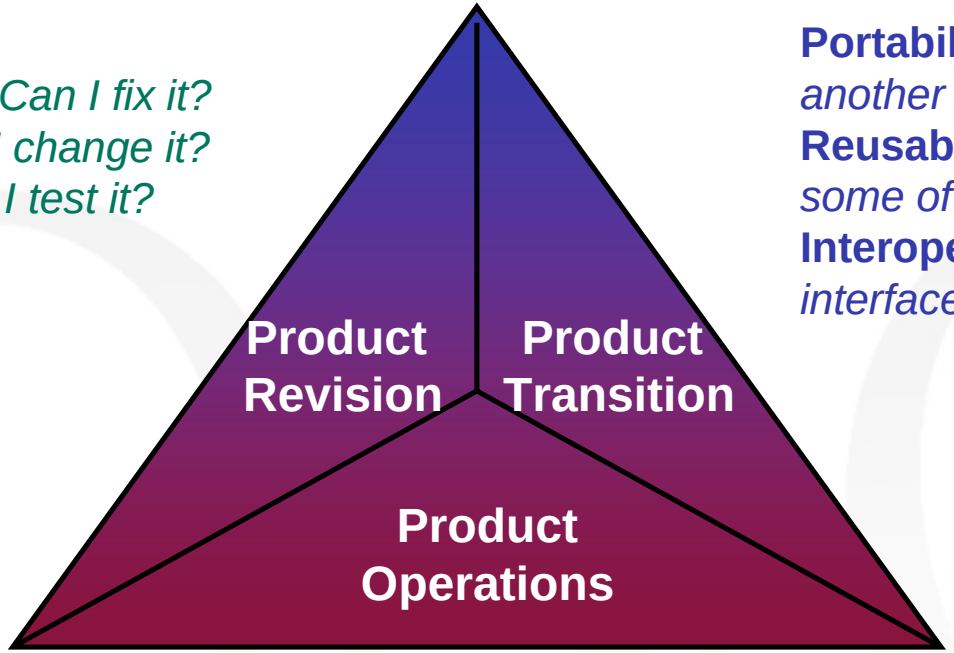
- 80% of technology projects cost more than they return *(Computerworld.com/ROI June 2001)*
- 73% of companies do not measure the success of technology spending *(CIO Insight March 2003)*
- 75% of companies do not track the cost of quality (Inside Quality survey 7/12/03)
- 48% of companies do not have formal test plans *(Information Week August 26, 2003)*

What is quality in Software?

- Our goal:
  - Build high-quality software systems
- Quality attributes:
  - Reliability
  - Usability
  - Understandability/Modifiability
  - Efficiency
  - Testability/Verifiability
  - Portability

Quality Attributes

Maintainability - *Can I fix it?*
Flexibility - *Can I change it?*
Testability - *Can I test it?*



Portability - *Will I be able to use on another machine?*
Reusability - *Will I be able to reuse some of the software?*
Interoperability - *Will I be able to interface it with another machine?*

Correctness - *Does it do what I want?*
Reliability - *Does it do it accurately all the time?*
Efficiency - *Will it run on my machine as well as it can?*
Integrity - *Is it secure?*
Usability - *Can I run it?*

Implications of Software Quality ?

• 55% of all errors are *made*, but *less than 10%* are detected during requirements capture and analysis

•Delayed error detection induces exponential cost increase for error correction

•Economic incentive for concentrating on early phases of software development

Why is High Quality Software Difficult to Build ?

- Software Quality is difficult to measure
- Software is easily changeable at any stage during the development process
- Implementation platforms and technologies change Rapidly

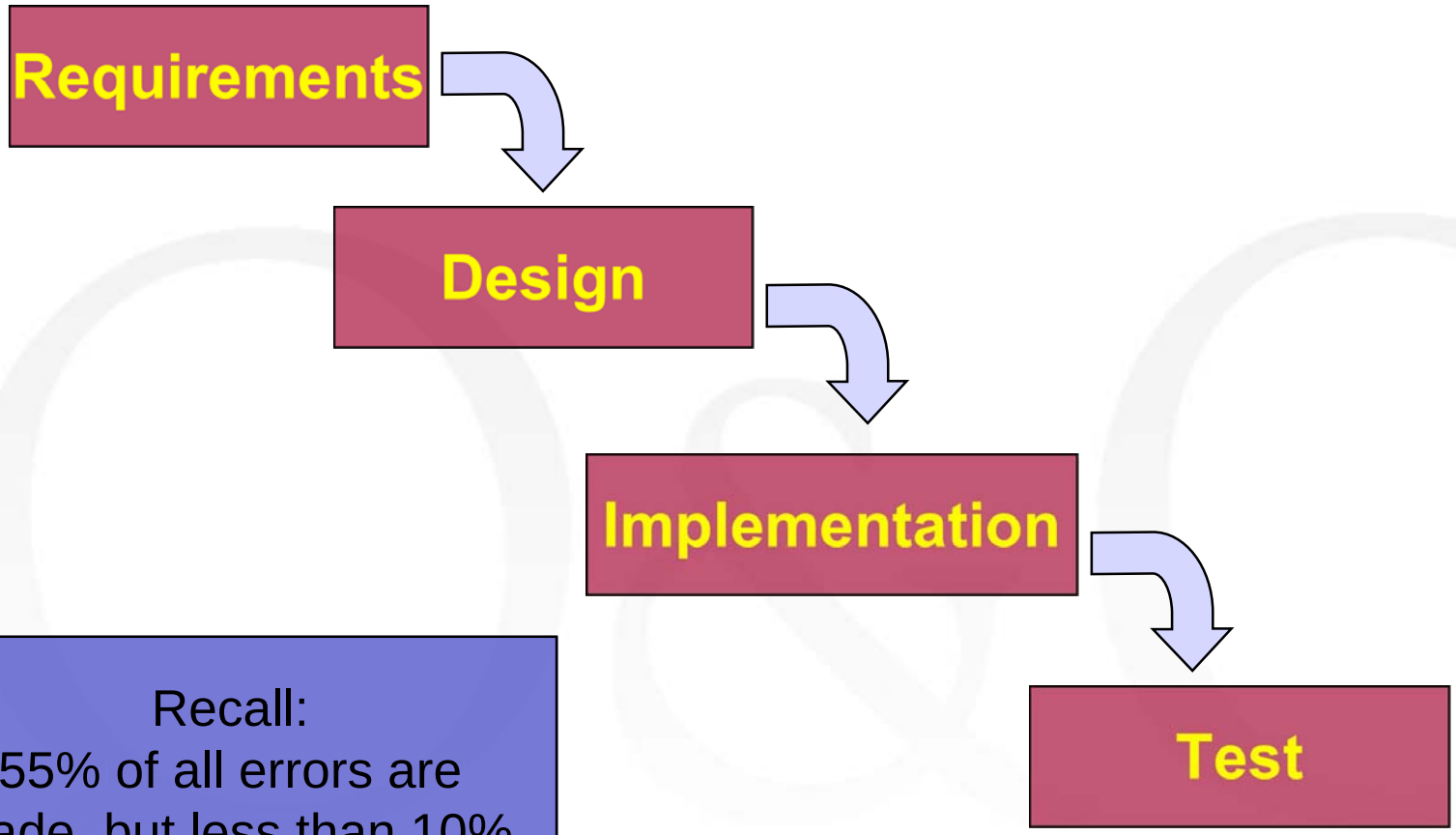
- Complexity is rapidly increasing
  - For every 25% increase in problem complexity there is a 100% increase in complexity of the software solution
  - Adequate methods and tools do not exist or are only partially adopted

- Requirements change

Quality Domains

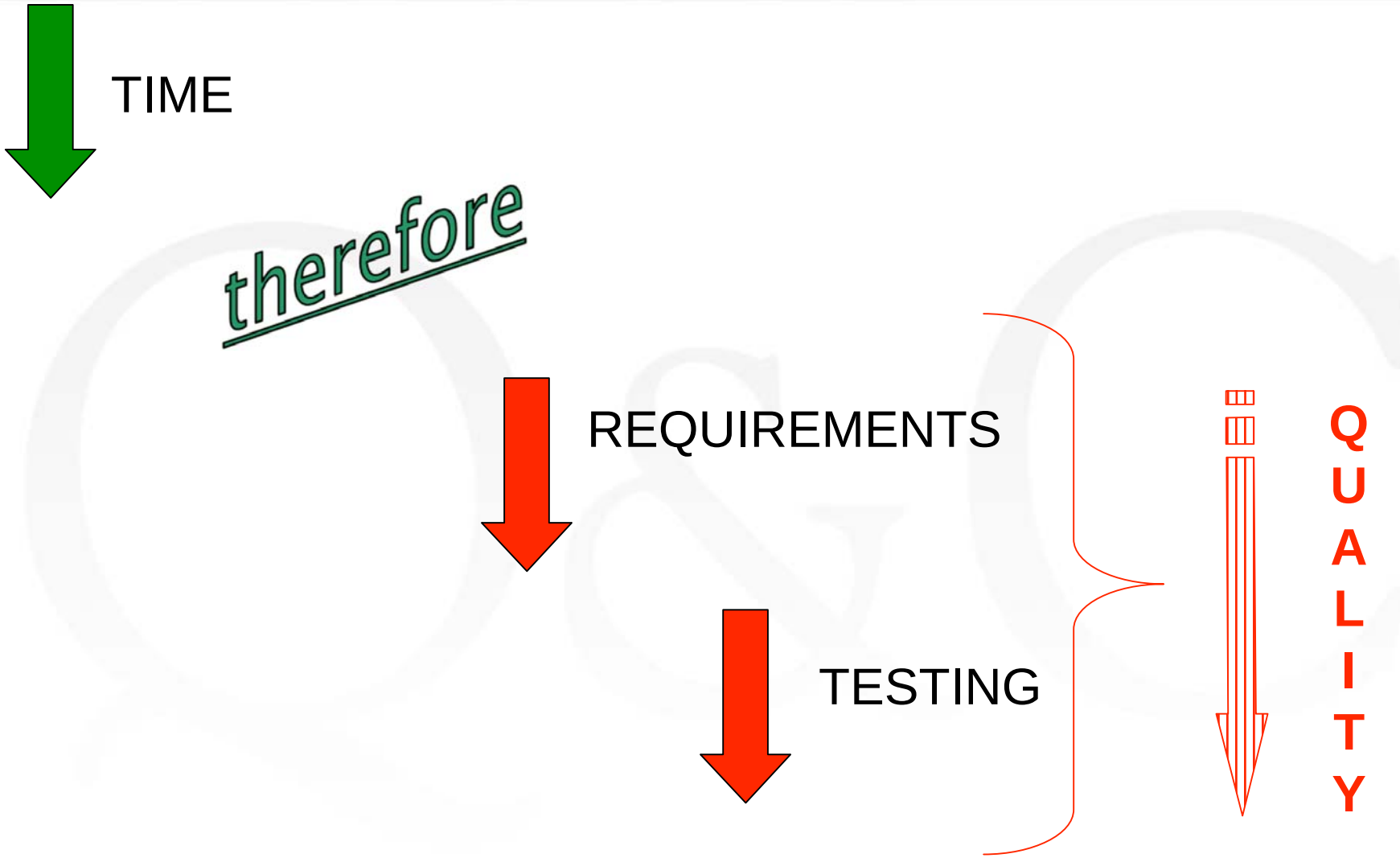
- Process (Software Development Methodologies)
  - Requirement Analysis
  - Risk Analysis (the unknown)
- Testing
  - Functional (use cases)
  - Development (regression / unit)
  - Stress (volume / response time)

Waterfall Model



Recall:
55% of all errors are made, but less than 10% are detected during RA

“Traditional” Development

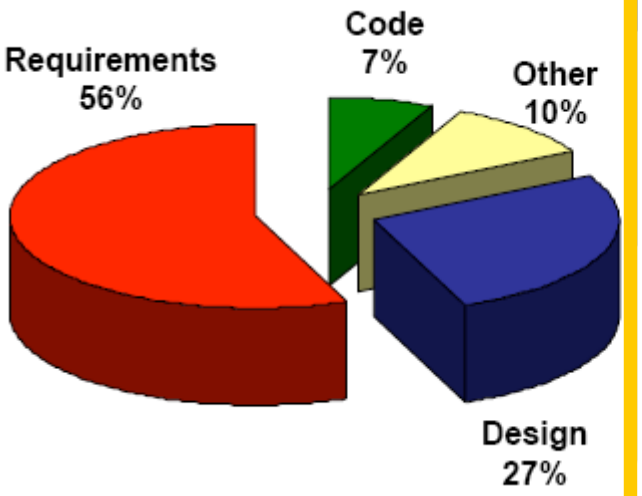


Results in

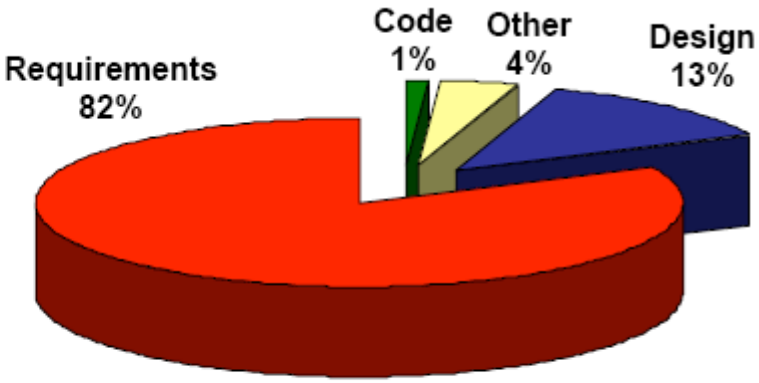
Gatotorra

Where are the bugs ?

- Distribution of Bugs



- Distribution of Efforts to fix Bugs



Quality Assurance

Architecture vs. Process

• Requirement deficiencies are the prime sources of project failures.

Denver International Airport (DIA) was scheduled to open on October 31, 1993 with all three of its concourses fully running on a newly developed, complex, automated baggage handling system.

It took until February 28, 1995 for DIA to finally open ***(sixteen months late!)****.*

Architecture vs Process

• Requirement deficiencies are the prime sources of project failures.
• Errors are most frequent during the requirements and design activities and are the more expensive the later they are removed.

• Protoyping (significantly) reduces requirement and design errors, especially for user interfaces.

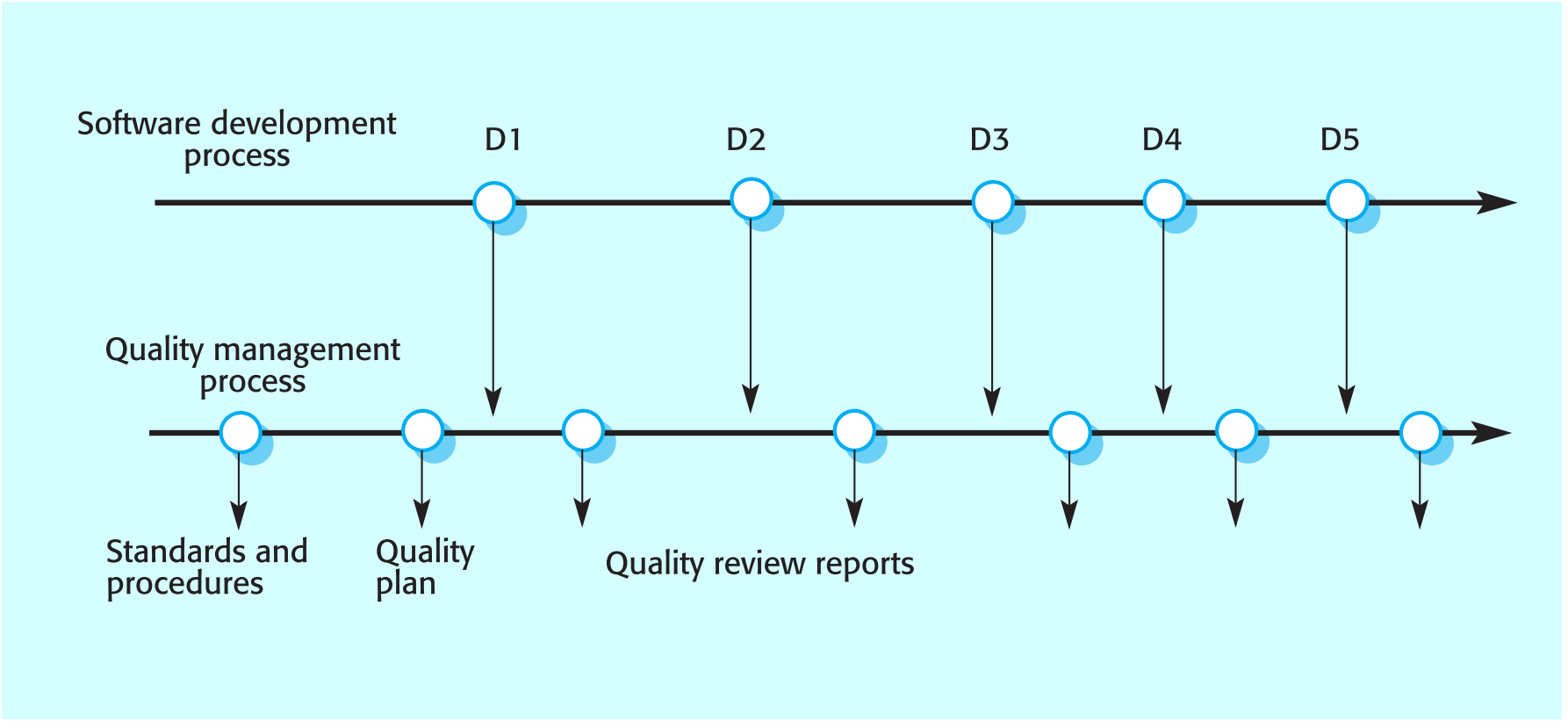
Cost to fix errors

Phase In Which Found	*Cost Ratio*
Requirements	1
Design	3-6
Coding	10
Development Testing	15-40
Acceptance Testing	30-70
Operation	40-1000

Quality management activities

- Quality assurance
  - Establish organisational procedures and standards for quality.
- Quality planning
  - Select applicable procedures and standards for a particular project and modify these as required.
- Quality control
  - Ensure that procedures and standards are followed by the software development team.
- Quality management should be separate from project management to ensure independence.

Quality management and software development



Can we or should we survive without a documented process ?

- If you cannot show that your process is better than random
  - It doesn't matter if is documented
- Anything not understood may be out of control
  - Measurement counts not words
  - You need to understand variation in a process
- **People think a documented process is the process**
  - It is not, it is a model of a process
  - There are plenty software models to choose from
    - Waterfall, V Model, Prototype Model, Extreme Programming
- **A process is only part of the equation**
  - Process + Culture + Knowledge = Action

Requirement Testability

- Often customers come up with requirements that are not testable, to validate ask:
  - Can we define the acceptance criteria for this requirement?
  - Clearly state the assumption you have made on this requirement. Check if the assumption is conflicting with any other assumption / requirement made so far.
  - Is this requirement clashing with any other requirement?
  - Can it be broken into multiple requirements?

NASA - Probability Risk Factors

Factors contributing to probability of software failure	Un-weighted probability of failure score					Weighting Factor	Likely-hood of failure rating
	1	2	4	8	16		
Software team complexity	Up to 5 people at one location	Up to 10 people at one location	Up to 20 people at one location or 10 people with external support	Up to 50 people at one location or 20 people with external support	More than 50 people at one location or 20 people with external support	X2	
Contractor Support	None	Contractor with minor tasks		Contractor with major tasks	Contractor with major tasks critical to project success	X2	
Organization Complexity*	One location	Two locations but same reporting chain	Multiple locations but same reporting chain	Multiple providers with prime sub relationship	Multiple providers with associate relationship	X1	
Schedule Pressure**	No deadline		Deadline is negotiable		Non-negotiable deadline	X2	
Process Maturity of Software Provider	Independent assessment of Capability Maturity Model (CMM) Level 4, 5	Independent assessment of CMM Level 3	Independent assessment of CMM Level 2	CMM Level 1 with record of repeated mission success	CMM Level 1 or equivalent	X2	
Degree of Innovation	Proven and accepted		Proven but new to the development organization		Cutting edge	X1	
Level of Integration	Simple - Stand alone				Extensive Integration Required	X2	
Requirement Maturity	Well defined objectives - No unknowns	Well defined objectives - Few unknowns		Preliminary objectives	Changing, ambiguous, or untestable objectives	X2	
Software Lines of Code***	Less than 50K		Over 500K		Over 1000K	X2	
Total							

Table 1 Likelihood of Failures Based on Software Environment

Refactoring Techniques

“Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure.

It is a disciplined way to clean up code that minimizes the chances of introducing bugs.

In essence when you refactor you are improving the design of the code after it has been written.”

But you introduce new bugs as well …

Safe Refactoring Techniques

"Refactoring:

- "minimally invasive" modifications to system structure
  - ⇒ *Strategy of small steps*

- Set up adequate test suite before changing the system

- Carry out tests during and after performing the change
  - ⇒ Increases confidence in correctness;
    Goal: no change of observable behavior

Importance of Testing

- Testing is most effort-intensive activity in software projects
  - But is still often stopped too early or curtailed in a late project
- Scope of testing
  - Unit testing = testing of individual modules
  - Integration testing = testing of assemblies of modules
    - System testing is the limiting case

Questions on Testing

- What is testing?
  - Seeing whether a program compiles?
  - Reading a program carefully?
  - Proving that a program works?
- What is the difference between testing and debugging?
- What gets tested?
  - Entire system?
  - Modules, one-by-one?

What you need for testing

- Something to test
  - code & stubs for code that calls
- Test cases
  - *concrete* inputs and expected outputs
- Some way of providing test case inputs
  - drivers (including CLI + main program)
- An “oracle” for judging whether test cases pass
  - usually the specification and your judgment

Terminology

- Failures & faults
  - *Failure:* Undesirable difference between observed & expected behavior
  - *Fault/defect*: Cause of failure
  - *Defect rate*: Number of defects per 1000 lines of code
- Testing vs. debugging
  - *Testing*: Systematically finding failures by running code
  - *Debugging*: Finding & fixing defects in the code
- *Black-box* vs. "*white-box*" testing
  - Do you have access to the code in producing test cases?
- *Scaffolding*: software to support testing

Testing

- Purpose
  - "Program testing can be used to show the presence of errors, but never to show their absence" (Dijkstra)
  - Gain confidence in the system by failing to find defects
- Who tests?
  - Programmers should only release code with which they are confident
  - *But programmers are predisposed to believe their programs are correct*
  - Test each other's code or use independent organization

Mutation testing

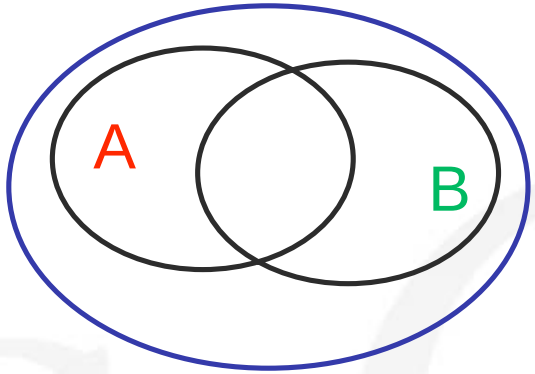
- How does a manager measure the adequacy of the test cases?
  - E.g. to decide when to release?
- (1) Track error detection and release when below a threshold
- (2) Mutation testing
  - Deliberately introduce bugs
  - Measure how many of those bugs are found
  - Use statistical reasoning to predict how many other bugs remain in the code
  - Assumptions:
    - Injected bugs and residual bugs are similar
    - Bugs are equivalent in effect

Stopping Testing

Assuming that defects detected are representative:

S/N = s/n

(total) (detected)



defects in 1000 LOC

size(A) = a
size (B) = b
size (A ^ B) = q
size (A v B) = n

effectiveness(A) = a/N = q/b
effectiveness (B) = b/N = q/a

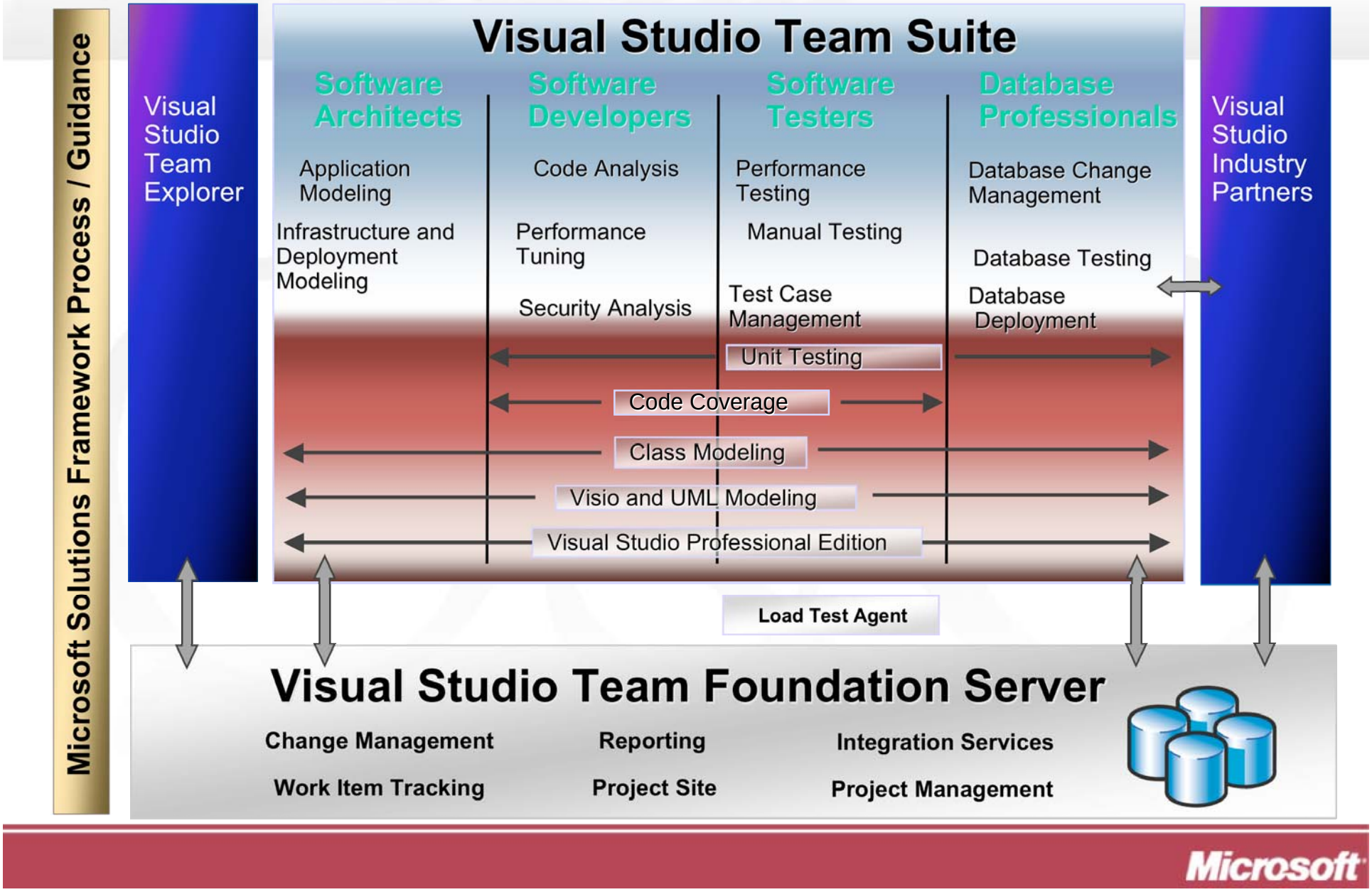
=> N = q/effectiveness(A)*effectiveness(B)

Example:
a =25
b = 30
q = 15
N = ?
(N-n)/N = ?
Would you release?

Visual Studio Team System

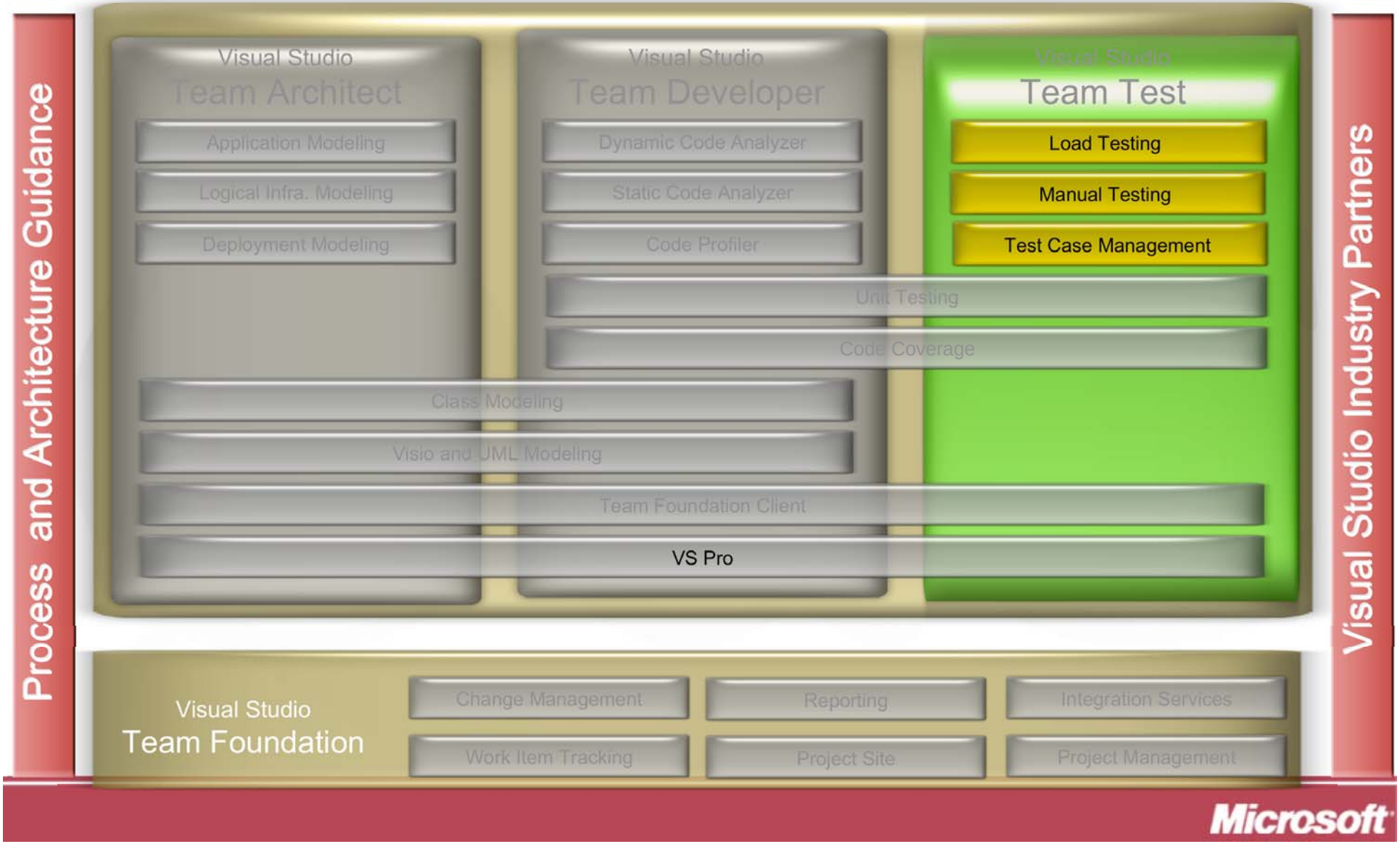
Visual Studio Team System

Visual Studio Team System



Visual Studio Team System

Team Test

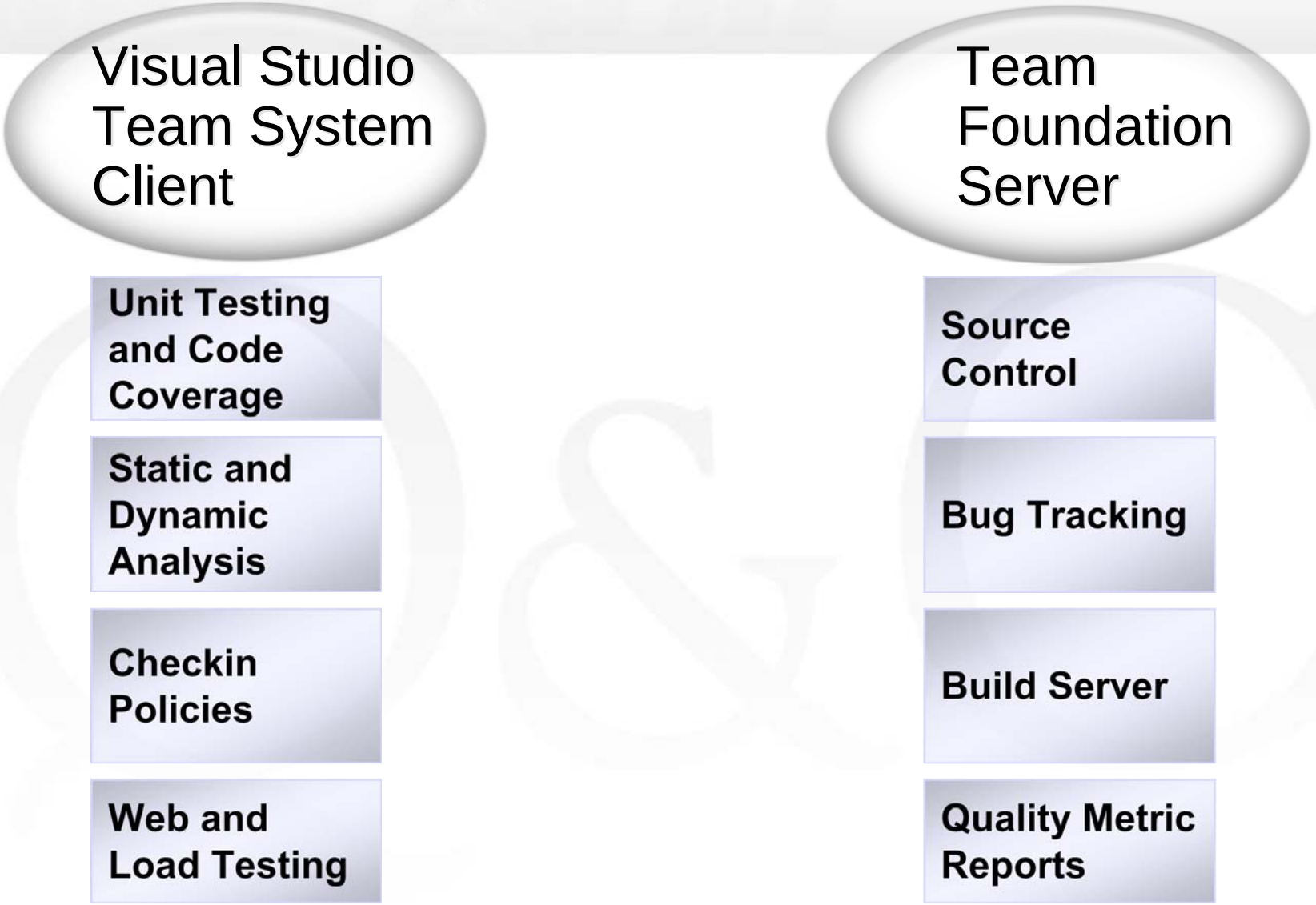


Visual Studio Team System

Team Test

- Load Testing
  - Web Test Recording
    - Customizable .NET code
  - Load Testing
    - Wait time, data loading
  - Counter collection
    - With established limits
- Functional test providing
  - Usage case handling

VSTS: Quality for all phases



Dev/Test Teamwork

- A test is a test is a test
  - Testers use Dev unit tests as Build Verification Tests or in functional test pass
  - Devs use Tester tests as Check-in tests
  - Devs use Tester tests to view test results and report bugs
- Integrated bug tracking
- Build server for smooth handoffs (continuous build supported)
- Seamless quality reporting

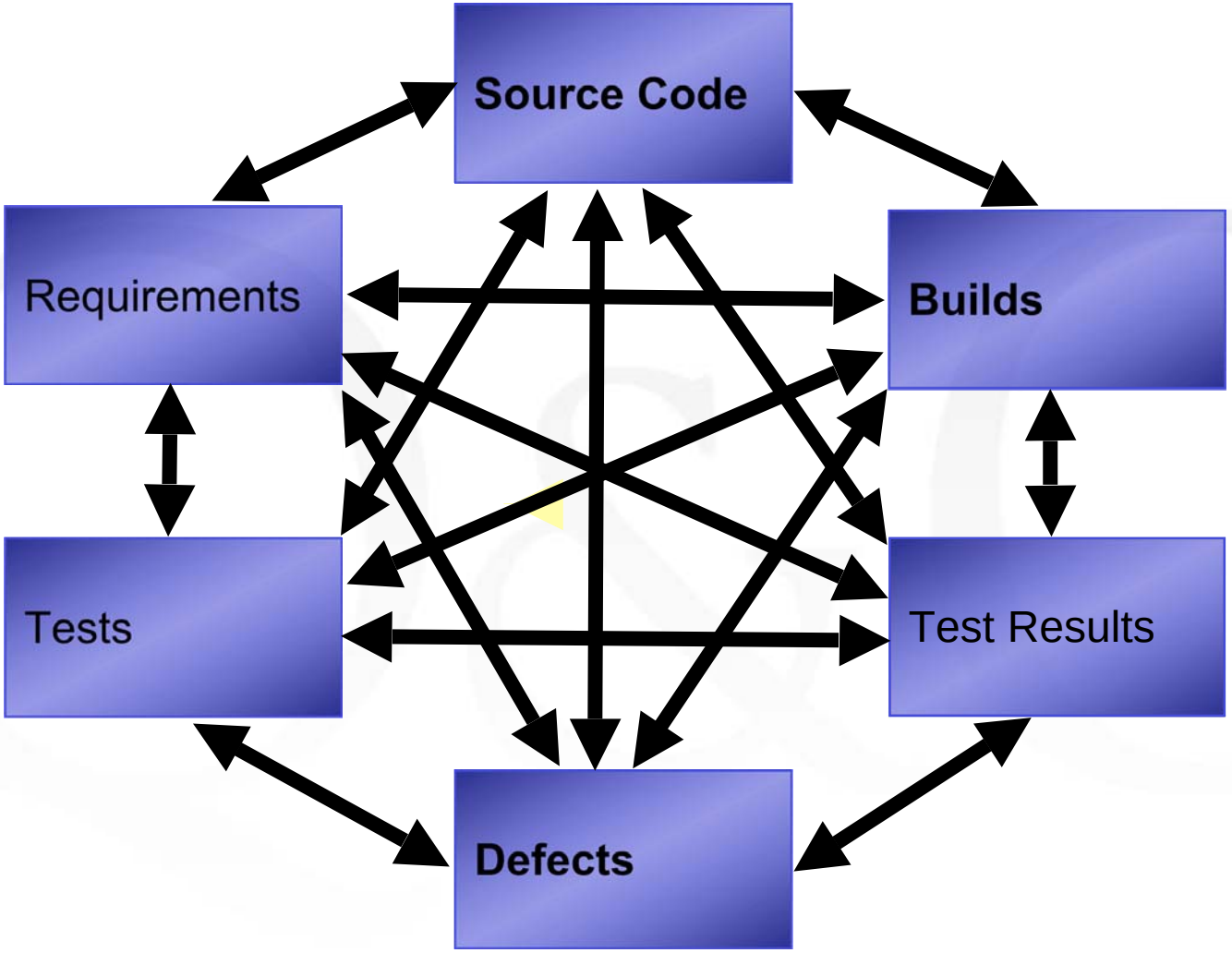
Demo

Load Test (05:27)

Managing Test Cases (03:32)

www.LearnVisual Studio.NET

Link Dev and Test Assets



Visual Studio Team System for Testers: Test Types

- Unit Tests
  - developer unit tests, but testers can also use them for testing web services
  - Any API, so testers can use to test web service
  - Code generation takes makes it fast to build new tests
  - Data driven tests: devs write tests, testers fill in input and expected output parameters

Visual Studio Team System for Testers Test Types

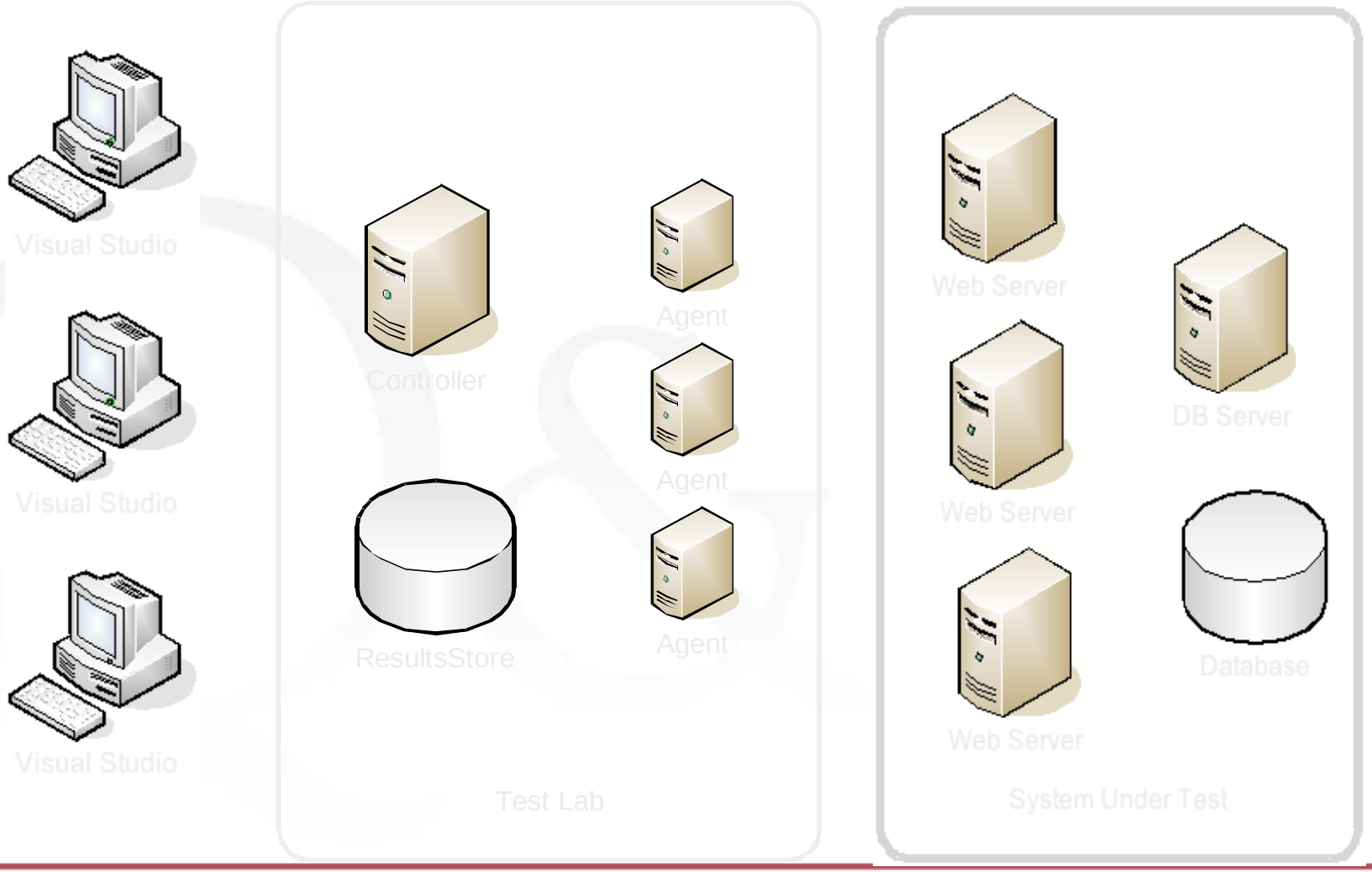
- Web Tests
  - Test your web application - Supports https, as well as NTLM and Basic auth
  - Performance under load
  - Extensible / supports coded web tests
- Load Tests
  - Performance counter sets for guidance
  - – Generic tests enable you to wire results from existing test harnesses to VSTS
  - – Manual tests enable you to report manual test results to TFS

Visual Studio Team System for Testers Test Types

- Generic Tests
  - enable you to wire results from existing test harnesses to VSTS
- Manual Tests
  - enable you to report manual test results to TFS
- Partner Tests
  - Extensible framework for partners

VSIP Partners

Visual Studio Team System
Team Test Load Agents



Resources

Team System for Testers Forum

http://forums.microsoft.com/MSDN/ShowPost.aspx?PostID=330419&SiteID=1&mode=1

Technical Chats and Webcasts

http://www.microsoft.com/communities/chats/default.mspx

http://www.microsoft.com/usa/webcasts/default.asp

Microsoft Learning and Certification

http://www.microsoft.com/learning/default.mspx

MSDN & TechNet

http://microsoft.com/msdn

http://microsoft.com/technet

Virtual Labs

http://www.microsoft.com/technet/traincert/virtuallab/rms.mspx

Technical Community Sites

http://www.microsoft.com/communities/default.mspx

User Groups

http://www.microsoft.com/communities/usergroups/default.mspx

Microsoft®

Your potential. Our passion.™