Self-Education for Testers

James Bach, Satisfice, Inc.

james@satisfice.com

www.satisfice.com

(540)631-0600

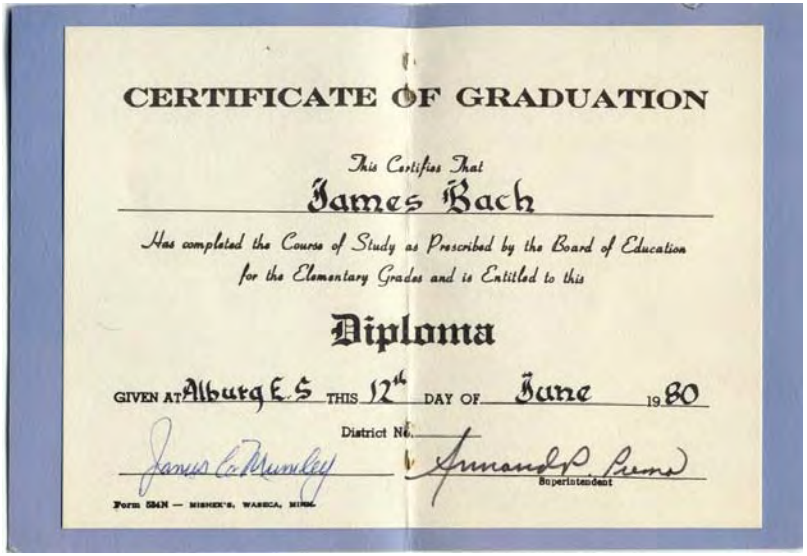
This is what I'm talking about

Self-education means
directing your own education*

* In the normal course of your life and work, and outside the confines of a learning institution.

- You choose what to learn.
- You decide when you learn.
- You control how you learn.
- You develop your own learning resources.

These are my credentials



CERTIFICATE OF GRADUATION

This Certifies That

James Bach

Has completed the Course of Study as Prescribed by the Board of Education for the Elementary Grades and is Entitled to this

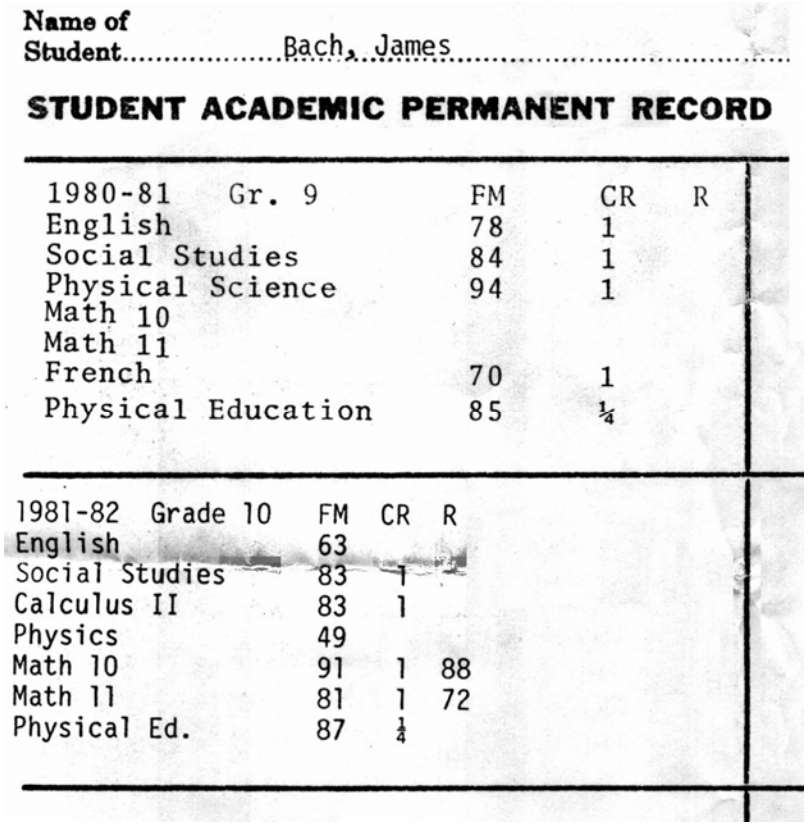
Diploma

GIVEN AT Alburg E S THIS 12th DAY OF June 1980

District No.

Superintendent

Form 5M — MINER'S, WABASCA, MINN.



Name of Student Bach, James

STUDENT ACADEMIC PERMANENT RECORD

1980-81 Gr. 9	FM	CR	R
English	78	1	
Social Studies	84	1	
Physical Science	94	1	
Math 10			
Math 11			
French	70	1	
Physical Education	85	¼	

1981-82 Grade 10	FM	CR	R
English	63		
Social Studies	83	1	
Calculus II	83	1	
Physics	49		
Math 10	91	1	88
Math 11	81	1	72
Physical Ed.	87	¼	

- I also have:
  - Level 1 paraglider pilot certification
  - PADI open water diver certification
  - Driver's license (state of Virginia)
  - Student private pilot license *(expired)*
  - Motorcycle license *(expired)*

What's Special About Testing

- There are few people around to teach you how to test.
- Most of what is taught as "testing" is unreliable or misleading folklore.
- Testing is a complex problem-solving activity.
- Learning testing on your own doesn't cost you much, you don't need anyone's permission, and it generally poses no threat to life or property.

However…

- It's hard to know if you are doing it well.
- Good testing varies quite a lot with the context.

What's Special About Testing

Testing IS learning!

...and unlearning what ain't so.

Self-Education:
Choosing What and When

- I learn whatever seems interesting about what's happening now.
- I scout resources and learn whatever looks helpful.
- When I need to solve an important problem, or look back on one, I learn whatever will help me solve it.
- When I screw up, I study my failure.
- I play games and learn what helps me win.
- I choose whom I respect, and then learn what gains me status among them.
- I learn things inspired by or recommended by someone else who seems interesting to me.
- I study whatever resolves my confusion about the world.
- I maintain a list of topics and skills that I believe will make me better, then learn about something within my personal syllabus.
- I study whatever helps me learn better.

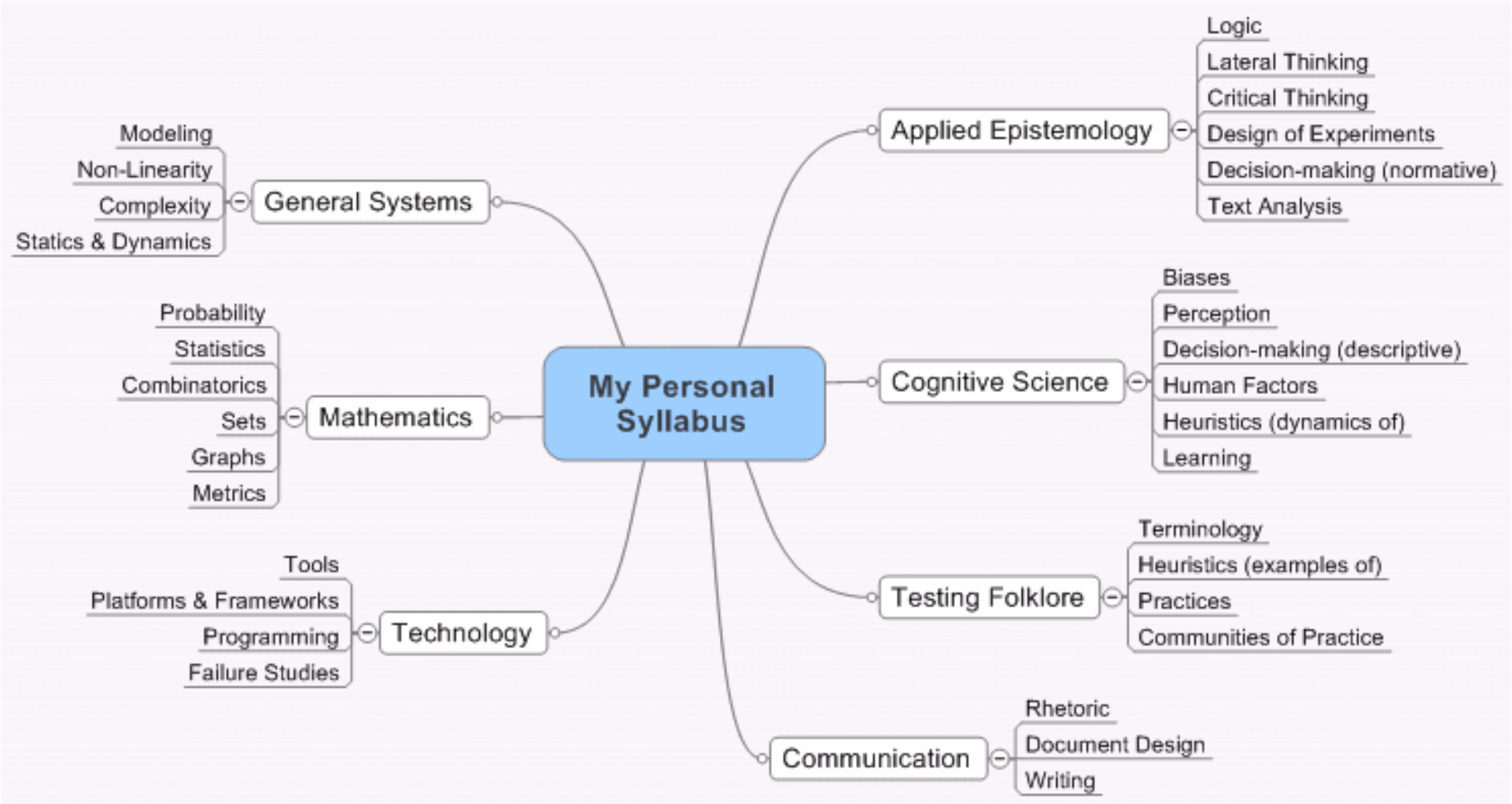
Self-Education:
Choosing What and When

- What's happening now?
- Are there books or references here?
- Am I facing an important problem now?
- Have I just screwed something up?
- Is this a game I want to win?
- Do I feel like impressing [Cem, Michael, etc.] today?
- What about that book [Cem, Michael, etc.] recommended?
- Am I confused right now?
- What do I know about testing?
- Can I get an edge by learning to learn better?

From My Personal Syllabus:
Important Things to Learn About Testing

- The key idea behind each “buzzword” test technique I might be asked about.
- A precise, expressive testing vocabulary for my philosophy of testing.
- The ability to speak in precise technical terms about my testing, and to articulate a story about my thought processes.
- How to analyze anything (now!)
- How to distinguish practices from contexts, and understand their relationship.
- Descriptive and normative aspects of observation and inference.
- Basic probability, statistics, and combinatorics.
- How to design cognitive artifacts.
- How to design and use heuristics.
- My personal collection of testing heuristics.
- Helpful software tools.
- A smattering of many different technologies.
- The Perl language.

My (Partial) Testing Syllabus

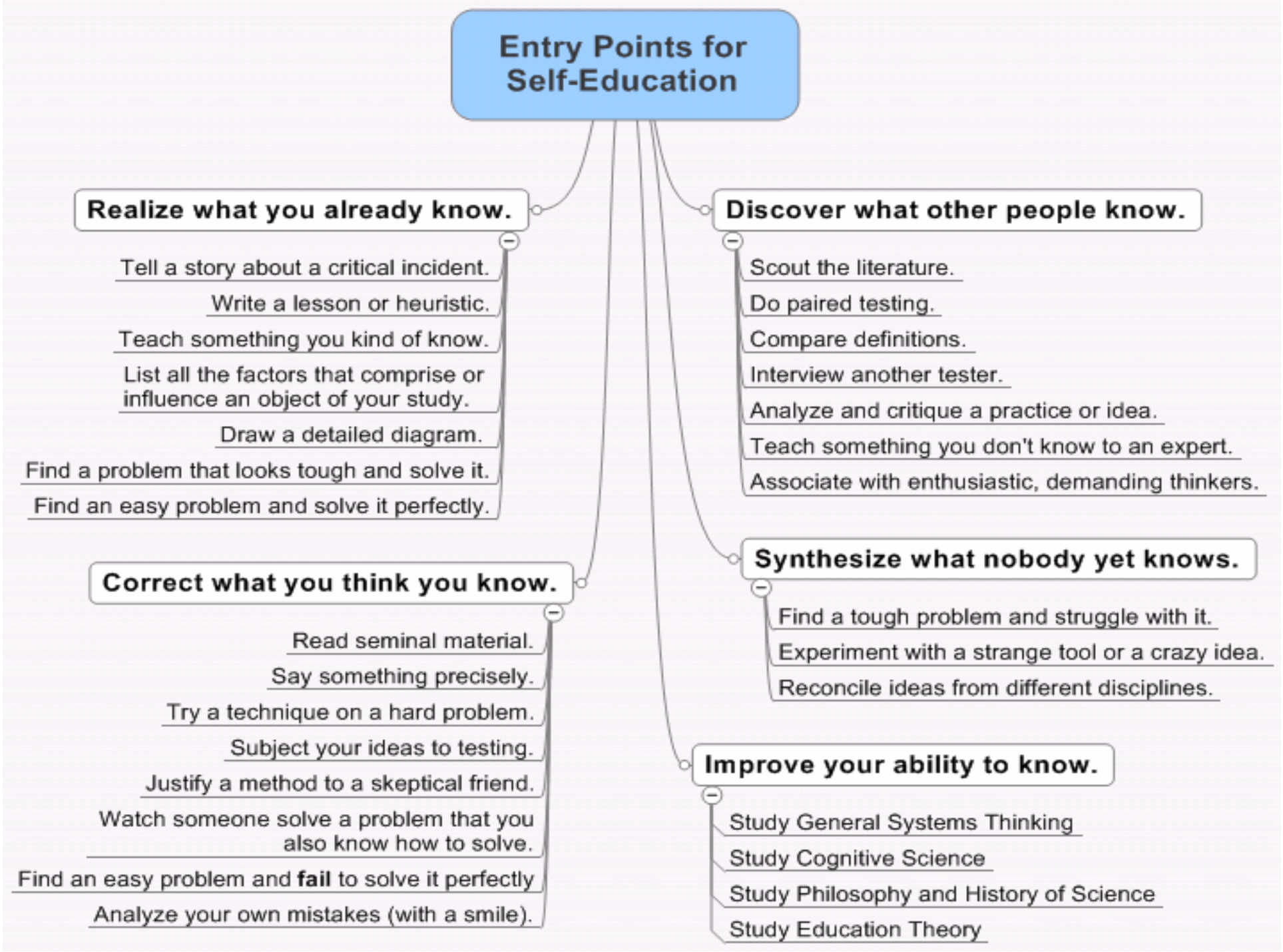


What Level of Learning?

- **Level 0:** "I overcame *obliviousness.*"
  - I now realize there is something here to learn.
- **Level 1:** "I overcame *intimidation.*"
  - I feel I *can* learn this subject or skill. I know enough about it so that I am not intimidated by people who know more than me.
- **Level 2:** "I overcame *incoherence.*"
  - I no longer feel that I'm pretending or hand-waving. I feel reasonably competent to discuss or practice. What I say sounds like what I think I know.
- **Level 3:** "I overcame *competence.*"
  - Now I feel productively self-critical, rather than complacently good enough. I want to take risks, invent, teach, and push myself. I want to be with other enthusiastic students.

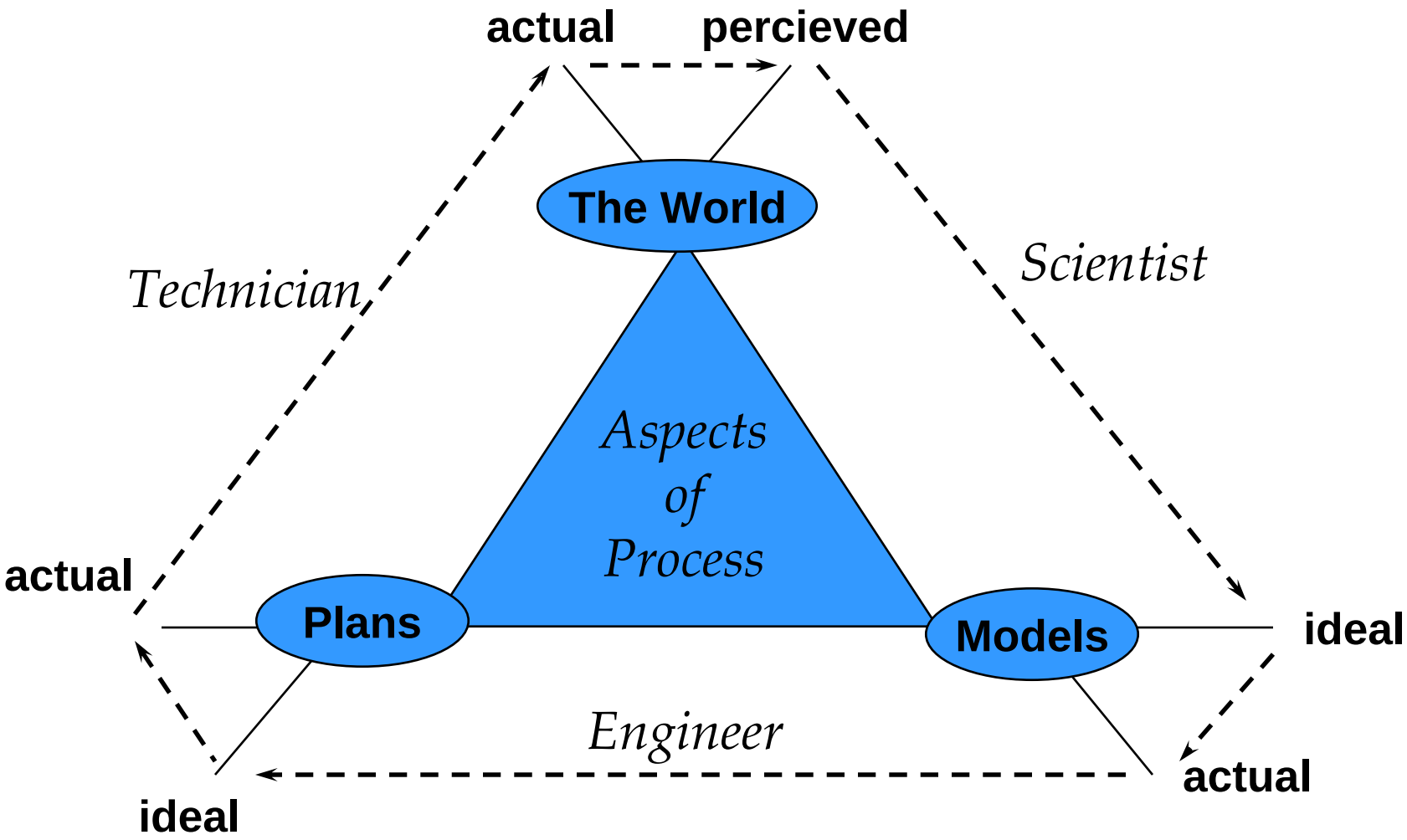
Choosing How to Learn

- **Touring:** I read a survey piece.
- **Experiencing:** I build an example; or do the activity.
- **Serendipity:** I learn from unexpected events.
- **Teacher:** I go see someone.
- **Reading:** I find famous books and papers.
- **Global Supermind:** I tour Google.
- **Standards:** I discover what is considered “correct.”
- **Communities:** I find a forum or professional association.
- **Conferences/Classes:** I attend with a critical attitude.
- **Browsing:** I skim and riffle.
- **Acquisition:** I gather a library.
- **Testing:** I contrast alternatives, critique, or consider extremes.
- **Teaching:** I try to explain it.


Entry Points for Self-Education
Realize what you already know.
Tell a story about a critical incident.
Write a lesson or heuristic.
Teach something you kind of know.
List all the factors that comprise or influence an object of your study.
Draw a detailed diagram.
Find a problem that looks tough and solve it.
Find an easy problem and solve it perfectly.
Discover what other people know.
Scout the literature.
Do paired testing.
Compare definitions.
Interview another tester.
Analyze and critique a practice or idea.
Teach something you don't know to an expert.
Associate with enthusiastic, demanding thinkers.
Correct what you think you know.
Read seminal material.
Say something precisely.
Try a technique on a hard problem.
Subject your ideas to testing.
Justify a method to a skeptical friend.
Watch someone solve a problem that you also know how to solve.
Find an easy problem and fail to solve it perfectly
Analyze your own mistakes (with a smile).
Synthesize what nobody yet knows.
Find a tough problem and struggle with it.
Experiment with a strange tool or a crazy idea.
Reconcile ideas from different disciplines.
Improve your ability to know.
Study General Systems Thinking
Study Cognitive Science
Study Philosophy and History of Science
Study Education Theory

Writing down a lesson or heuristic

Happening to Learning to Doing to Happening



- *Is the underlying problem just difficult to understand?* Some systems that we try to automate are inherently complex or involve difficult technical issues. The programmers will find them complex and difficult too, and that will lead them to mistakes of omission, misunderstanding, and oversimplification.

The more you learn about the product, the technology, and testing in general, the more powerful a compass your confusion becomes, showing you where important problems lie.

In testing, if you know nothing else about a product, you at least know that you're confused. In that situation, confusion can become your best deliverable, in the form of issues and questions that perhaps no one else has the courage to raise.

Fresh eyes find failure.

Making sense of something is a rich intellectual process of assimilating new information into what you already know, while modifying what you know to accommodate the new information. After you've made sense of a product or feature, you have a mental map of it, and your mind doesn't work so hard. This can be a problem for testers. When you know a product well, you make more assumptions about it, and you check those assumptions less often.

This situation has at least three implications for your testing:

- When you come to a product or feature for the first time, pay special attention to what confuses and annoys you. That may tell you something about how a user would react, too.
- When you work with new additions to the team, test alongside them. Watch how they react to the product as they're learning it.
- Beware of getting into a testing rut. Even if you're not following rigid test scripts, you may get so familiar with a particular feature that you test it in progressively narrower ways. Introduce variation wherever you can or switch testing duties with another tester.

Avoid following procedures unless they followed you first.

Beware of other people's procedures. It's common for test cases and procedures to be expressed in a way that says nothing about the underlying design goals of the test. That creates the strong likelihood that you will follow the tests without quite understanding how to set them up or what to look for. In other words, you won't really follow them. In general, test procedures are poorly written and poorly designed, because few good testers are good at what amounts to programming humans like computers. If you're going to follow test procedures, prefer to follow the ones that you designed, you own, or that you thoroughly comprehend.

For best results, *you* should be in control of your testing, not your documentation. Make it follow you.

If you are convinced that procedures are a good thing, at least study how they work. See *Things that Make Us Smart: Defending Human Attributes in the Age of the Machine* (Norman 1993) and *The Social Life of Information* (Brown and Duguid 2000).

When you do create test procedures, avoid "1287."

One of us, Bach, once witnessed a tester write a test procedure that included the line "Type 1287 characters into the field." Where did 1287 come from? The tester explained that her test idea was simply to enter a very large number of characters into the little input field. Because she had heard that test procedures should be specific, she went back and carefully counted the number of characters she had entered, 1287, and that's what she put in the procedure—an arbitrary number, now enshrined forever like cat tracks in a cement sidewalk.

Over-specification is not helpful. When you write down a test procedure, avoid any specificity that is not germane to the concept of the test. Include any information and specificity necessary to frame and explain the test, but let the future tester exercise creativity and judgment. Let the future tester introduce variation that will keep your test procedure fresh and productive.

One important outcome of a test process is a better, smarter tester.

We often hear arguments against any form of testing that results in minimal or no documentation, as though the only value of testing is what comes from writing down our tests. This ignores a profoundly important product of testing: the tester herself.

Good testers are always learning. As the project progresses, they gain insight into the product and gradually improve their reflexes and sensibilities in every way that matters in that project. An experienced tester who knows the

Examples

A selection from Chapter 2: *Thinking Like a Tester*

#19 Testing is in your head.
#20 Testing requires inference, not just comparison of output to expected results.
#24 All "tests" are an attempt to answer some question.
#25 All testing is based on models.
#27 To test, you must explore.
#33 Use implicit as well as explicit specifications.
#37 Use heuristics to quickly generate ideas for tests.
#39 You're harder to fool if you know you're a fool.
#42 Confusion is a test tool.
#43 Fresh eyes find failure.
#44 Avoid following procedures unless they followed you first.
#45 When you do create test procedures, avoid "1287."

How to Write a Heuristic

1. Notice yourself solving a problem or making a decision without an infallible procedure.
2. Identify a non-obvious thought or process that seems to help you solve it.
3. Find an evocative label or phrase for that thought or process.
4. **Try using the label or phrase next time you have the problem, or when you are talking about the problem with someone else.**
5. **Does it help? If not, drop it. If so, see if it sticks.**
6. If you find yourself resisting your own heuristic, either there is something wrong with the heuristic, or the problem isn't sufficiently interesting to you.

Ground Rules for *Lessons Learned in Software Testing*

- Lessons begin with a declarative or imperative sentence.
- Each lesson stands alone.
- Each lesson is peer reviewed.
- Each lesson has a champion who has final say on edits.
- Any author may decline to endorse any lesson.
- Only include lessons that:
  - *are non-obvious or deserve special emphasis.*
  - *at least one of the authors strongly believes in.*
  - *we wished we had learned earlier in our careers.*
  - *can be outlined usefully in a few paragraphs.*
  - *are grounded in the direct experience of at least one author.*
- We do not have to be complete; just helpful.

Try it now

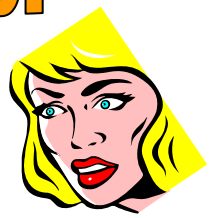
Analyze and critique a practice or idea

Questions for General Analysis

- Why am I here, doing this, right now?
- What things are here, now? What is not here?
- What is happening right now? What is not happening?
- What here do I sense directly, and what am I inferring?
- How might I be wrong about what is really here and happening?
- Where is the system here? What things connect and influence each other?
- How is this like other things? How is it different?
- What is changing and what is constant? In what ways could this change?
- What is important about this? To whom is it important?
- What could be changed or removed without affecting anything that matters?
- How could this be better? Better to whom?
- How is my presence influencing this? What options do I have, right now?
- Project this forward/backward in time. How does it look?
- What would this look like in slow or fast motion? Or zoomed in or out?
- What would be an easy way to learn from this, or about this?
- Is there someone I can talk to about it?

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?

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.

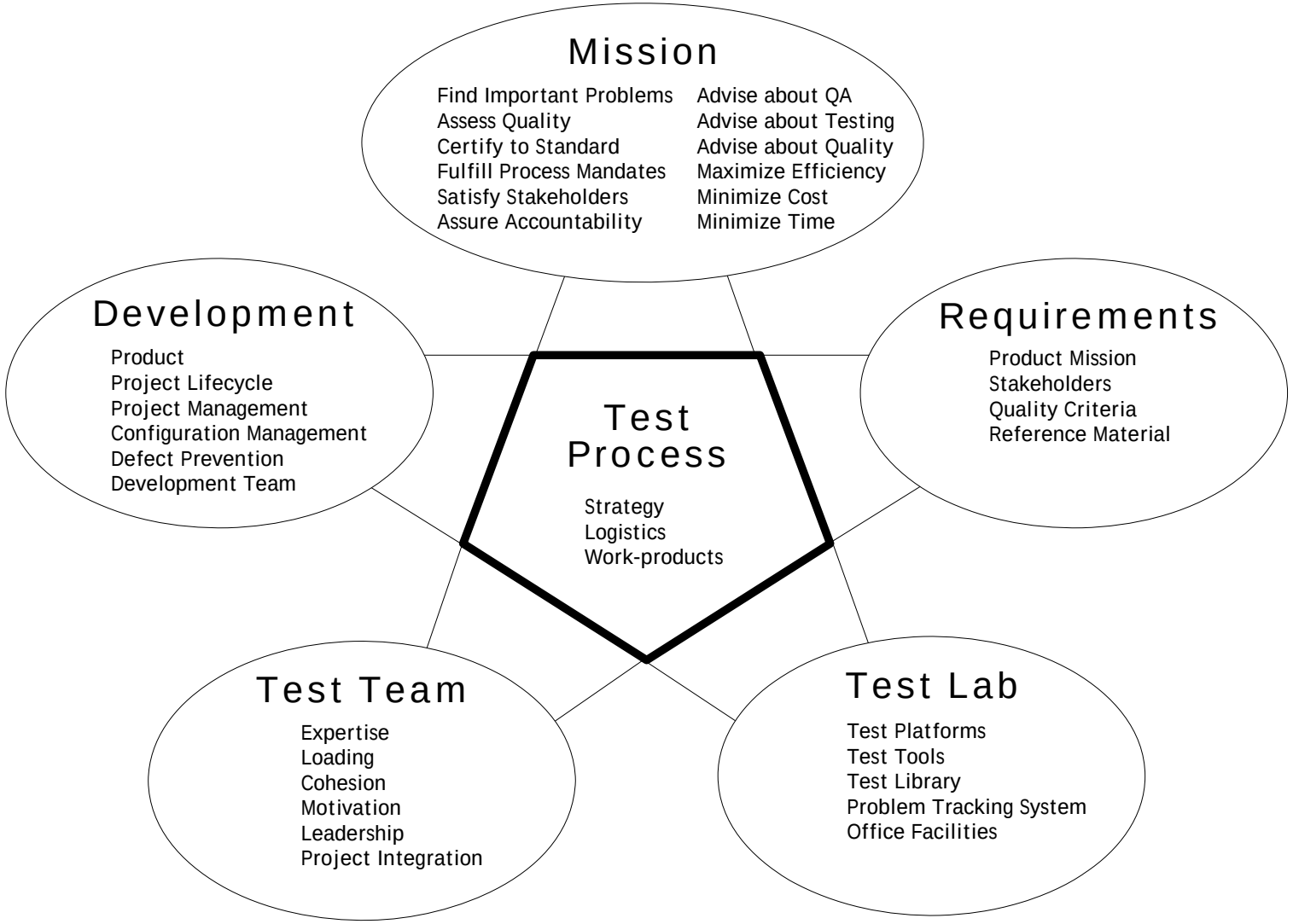
Beware of...

- **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."

Test Project Dynamics: *Context Model*



Try it now

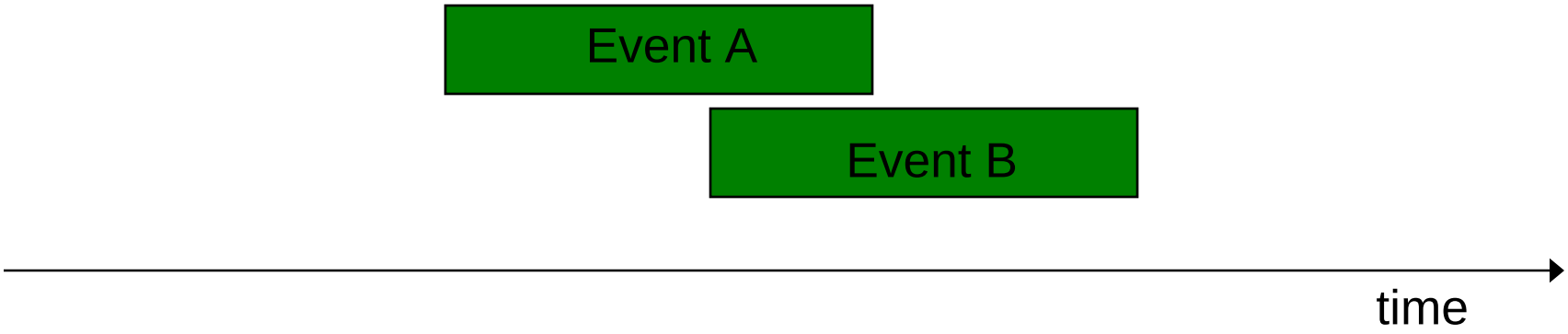
Find an easy problem and solve it perfectly

Try it now

You are using a calculator.
You press the keys “2+2=”

What is the expected result?

Try it now



You want to test the interaction between two potentially overlapping events.

What are the test cases?