Shifting Left – A More Intelligent Direction

Jim Thomas
Director of Software Testing
Combining development and quality assurance earlier and more deeply
The Results

✓ Caught bugs earlier
✓ Saved time by working in parallel
✓ Saved resources
✓ More testable design
✓ Speeded up bug fixing
✓ Did more testing overall
✓ Motivated QA – part of the team, not seen as an overhead
We Do This ....

Requirements
High Level Design
Low Level Design
Code
Dev Test
Accept Test

Defects

Defect Detection Point
... But We Could Shift ‘Testing’ Left

Defect Detection Point
Quality comes not from testing, but from improvement of the development process. Testing does not improve quality, nor guarantee quality. It’s too late. *Deming*

It is always cheaper to do the job right the first time than to correct problems later (quality should be ensured through prevention, not appraisal) *Crosby (Quality is Free)*
Improving Requirements

**Individual**
- Correct
- Feasible
- Necessary
- Unambiguous
- Verifiable

**Set**
- Prioritized
- Consistent
- Modifiable
- Traceable

- Reviews/Inspections
- Multiple Viewpoints
  - Including Test/QA
"Charge numbers should be validated on-line against the master corporate charge number list, if possible."

"The system shall validate the charge number entered against the on-line master corporate charge number list. If the charge number is not found on the list, an error message shall be displayed and the order shall not be accepted."
Improving Software Design

• Define clear interfaces
• Understand the run-time environment
• Identify technical risk areas
  – Performance
  – Security
• Carry out benchmarking
• Prototype if necessary

  • Reviews/Inspections
  • Involve Technical Experts
Assists in code reviews
More advanced static analysis tools also weed out critical bugs, early!

Unit/integration testing then focuses on finding design issues
More Intelligent Dynamic Testing

Integration of Pre-tested Components
More Effort, Time and Cost Upfront

- Inspections/reviews
- Benchmarking/prototyping
- Static analysis
- Levels of testing
- Early Test/QA involvement

but

✓ Improved requirements, design and code
  ✓ Earlier test specifications and testing
  ✓ More predictable testing
  ✓ Improved motivation
Changing the Balance

Quality

Built in  Tested in
We All Know This ....

Relative Cost of Defect Fix

1X  5X  10X  20X  50X  >150X

Requirements  Design  Code  Dev Test  Accept Test  Operation

Source: Boehm – Equity Keynote Address 2007
Some More Data

40-50% effort on avoidable rework

Peer reviews catch 30-90% defects

Personal practice reduces defects up to 75%

Source: Boehm – Software Management Article 2001
And More ...

Best v Average SW Quality 6 times fewer defects

Programmers <50% efficient finding own defects

Inspections, Static analysis, Testing:
>97% efficient
Lower costs & schedules by >20%
Lower TCO by >45%

Source: Capers Jones – Software Quality in 2012: A Survey of the State of the Art
The Challenges

- Shift Left can look expensive
- Pressure to deliver software
- Requirements and design often rushed
- Inspections/reviews not popular
- Dev test too ad-hoc
- Test/QA has become developers’ crutch
Software Project Failures – Top Four Factors

1. Delivery date impacted the development process (93%)

2. Project was under-estimated (81%)

3. Risks were not re-assessed, controlled or managed through the project (76%)

4. Staff were not rewarded for working long hours (73%)

Verner, Sampson & Cerpa
All the Right Stages but Not Necessarily in the Right Order

- Software Requirements
- High Level Design
- Unit Level Design
- Coding
- Integration Testing
- Unit Testing
- Software System Testing
What's Needed

Mindset Change
Shift Left in Practice

Project Comparison

- Project A
- Project B

Resource

Months
Agile and Shift Left

• Not the same, but ....

• Compatible

• Validate earlier

• Test earlier

• Shorten test cycle
Agile Development

Sprint 1
Sprint 2
Sprint 3
Sprint 4
Sprint 5

Sprint 1 Dev
Sprint 2 Dev
Rwk
Sprint 3 Dev
Rwk
Sprint 3 Dev
Rwk
Better, Faster, Cheaper

- Shift Left isn’t new
- Focus on building quality in
- Bring verification/testing into the development phase
- Inspection/review, static analysis and more intelligent testing can be highly effective
- With a changed mindset

Better quality, Faster and Cheaper development (and happier users)