Presentation

Bio
Conference-at-a-Glance
Return to Main Menu

WG1

Wednesday, October 31, 2001
9:15 AM

NO BEST PRACTICES: HOW TO BEST THINK ABOUT METHODOLOGY

James Bach
Satisfice, Inc.

International Conference On
Software Testing Analysis & Review
October 29-November 2, 2001
San Jose, CA USA

No Best Practices:
How to Think About Methodology

James Bach, Satisfice, Inc.

James@satisfice.com

www.satisfice.com

best (best) *adjective*
Superlative of **good**

1. Surpassing all others in excellence, achievement, or quality; most excellent: *the best performer; the best grade of ore.*

2. Most satisfactory, suitable, or useful; most desirable: *the best solution; the best time for planting.*

3. Greatest; most: *He spoke for the best part of an hour.*

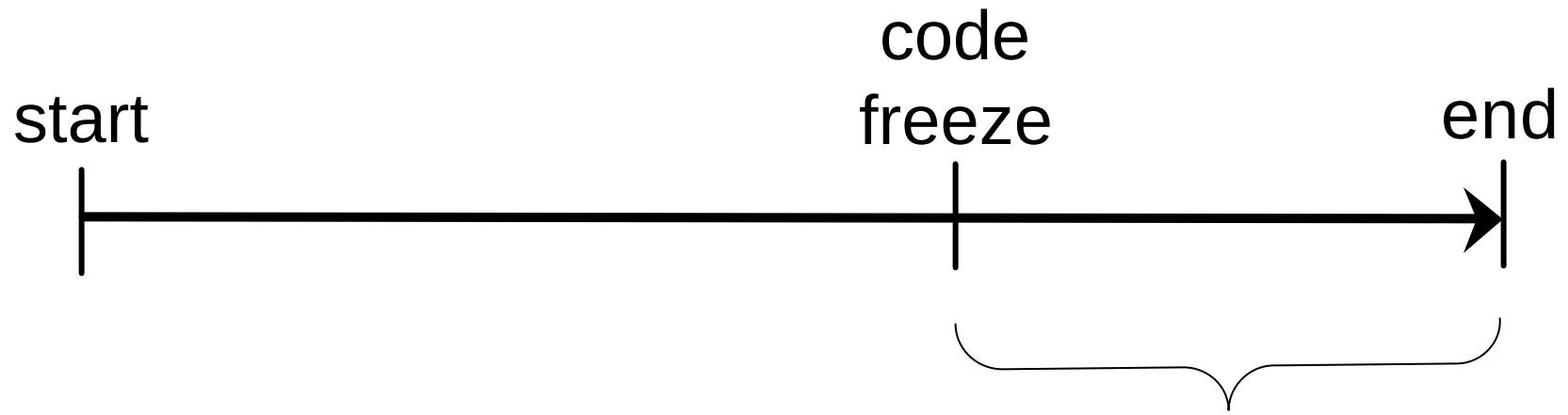
prac·tice (prak' tis)
noun

- A habitual or customary action or way of doing something: *makes a practice of being punctual.*
  - Repeated performance of an activity in order to learn or perfect a skill: *Practice will make you a good musician.*
  - A session of preparation or performance undertaken to acquire or polish a skill: *goes to piano practice weekly; scheduled a soccer practice for Saturday.*
  - *Archaic.* The skill so learned or perfected.
  - The condition of being skilled through repeated exercise: *out of practice.*
- The act or process of doing something; performance or action: *a theory that is difficult to put into practice.*
- Exercise of an occupation or profession: *the practice of law.*
- The business of a professional person: *an obstetrician with her own practice.*
- A habitual or customary action or act. Often used in the plural: *That company engages in questionable business practices. Facial tattooing is a standard practice among certain peoples.*
- *Law.* The methods of procedure used in a court of law.
- *Archaic.*
- The act of tricking or scheming, especially with malicious intent.
- A trick, scheme, or intrigue.

Can we agree on this?

- There are no “practice” championships; no Olympics to determine which practices to honor.
- There is no consensus about what practices are best, unless consensus means “people I respect also say they like it.”
- There *are* practices that are more likely to be considered good and useful than others, within a certain community and assuming a certain context.
- But… So what? *Good practice is not a matter of popularity. It’s a matter of skill and context.*

“Code Freeze”
A Best Practice?

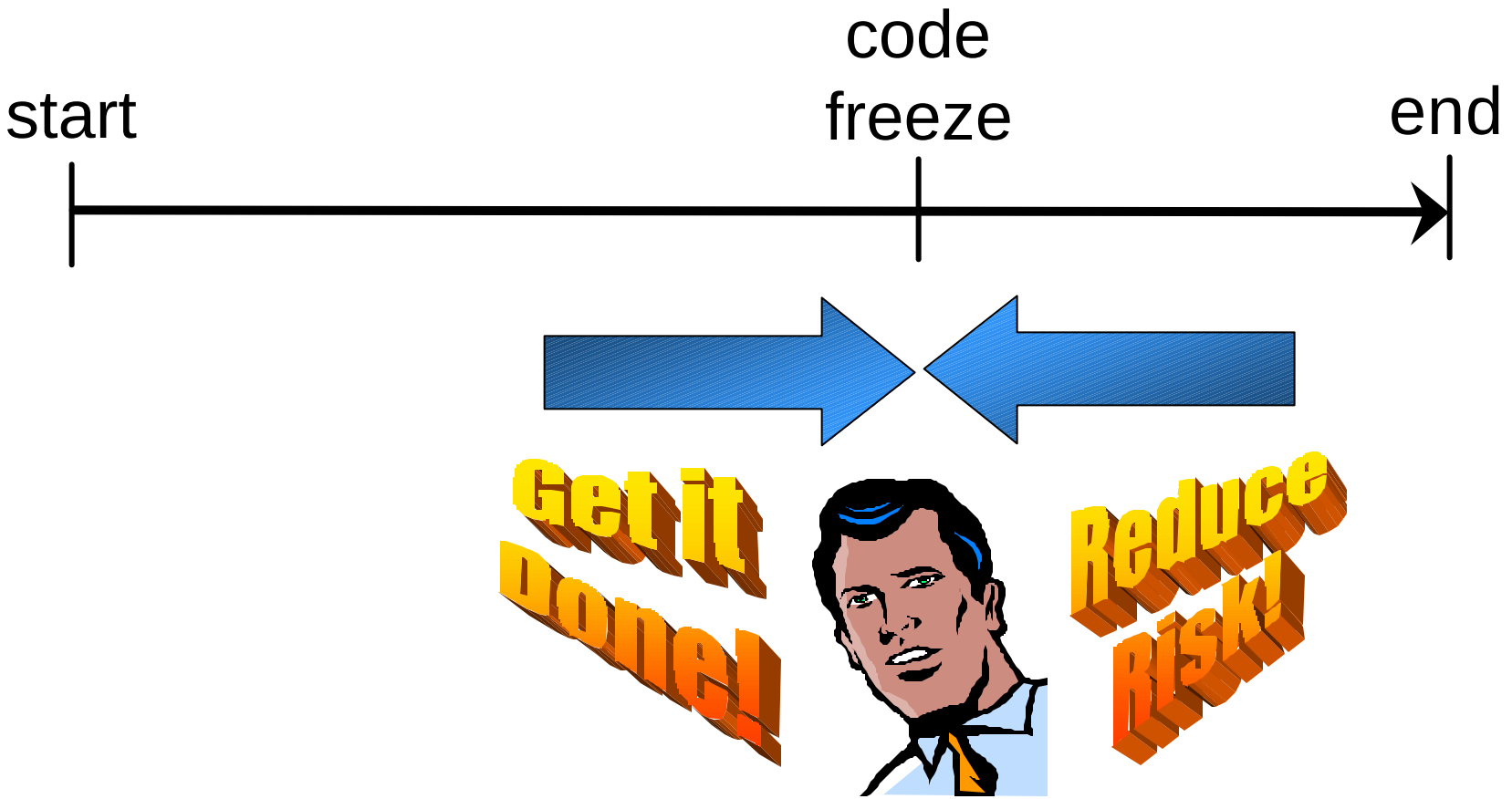


Change board reviews all changes

Pros: reduces testing load and the probability of new bugs.

Cons: creates review workload and reduces pace of improvement.

“Code Freeze”
A Best Practice?



Methodology Equivocation

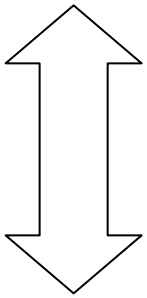
Beware of the "process" switcharoo

source: Total Quality Management for Software, Schulmeyer & McManus

Process is what happens

"This transformation [from inputs to outputs] is called a process." p.120

"The focus of TQM for software is on controlling and improving the subprocesses *represented by* the various software development phases." p. 123



Process is what we *say* happens

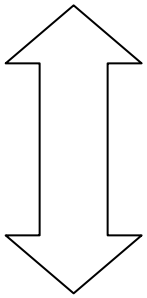
"The important thing is to have paradigm that forces a top down structure on the process definition task." p. 145

"For all practical purposes the following rule applies: Any part of the development process that is not clearly documented does not exist." p.145

Methodology Equivocation

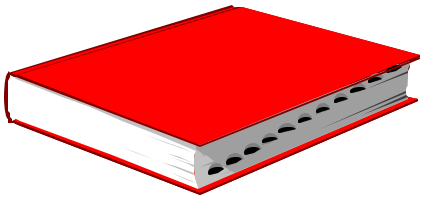
A "best practice" may not even be practiced.

Process is what happens

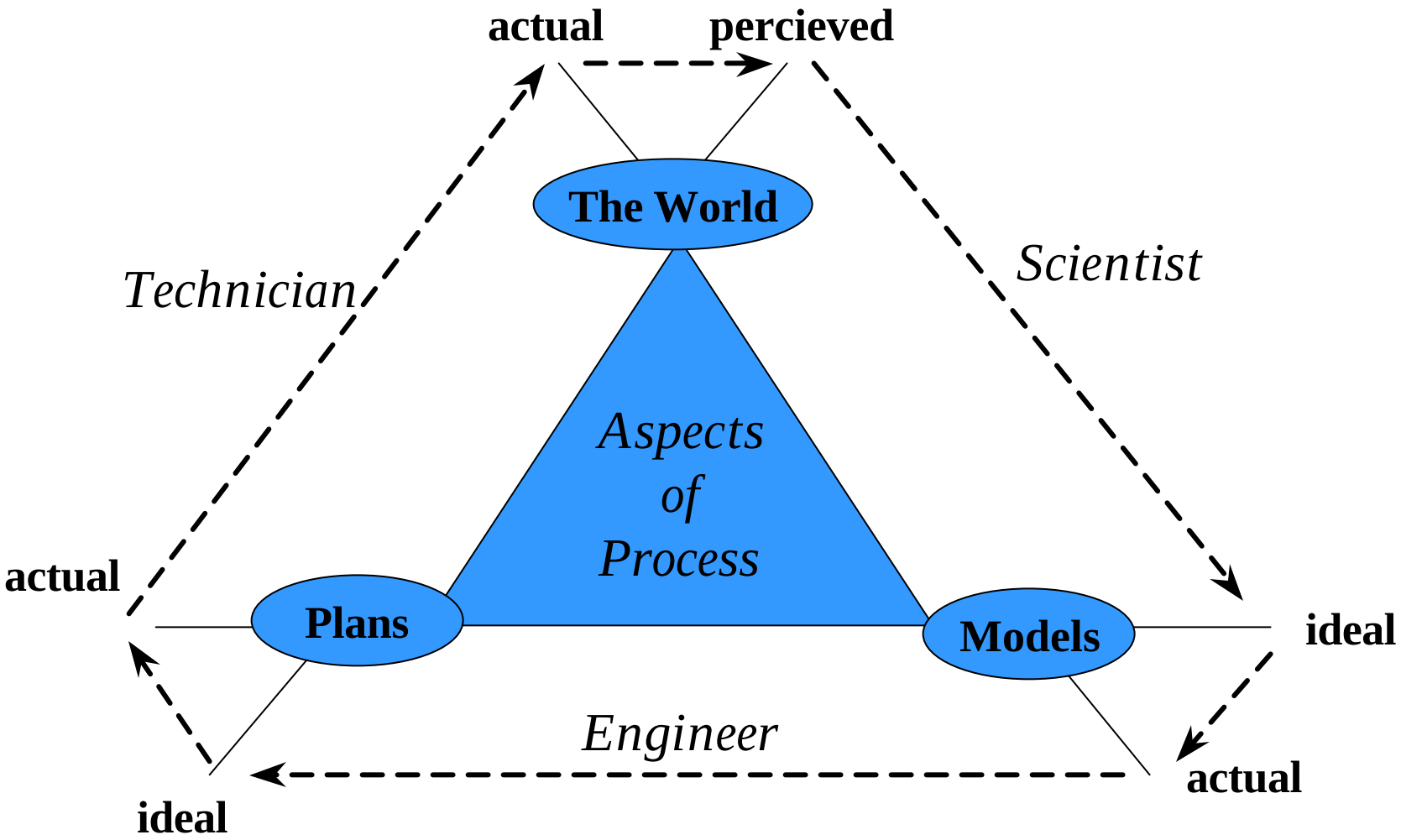


Process is what we *say* happens

There is a large gap between what we can do and what we can say about it.

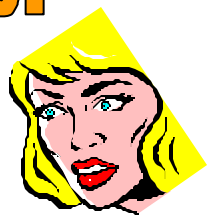


How do things happen?



What does “best practice” mean?

Someone believes you might suffer unless you do this practice



- **Someone:** Who is it? What do they know?
- **Believes:** What specifically is the basis of their belief?
- **You:** Is *their* belief applicable to *you*?
- **Might:** How *likely* is the suffering to occur?
- **Suffer:** So what? Maybe it’s worth it?
- **Unless:** Really? There’s no alternative?
- **You do this practice:** What does it mean to “do” it? What does it cost? What are the side effects? What if you do it badly? What if you do something else really well?

Beware of...

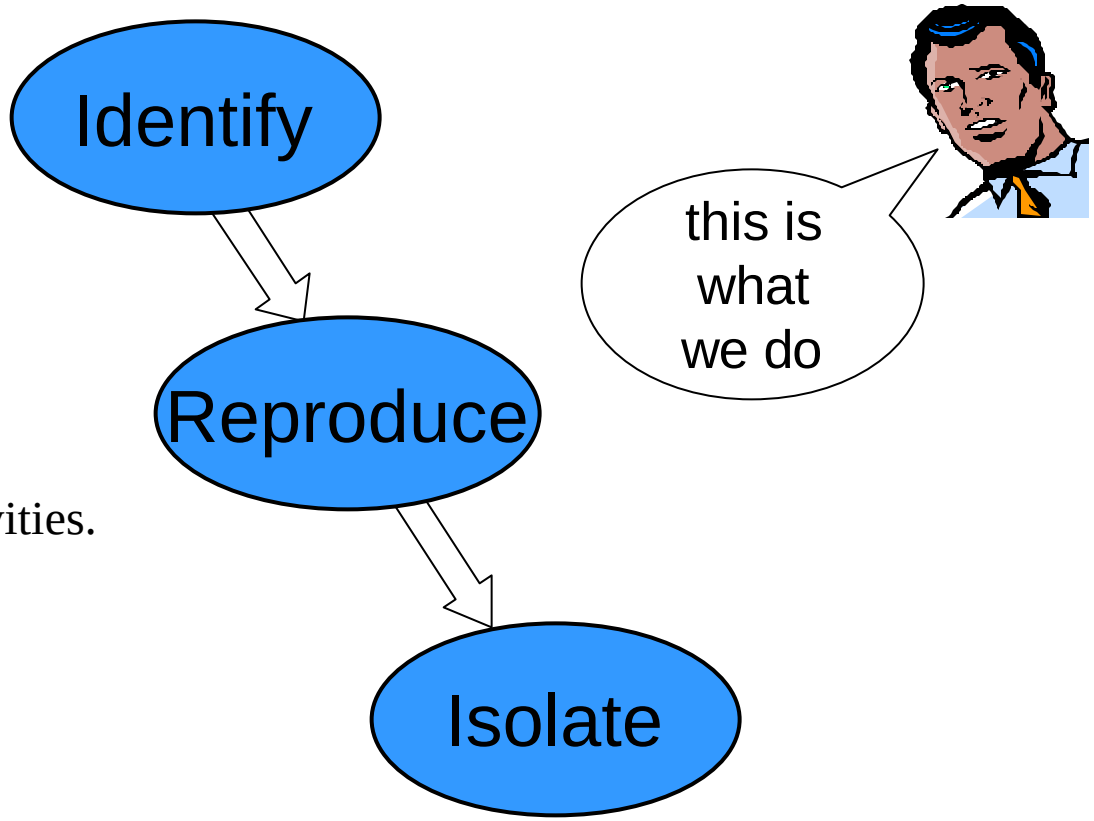
- **Numbers:** "We cut test time by 94%."
- **Documentation:** "You must have a written plan."
- **Judgments:** "That project was *chaotic*. This project was a *success*."
- **Behavior Claims:** "Our testers follow test plans."
- **Terminology:** Exactly what is a "test plan?"
- **Contempt for Current Practice:** CMM Level 1 (initial) vs. CMM level 2 (repeatable)
- **Unqualified Claims:** "A subjective and unquantifiable requirement is not testable."

Look For...

- **Context:** "This practice is useful when you want the power of creative testing but you need high accountability, too."
- **People:** "The test manager must be enthusiastic and a real hands-on leader or this won't work very well."
- **Skill:** "This practice requires the ability to tell a complete story about testing: coverage, techniques, and evaluation methods."
- **Learning Curve:** "It took a good three months for the testers to get good at producing test session reports."
- **Caveats:** "The metrics are useless unless the test manager holds daily debriefings."
- **Alternatives:** "If you don't need the metrics, you ditch the daily debriefings and the specifically formatted reports."
- **Agendas:** "I run a testing business, specializing in exploratory testing."

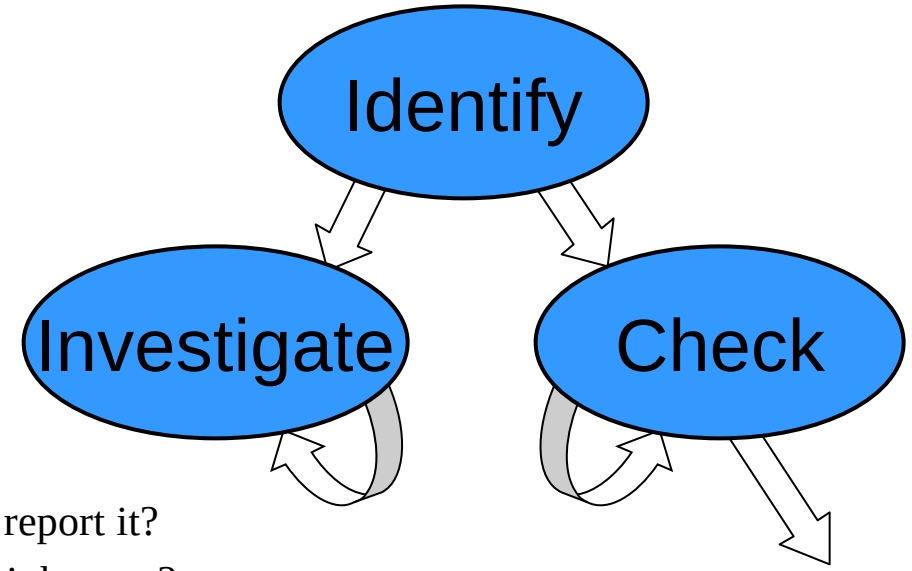
Bug Investigation Practice
(reported)

- Identify
  - Notice a problem.
- Reproduce
  - Make it happen again.
- Isolate
  - Cut out non-essential activities.



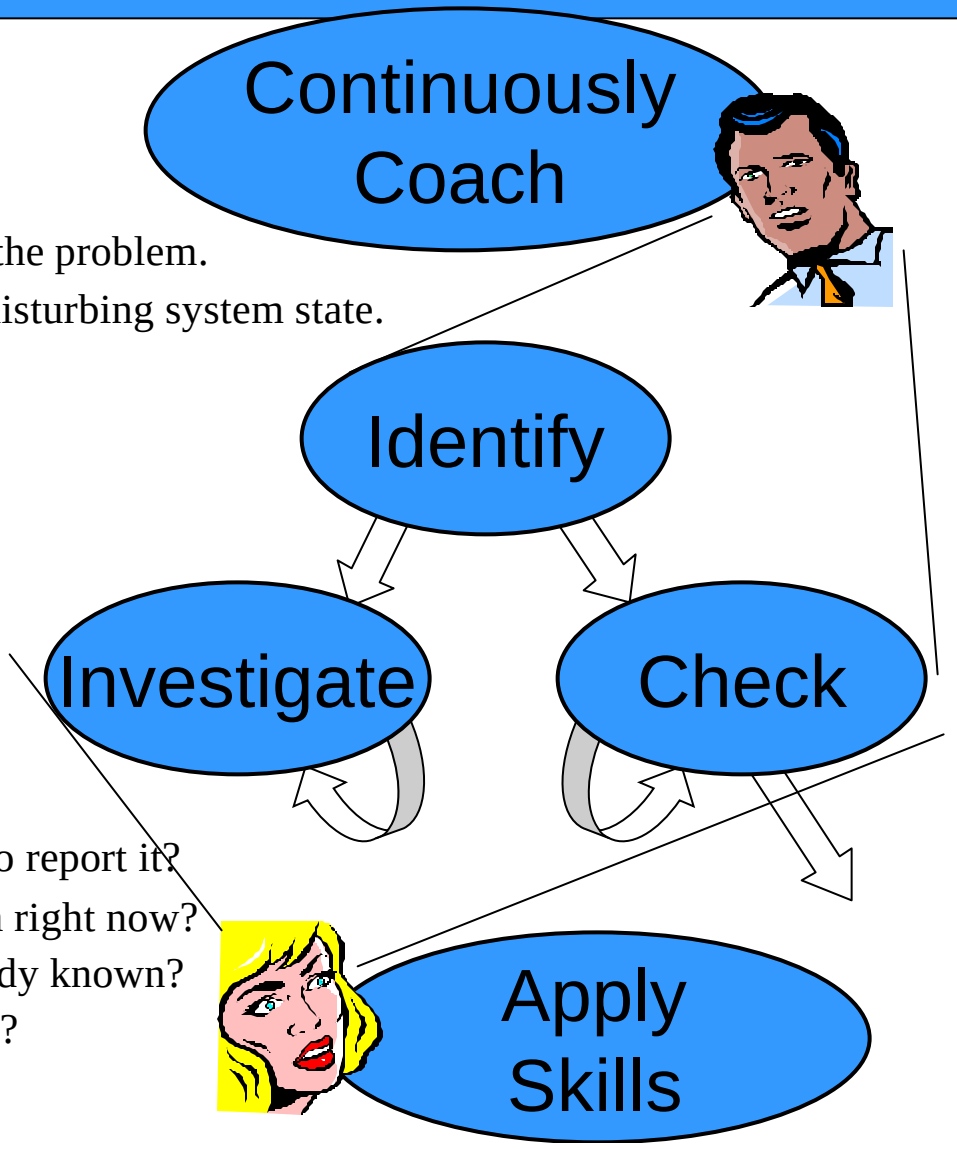
Bug Investigation Practice
(more accurate)

- Identify
  - Notice a problem.
  - Recall what you were doing just prior to the problem.
  - Examine symptoms of the problem w/o disturbing system state.
  - Consider possibility of tester error.
- Investigate
  - How can the problem be reproduced?
  - What are the symptoms of the problem?
  - How severe could the problem be?
  - What might be causing the problem?
- Reality Check
  - Do we know enough about the problem to report it?
  - Is it important to investigate this problem right now?
  - Is this problem, or any variant of it, already known?
  - How do we know this is really a problem?
  - Is there someone else who can help us?



Bug Investigation
(even more accurate, but hard to formalize)

- Identify
  - Notice a problem.
  - Recall what you were doing just prior to the problem.
  - Examine symptoms of the problem w/o disturbing system state.
  - Consider possibility of tester error.
- Investigate
  - How can the problem be reproduced?
  - What are the symptoms of the problem?
  - How severe could the problem be?
  - What might be causing the problem?
- Reality Check
  - Do we know enough about the problem to report it?
  - Is it important to investigate this problem right now?
  - Is this problem, or any variant of it, already known?
  - How do we know this is really a problem?
  - Is there someone else who can help us?



Context-Driven Testing Principles

1. The value of any practice depends on its context.
2. There are good practices in context, but there are no best practices.
3. People, working together, are the most important part of any project's context.
4. Projects unfold over time in ways that are often not predictable.
5. The product is a solution. If the problem isn't solved, the product doesn't work.
6. Good software testing is a challenging intellectual process.
7. Only through judgment and skill, exercised cooperatively throughout the entire project, are we able to do the right things at the right times to effectively test our products.

source: Lessons Learned in Software Testing, by Kaner, Bach, Pettichord

James Bach

I started in this business as a programmer. I like programming. But I find the problems of software quality analysis and improvement more interesting than those of software production. For me, there's something very compelling about the question "How do I know my work is good?" Indeed, how do I know anything is good? What does good mean? That's why I got into SQA, in 1987.
Today, I work with project teams and individual engineers to help them plan SQA, change control, and testing processes that allow them to understand and control the risks of product failure. I also assist in product risk analysis, test design, and in the design and implementation of computer-supported testing. Most of my experience is with market-driven Silicon Valley software companies like Apple Computer and Borland, so the techniques I've gathered and developed are designed for use under conditions of compressed schedules, high rates of change, component-based technology, and poor specification.