From State-of-Art to State-of-Practice
No Silver Bullet

Dottie Acton
Senior Fellow
Lockheed Martin

Topics

- The Wish: Silver Bullets
- The Reality: Change is Hard
- The Result
- Reducing the Gap
- Do These Things Now for Software
- Questions

Finds Bugs Faster!

Shortens Cycle Time!

Improves Productivity!

Eliminates Defects!

Improves Code Quality!

Automatically Generates Code!

Automates Tests!

Makes Inspections More Effective!

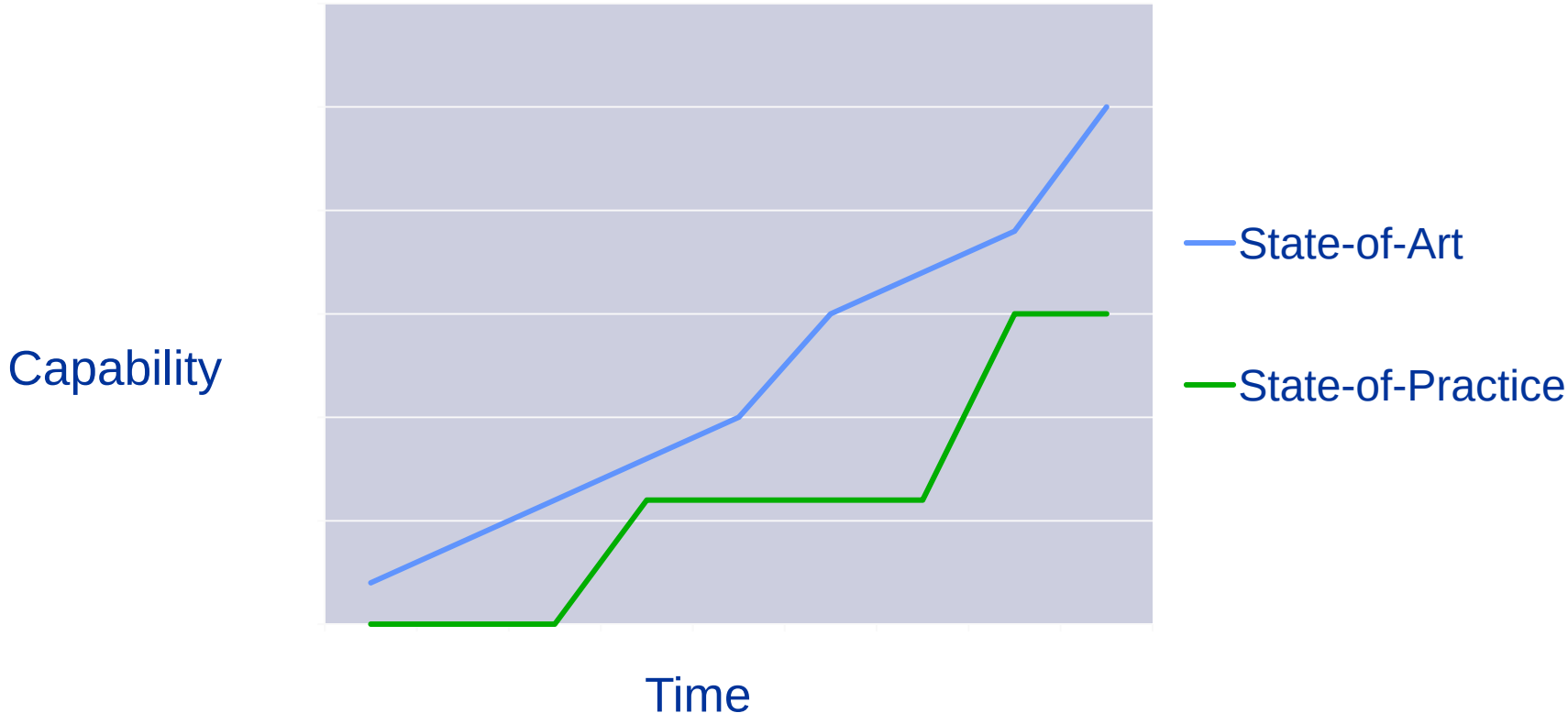
Improves ROI!

Lowers Cholesterol!

Impediments to Change

- **Existing infrastructure**
  - Large investment in tools and training
  - Large code base
- **Organizational Processes**
  - Mostly waterfall based
  - Document centric
  - Risk averse (lessons learned)
- **Human resistance to change**
  - Personality types
  - Fear of failure
  - Belief that entropy is inevitable
  - Easier to blame than to change
- **Program commitments**
  - We cannot let change impact our ability to meet customer expectations
- **Pace of change**
  - By the time we implement a change, it's obsolete
  - If we don't change, we become obsolete
- **Infinite possibilities**
  - The better capabilities we have, the harder the problems we try to solve

The Result



An Alternative

Focus on Evolutionary Change, not Revolutionary Change

Some Techniques for Evolutionary Change

- Introduce new ideas in a non-threatening way
  - Brown bag sessions, book clubs, presentations
  - Expose people to the possibilities
- Use pilots and experiments
  - Use pathfinders within projects
  - Use on IRADS and internal projects
  - Experiment with change for a defined period of time
    - Let teams decide about permanent adoption
- Celebrate success
  - Every small step is important
  - Celebrate ‘failures’ as successful learning
- Be persistent
  - It can take years to completely adopt a new approach

Do These Things Now for Software

Accept responsibility

Resolve to do no harm

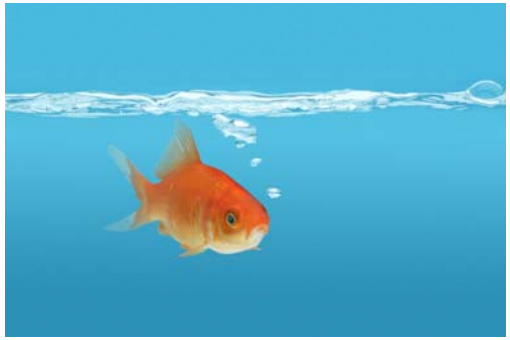
Reduce unnecessary dependencies

Automate verification

Accept Responsibility

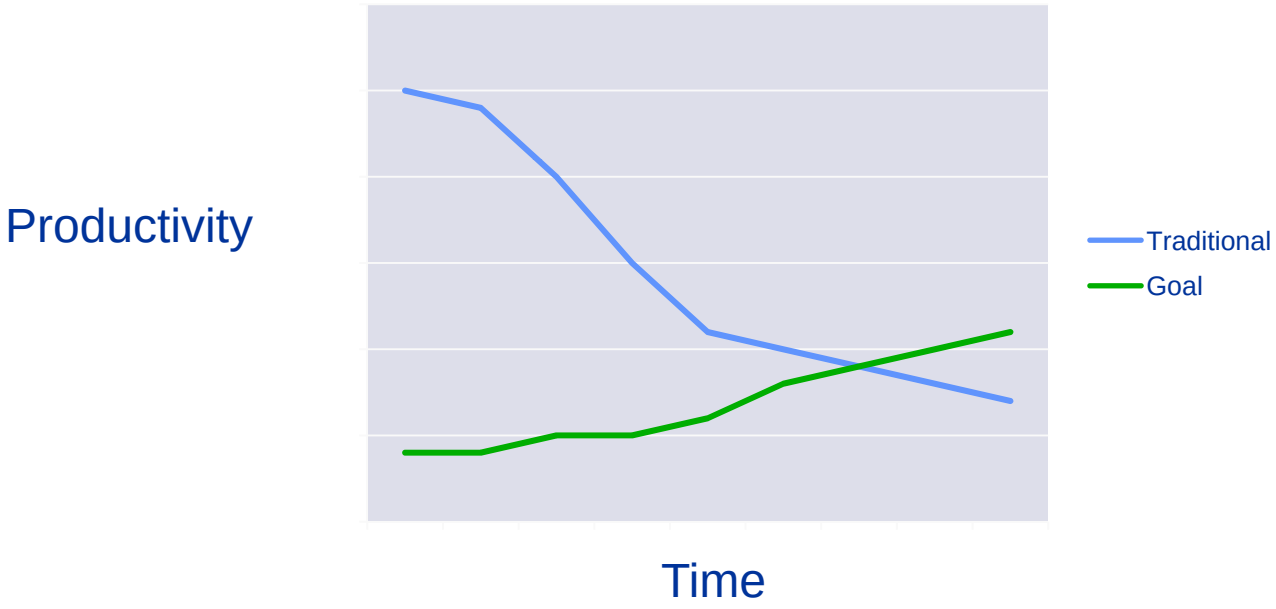
"If our designs are failing due to the constant rain of changing requirements, it is our designs that are at fault. We must somehow find a way to make our designs resilient to such changes and protect them from rotting."

Robert Martin



Resolve To Do No Harm

Any time you touch the code, make it better – or at least no worse



Change the economics of change!

Reduce Dependencies

Adopt a test-driven approach to software development

Write a small automated test Run the test It should fail Write the code to make the test pass Refactor (to patterns) to make the code clean Repeat	A 5-10 minute cycle
Integrate	At least daily

Benefits:

- Better interfaces
- Fewer dependencies
- Safety net for future changes
- Finds defects faster

Downside:

- Difficult if code or tests ‘smell’

Automate Verification

- Perform static analysis as part of code check-in
  - Ensures standards compliance
  - Provides immediate feedback to developers
    - Discourages bad coding practices
- Run automated unit tests before code is accepted into the baseline
  - All tests must pass every time
  - Detects inconsistencies and defects earlier
- Automate regression tests at higher levels of testing

Summary

- With the capabilities of today's free and/or commercial tools, there is no excuse for producing code that 'smells'
  - As software professionals, we must commit to quality
- Adopting the newer approaches can be evolutionary
  - Just start today, with the code you are writing today
    - Apply patterns to eliminate dependencies
    - Refactor to keep code clean
    - Automate unit testing to create a safety net for change
- Learning is a journey
  - Keep on investing in your career