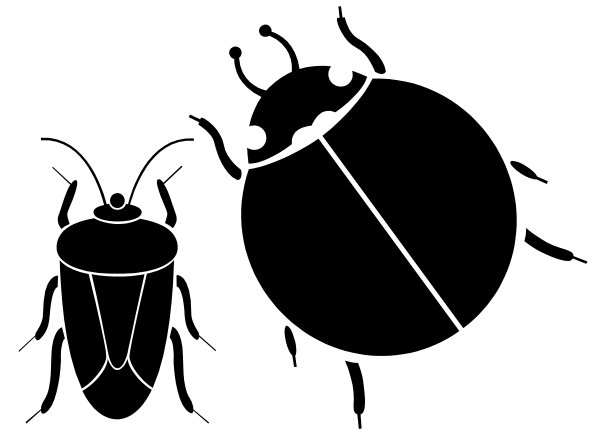




Exterminate your software bugs to lower your costs

Top five things you can do right now!

Jeff Van Fleet, President and CEO



Jeff Van Fleet – Who is this guy?

>25 Years Developing and Managing S/W

Former SW Manager at SAIC & TRW

- Helped lead SAIC to SEI Level 3
- Managed multi-million \$ projects and teams
- Developed, released, and supported global applications

Trainer

Founder - Lighthouse Technologies, Inc.

- Uses CMMI Level 3 practices to assure software is a success

Likes to have fun

- Dark beer, family, and the Pittsburgh Steelers
- Bug exterminator .. OK, software bugs only

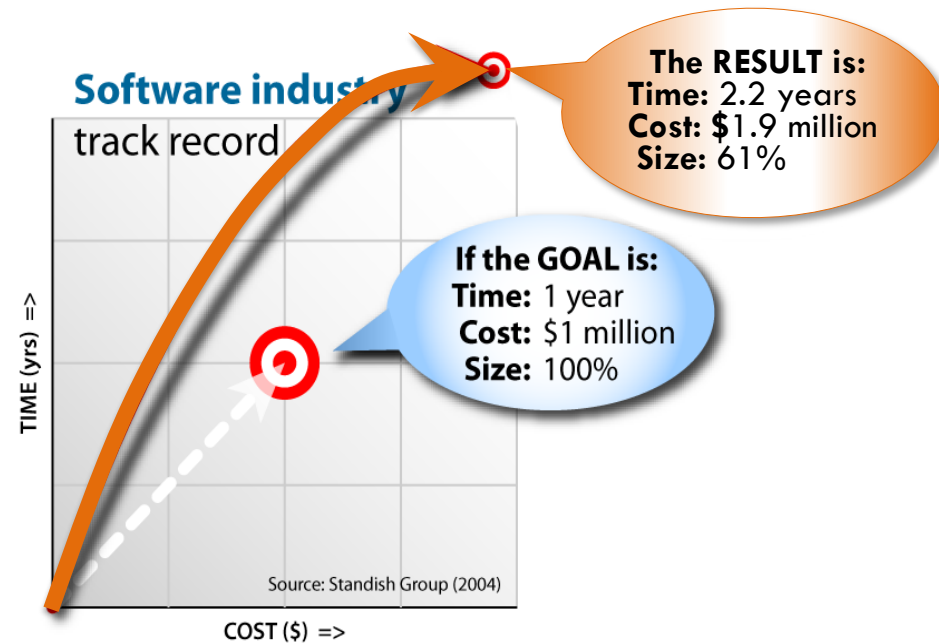


What we want

- On time delivery
- Expected functionality
- High quality
- Competitive price

- Open communications
- Clear expectations

What we don't want anymore



What is testing?

Test Phase	Who performs?	What is it?
Unit Testing	Developers	White box. Assuring each piece of code does what I intended it to do.
Component/ Development Integration Testing	Developers	White box. Assuring all the interfaces work as expected and the system integrates.
System testing	Test Team (ideally independent)	Black box. Verifies the system meets the requirements. Includes performance and load testing.
User acceptance testing	Generally end-users. Sometimes test team	Black box. Validates the system meets its intended purpose.

Percent of defects detected

CMMI Level	1	2	3	4	5
Unit Test	40%	60%	70%	75%	80%
Integration Test	10%	10%	10%	10%	7%
System Test	10%	10%	10%	7%	7%
Acceptance Test	20%	10%	5%	5%	4%
Defect Removal % prior to production	80%	90%	95%	97%	98%

data adapted from Capers Jones

The GOAL 

#5 – Focus where the risk is

Use a risk-based approach

Too expensive to test 100% of the system

- Need to target high-risk areas

Only 20-30% of the code assures the system does what it's supposed to do

- Be sure to have an equivalent # of negative test cases

Bugs hide where it is dark, warm, and damp.

Know where to look.

- Consider business risks
 - Critical business functions, High volume transactions
- Consider technical risks
 - Architecture, Critical interfaces, Code complexity



#5 – Focus where the risk is

Keep it simple

Functional Area	Business Requirement	Business Risk	Code complexity	Total Risk
Accounting	Month-end close	5 (High)	5 (High)	10
Accounting	Process Invoices	5 (High)	3 (Med)	8
Accounting	Create bills	1 (Low)	1 (Low)	2
Supply Chain	Order Parts	3 (Med)	1 (Low)	4
...				

“Total risk” level drives test case coverage and depth

#4 – Reduce Churn

Requirements and data

Agreement on what is being tested

- Build test cases from Use Cases
- Helps identify missing and ambiguous requirements
- No bias
- Use RTM traced to requirements

Test data is critical

- Well architected
- May need SMEs to help
- Primarily use controlled data
- For end-to-end, use live data



#4 – Reduce Churn

Know handoffs and expectations



Clear handoff from development

- Release notes - know what they think is in each build

Use separate testing environments

- Development, System Test, User Acceptance Test

Strong configuration control and release management

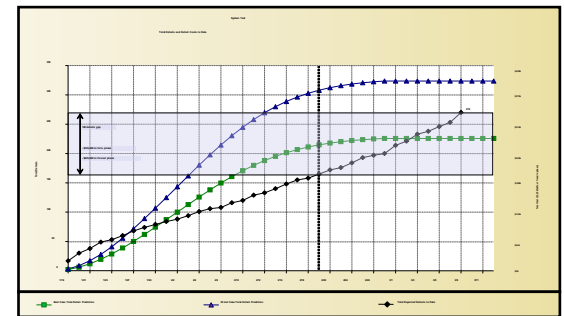
- Promote code through the environments

Well defined defect and change management

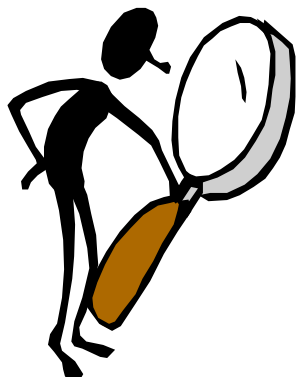
#3 – Establish Metrics and Set Goals

You can't manage what you can't measure

- Keep it simple to start
- Establish metrics role
- Develop metrics plan and initial spreadsheet
 - Test team productivity
 - Product quality metrics
 - Use to develop test exit criteria
- Record on all projects and use for test planning



Track actual performance



Start simple ... pragmatic

- Test team productivity
 - Test cases developed - planned vs. actual
 - Test case development effort (hrs) - planned vs. actual
 - Test cases executed - planned vs. actual
 - Test case execution effort (hrs) - planned vs. actual
- Product quality
 - Test failure rate (i.e., defect rate) – planned vs. actual
 - Total daily defects (critical, high, med, and low)
 - Daily defects discovered vs. defects fixed
 - Advanced - Defects (by functional area, by phase)
 - Advanced – Defect remediation (hrs)

#2 – Find bugs earlier

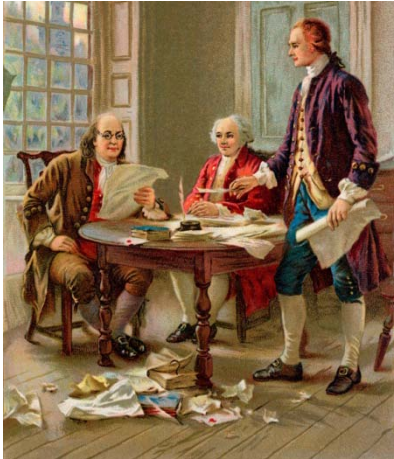
Earlier is 5x cheaper

Inspect all work products

- Requirements, designs, code, test cases, user manual, etc.
- If agile, have input into SCRUM sessions, build test cases, and assure functionality performs as expected

Use code and complexity analyzers

- Code size is directly proportional to # of test cases
- McCabe complexity identifies technical risk areas



A single requirements defect can cost over \$30,000 if found in production !!

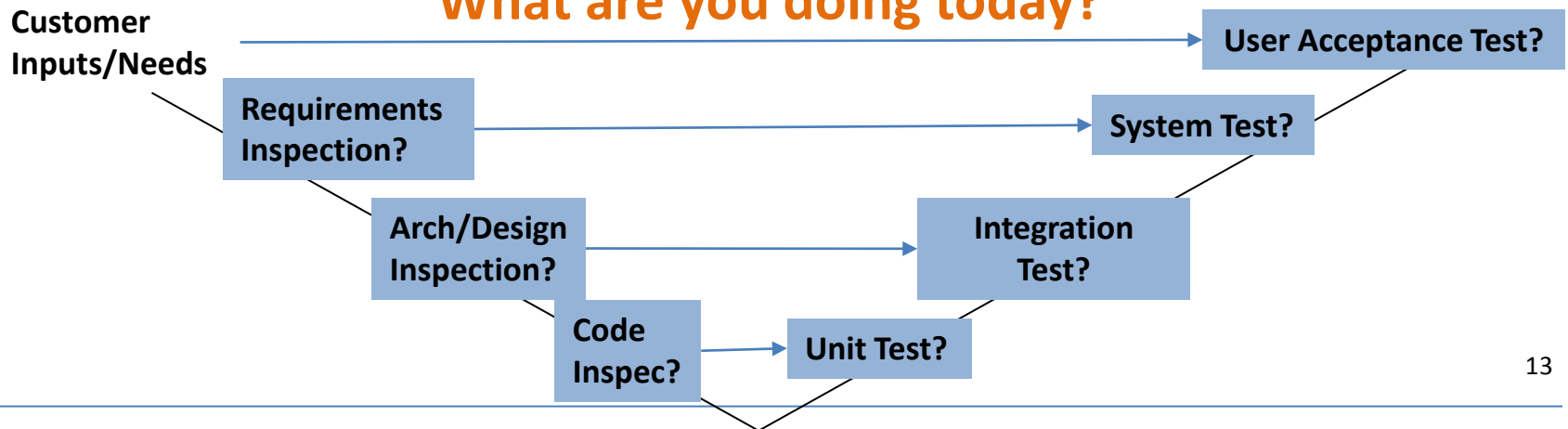
#2 – Find bugs earlier

Phase	Defects/ 1 KSLOC	Defect %
Reqmts	2.2	20%
Design	2.7	25%
Code	3.7	34%
Docs	1.2	11%
Fixes	1.1	10%
Total	10.9	

Note that 45% of defects occur before coding

*data adapted from Capers Jones (100K SLOC app),
Software Quality: Analysis and Guidelines for Success

What are you doing today?



(Competency + Motivation = Productivity)

Train your staff

- Use the right person for the right job
- Developers – basic QA + unit test training
- Users – basic QA training
- Testers should have extensive QA and test training

Use proficiency testing

- To establish baseline and as annual objectives

Motivate staff with incentives to meet and exceed productivity and quality

- “Win a TV”

Use the right person for the right job

Unit Testing

- Developers

Development Integration Testing

- Developers

System Testing

- Dedicated Test Team -- Internal or Outsourced
- Build repeatable test cases (i.e., don't require a SME to execute)

User Acceptance Testing

- Users + Dedicated Test Team

System Testing – Internal team or outsource?

Internal Team

- **Developers** - Expensive and bored – They want to write code
- **Users** - Have a day job and generally focus on “what it’s supposed to do”
- **Dedicated Test Team** - Good choice!!

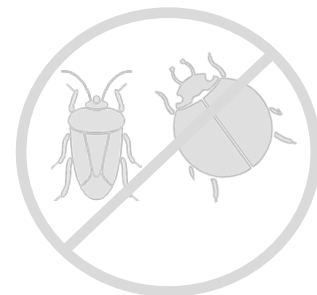
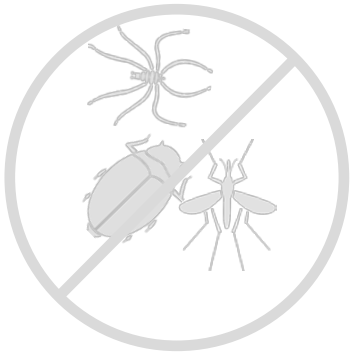
Outsourced Team considerations

- Product/business knowledge
- Dedication
- Ramp up and ramp down
- Test methodology
- Productivity, metrics, and transparency

Summary

Top five things you can do today

- 5 – Focus where the risk is
- 4 – Reduce churn
- 3 – Establish metrics and set goals
- 2 – Find bugs earlier
- 1 – Improve staff -- use professional testers



full lifecycle SOFTWARE SUCCESS

- Software Testing
- Software Quality Assurance / IV&V
- Oracle e-Business Suite

For more information, contact Jeff Van Fleet

jvanfleet@lighthousetechnologies.com

www.lighthousetechnologies.com

(937) 458-0055 x201