

# Skilled Testers and Their Enemies

James Bach, Satisfice, Inc.

James@[satisfice.com](mailto:James@satisfice.com)

[www.satisfice.com](http://www.satisfice.com)

# That was "Quick Testing" which is *one essential tactic* of good testing

*When testing is turned into an elaborate set of rote tasks,  
it becomes ponderous without really being thorough.*

*Management likes to talk  
about thorough testing, but  
they don't want to fund it and  
they don't know how to do it.*

Cost vs. Value

*"Rapid testing" is  
**thorough enough**  
and **quick enough**.  
It fulfills the mission  
of testing.*

<b>Thorough</b> <i>Slow, very expensive, and difficult</i>	<b>Ponderous</b> <i>Slow, expensive, and easier</i>
<b>Rapid</b> <i>Faster, less expensive, still challenging</i>	<b>Slapdash</b> <i>Much faster, cheaper, and easier</i>

More Work

Better Thinking & Better Testing

*You can always test quickly...  
But it might be poor testing.*

# I teach testing as a sort of martial art.

---

***Testing*** is questioning a product in order to evaluate it.

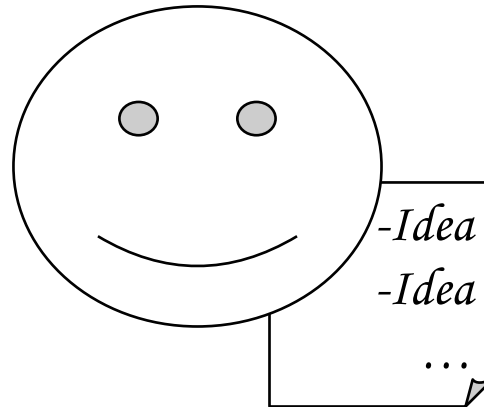
- The “questions” consist of various ways of *configuring* and *operating* the product. (Asking questions about the product without operating it is usually called review rather than testing)
- The product “answers” by exhibiting behavior, which the tester *observes* and *evaluates*.
- To evaluate a product is to infer from its observed behavior how it will behave in the field, and to identify important problems in the product.

# My approach is not procedural.

## *Skill + Heuristics*

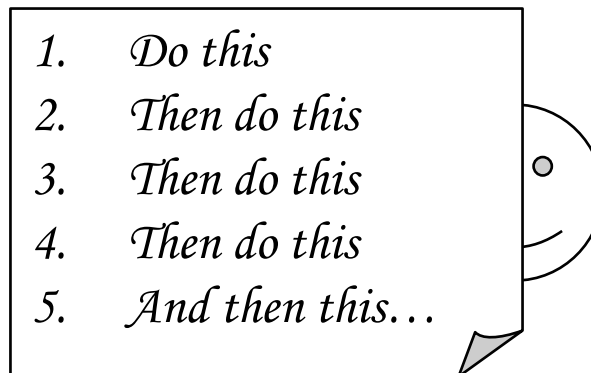
---

This...



A heuristic is a fallible method for solving a problem.

...not this.

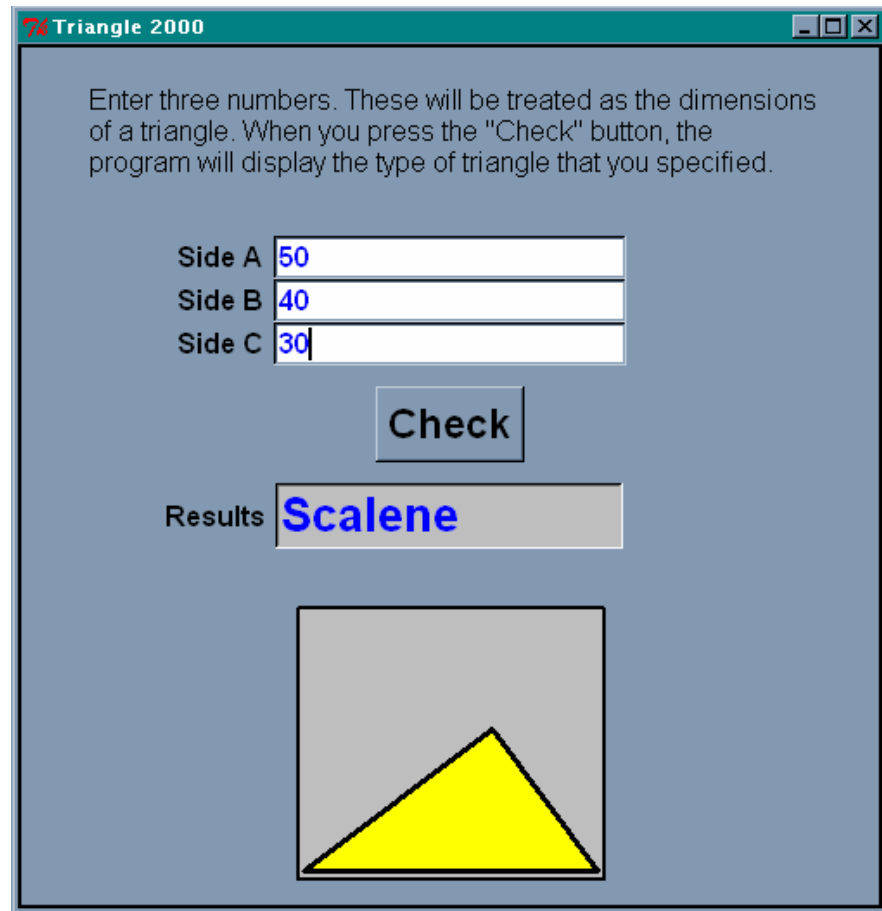


Hey! Testing is not a clerical process.

# Why Procedures Can't Suffice: Input Constraint Testing

Students in my *Rapid Software Testing* class are given 20 minutes to test this program.

We see interesting differences in how testers approach this task.



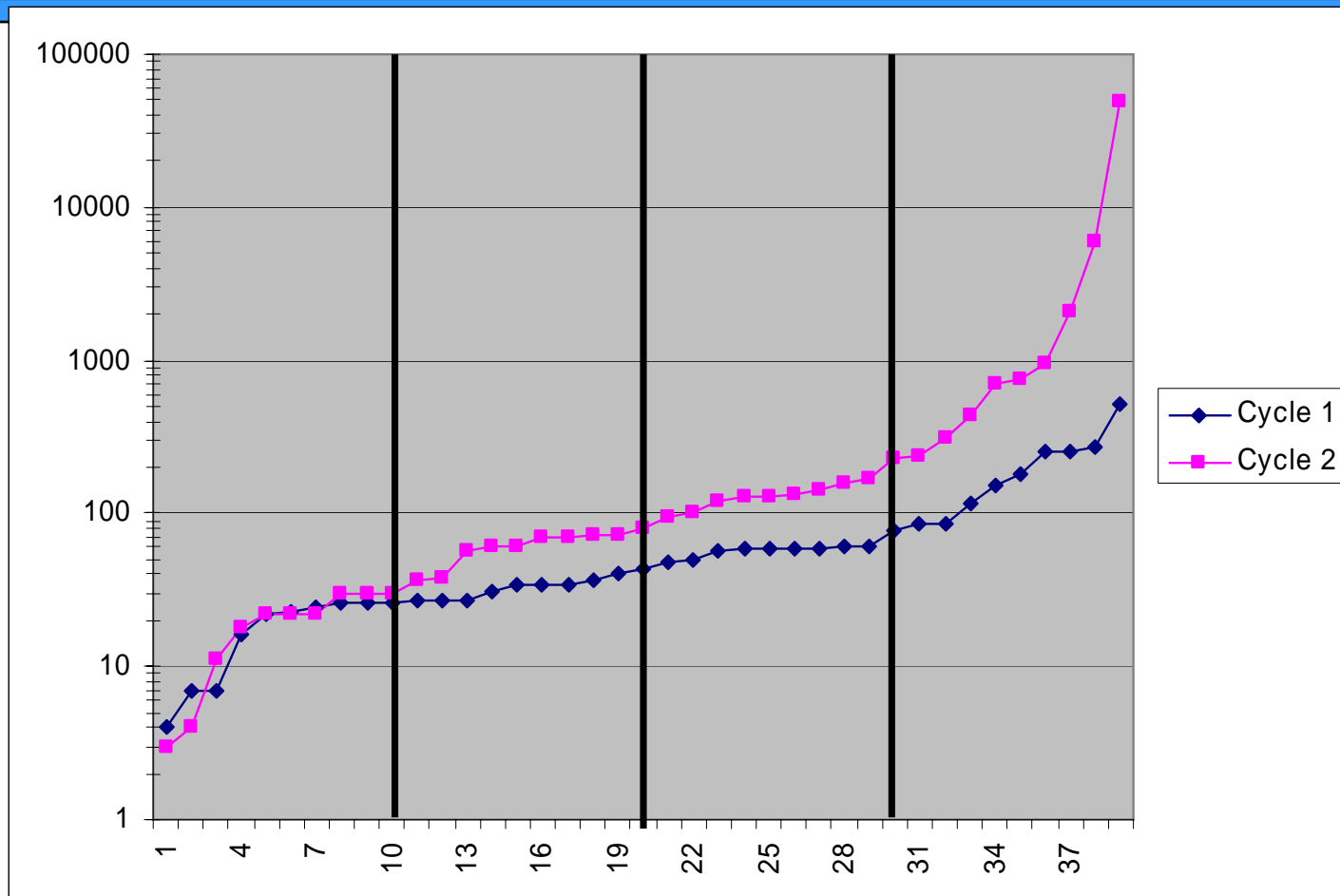
# How Well Do the Fields Handle Long Inputs?

---

Side A	50
Side B	40
Side C	30

- What does “long” mean?
- What does “handle well” mean?
- What will users do? What will they expect?
- So what?

# Max String Sizes Chosen by 39 Testers (*no limit was specified*)



For cycle two I specifically asked students to try long inputs

# Interesting Lengths

---

- **16** digits & up: loss of mathematical precision.
- **23** digits & up: can't see all of the input.
- **310** digits & up: input not understood as a number.
- **1,000** digits & up: exponentially increasing freeze when navigating to the end of the field by pressing <END>.
- **23,829** digits & up: all text in field turns white.
- **2,400,000** digits: crash (reproducible).

*Only the first two boundaries were known to the programmer.*

# What stops testers from trying longer inputs?

---

- Seduced by what's visible.
- Think they need a spec that tells them the max.
- If they have a spec, they stop when the spec says stop.
- Satisfied by the first boundary (16 digits).
- Let their fingers do the walking instead of using a program like notepad to generate input.
- Use strictly linear lengthening strategy.
- Don't realize the significance of degradation.
- Assume it will be too hard and take too long.
- Think "No one would do that" (hackers do it)

# Five Schools of Testing

## *The Breakthrough Schools*

---

- The Analytical School

- “Testing is the math problem I know how to solve within a specified micro-world. My work *could* lead to a profound breakthrough... *further research is needed.*”

- The Quality Control School

- “Testing? I am a *quality engineer*. Have you heard the good news about *prevention*? Zero defects is my motto! Who’s up for a fishbone diagram?

Hey, come back!”

# Five Schools of Testing

## *The Pragmatic Schools*

---

- **The Factory School** (*big*)
  - “It is important to minimize process variation by *restricting the human factor*. Therefore, testing is better when it is expressed and controlled through artifacts according to documented, detailed and unambiguous procedures.”
- **The Context-Driven School** (*small*)
  - “Good testing is so challenging to do, and varies so much with the context of the project, that consistently good testing requires that we *exploit the human factor*, not restrict it. Grow the skills of testers, and give them the authority and resources to solve the problem in the most appropriate ways.”
- **The “Oblivious” School** (*huge*)
  - “What? Just test it.”

# Agile and Traditional versions of Factory Testing

---

## ■ **Traditional Factory Testing**

- Conflates testing with “test cases.”
- Does not grow the skills of testers.
- Embraces testing mainly as a *clerical* role.
- Considers testing to be a straightforward, simple task.
- Tests what is easy to write down, rather than what is most important to test.
- Testing is conveniently outsourceable to stupid people.

## ■ **Agile Development** (*Extreme Programming*)

- Conflates testing with “test cases.”
- Does not grow the skills of testers, but very interested in programmer skill.
- Rejects testing as a role.
- Considers testing to be a straightforward, simple task.
- Tests what is easy to automate, rather than what is most important to test.
- Testing is conveniently outsourceable to tools.

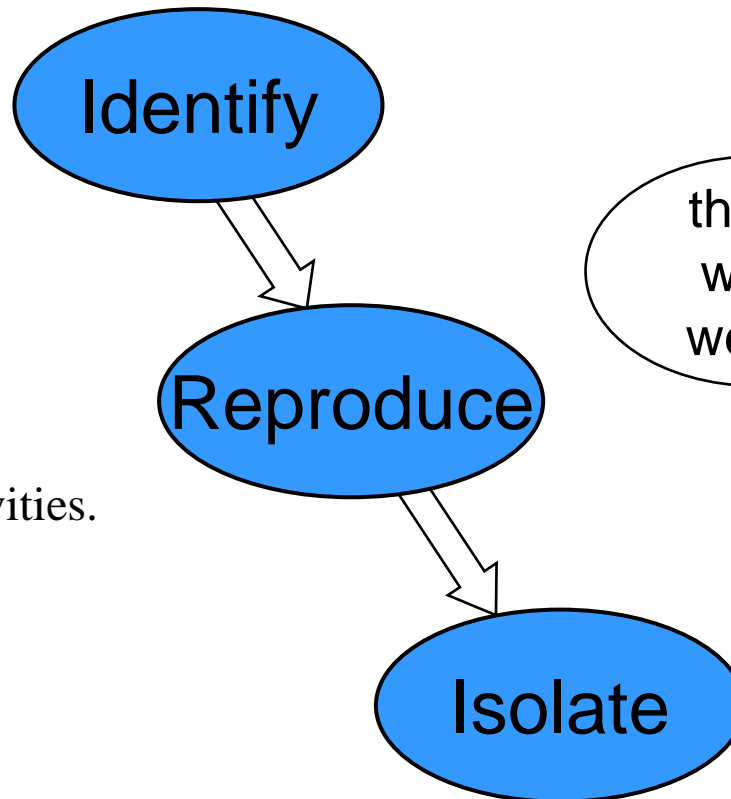
# Context-Driven Testing Principles (v1.0)

---

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.

# Case: Skilled Bug Investigation (*reported*)

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



# Skilled Bug Investigation (*more accurate*)

## ■ Identify

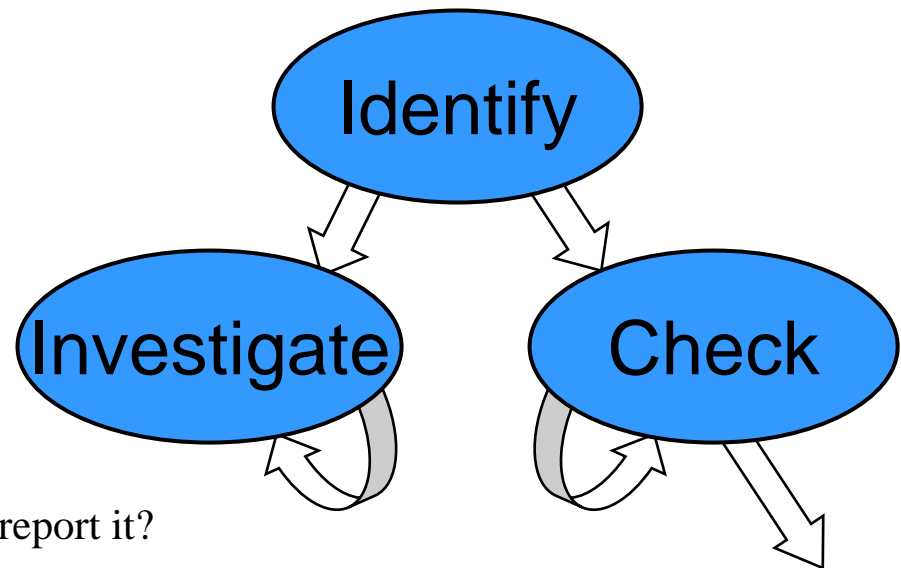
- Notice a problem.
- Recall what you were doing just prior to the problem.
- Examine symptoms of the problem w/o disturbing system state.
- Consider possibility of tester error.

## ■ 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?



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

## ■ Identify

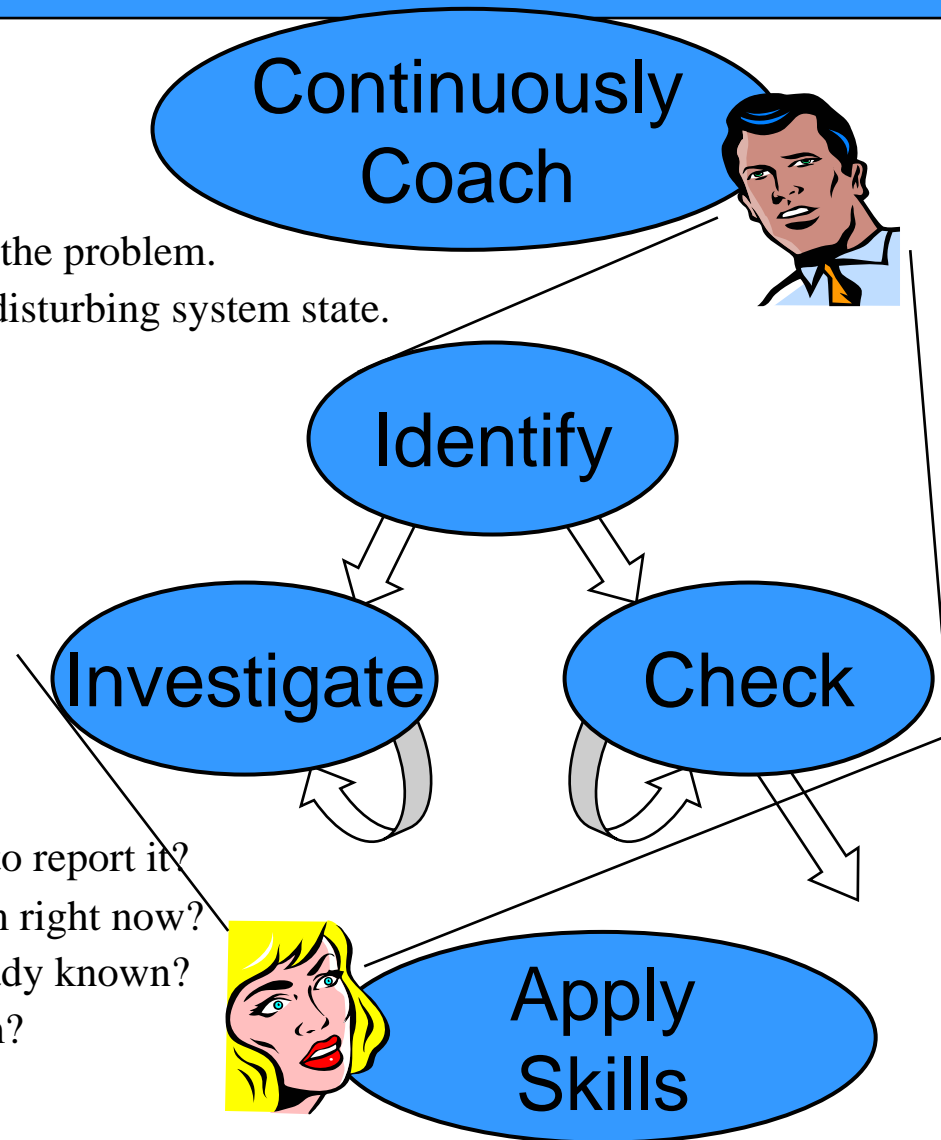
- Notice a problem.
- Recall what you were doing just prior to the problem.
- Examine symptoms of the problem w/o disturbing system state.
- Consider possibility of tester error.

## ■ 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?



# Surface Skills I Identified

---

- Skeptical questioning
- Performing an open investigation
- Understanding external and internal product functionality
- Consulting with developers or other testers
- Test design
- Exploratory testing
- Noticing problems
- Assessing problem severity
- Identifying and using technical documentation
- Recording and maintaining information about problems
- Identifying and using tools
- Identifying similar known problems
- Managing simultaneous investigations
- Issue escalation

# I think Those Surface Skills are Based on This Essential Skill Set

---

## *TESTING SKILL CONSISTS OF THE SKILLS OF*

- General Systems Thinking
- Critical Thinking & Research Methods
- Rapid & Exploratory Learning

## *PLUS KNOWLEDGE OF*

- Cognitive Science & Human Factors
- Decision & Utility Theory
- Folklore & Heuristics (most testing textbooks are folklore)

## *PLUS*

- Context-Specific Technology and Social Skills

# Tools and Skills

---

- All testing schools want better tools. The difference is that the context-driven school uses tools in a *supporting* role.
- Skilled testers need tools to extend and supercharge their work.
  - Probes to make visible the invisible (*Filemon*)
  - Logging tools to help them remember (*Spector*)
  - Control tools (*COM+ interface*)
  - Reverse Engineering Tools (*Restorator*)
  - Test Generators (*model-based testing*)

# How I Teach Testing Skills

---

- Immersive Exercises with Socratic Debrief
  - Some are highly authentic, natural problems.
  - Some are exaggerated or simplified to focus on an issue.
  - Some are done in pairs, others individually.
  - Performance is compared among students.
  - Many can be done remotely, but still require instructor interaction.
  - Coaching during a real project seems to work best.
  - I encourage students to argue with me and question everything.
- The experience of being fooled fairly helps *open minds*.
- The experience of escaping a trap that catches other students is *motivating*.
- Diverse exercises help build a general mental schema.

# The Future

---

- More LAWST meetings to share experience reports. (Almost fifty of these three-day, facilitated peer conferences have been held since 1997)
- Professional connections with magicians, cognitive scientists, criminologists and other disciplines.
- One of those conferences is the annual Workshop on Training Software Testers, hosted at Florida Tech to help university professors become better at teaching testers.
- Cem Kaner and the contributing researchers at Florida Tech (including me) are putting together free self-education courseware for testers. Much is already online at **[www.testingeducation.org](http://www.testingeducation.org)**
- *I believe my community knows how to train testers, today.* The next stage, for us, is to develop methods of training the people who train testers.