The Nature of Exploratory Testing

Cem Kaner, J.D., Ph.D.

Keynote at the
Conference of the
Association for Software Testing
September 28, 2006

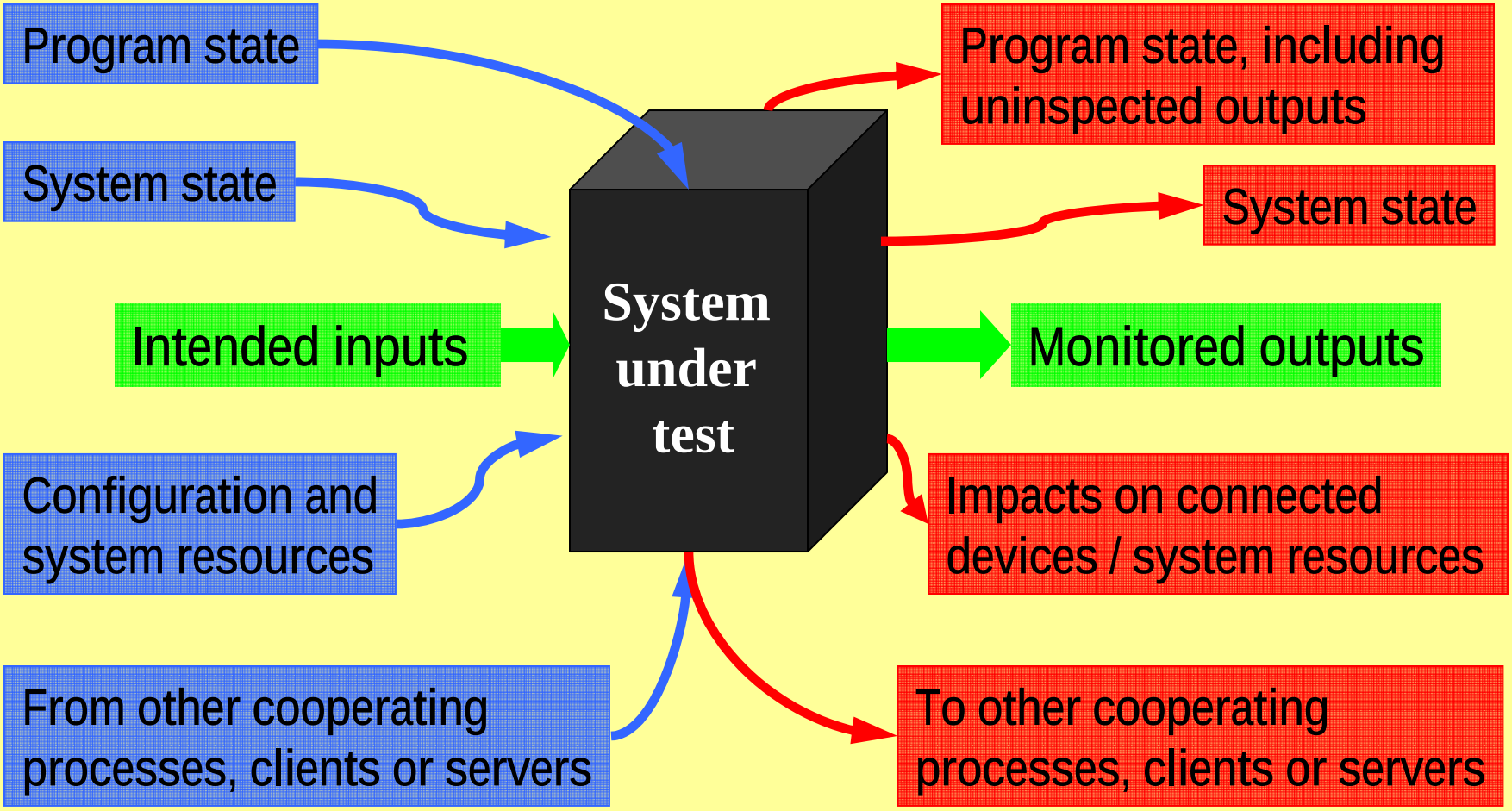
Copyright (c) Cem Kaner 2006. This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/2.0/ or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

These notes are partially based on research that was supported by NSF Grant EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers." Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Opening Demonstration

A program can fail in many ways

Based on notes from Doug Hoffman



 Copyright © 2003-05

What does this tell us about scripted testing?

- *People are finite capacity information processors*
  - *We pay attention to some things*
    - *and therefore we do NOT pay attention to others*
    - *Even events that "should be" obvious will be missed if we are attending to other things.*
- *Computers focus only on what they are programmed to look at (inattentionally blind by design)*
- *A script specifies*
  - *the test operations*
  - *the expected results*
  - *the comparisons the human or machine should make*
  - ***and thus, the bugs the tester should miss.***

Scripted Testing

- *Time sequence*
  - *Design the test early*
  - *Execute it many times later*
  - *Look for the same things each time*
    - *The earlier you design the tests, the less you understand the program and its risk profile*
    - *And thus, the less well you understand what to look at*
    - *The scripted approach means that the test stays the same, even if the risk profile changes.*

Scripted Testing

- *Cognitive sequence*
  - *The smart test designer*
    - *who rarely runs the tests*
  - *designs the tests for the cheap tester*
    - *who does what the designer says and looks for what the designer says to look for*
    - *time and time again*
    - *independently of the risk profile.*
- *Who is in a better position to spot changes in risk or to notice new variables to look at?*

Manufacturing QC

- *Fixed design*
- *Well understood risks*
- *The same set of errors appear on a statistically understood basis*
- *Test for the same things on each instance of the product*
- *Scripting makes a lot of sense*

Design QC

- *The design is rich and not yet trusted*
- *A fault affects every copy of the product*
- *The challenge is to find new design errors, not to look over and over and over again for the same design error*

- *Scripting is probably an industry worst practice for design QC*

- *Software testing is assessment of a design, not of the quality of manufacture of the copy*

What we need for design...

- *Is a constantly evolving set of tests*
- *That exercise the software in new ways (new combinations of features and data)*
- *So that we get broader coverage*
- *Of the infinite space of possibilities*

For <u>that</u>

we do

exploratory testing

Software testing

- *is an empirical*
- *technical*
- *investigation*
- *conducted to provide stakeholders*
- *with information*
- *about the quality*
- *of the product or service under test*

Quality

- *is value*
- *to some person*

  —Gerald Weinberg

- *Note the inherent subjectivity*
- *Note that different stakeholders will perceive the same product as having different levels of quality*

- *Testers look for different things*
  - *for different stakeholders. . . .*

Exploratory software testing

- *is a style of software testing*
- *that emphasizes the personal freedom and responsibility*
- *of the individual tester*
- *to continually optimize the value of her work*
- *by treating test-related learning, test design and test execution*
- *as mutually supportive activities that run in parallel*
- *throughout the project.*

What's a test technique?
Ten dominating techniques

- Function testing
- Specification-based testing
- Domain testing
- Risk-based testing
- Scenario testing
- Regression testing
- Stress testing
- User testing
- State-model based testing
- High volume automated testing

These are 10 common Examples.

There are ***many*** Others.

 Copyright © 2003

http://www.testingeducation.org/BBST/BBST--IntroductiontoTestDesign.html

Test attributes

To different degrees, good tests have these attributes:

- **Power**. When a problem exists, the test will reveal it.
- **Valid**. When the test reveals a problem, it is a genuine problem.
- **Value**. It reveals things your clients want to know about the product or project.
- **Credible**. Your client will believe that people will do the things that are done in this test.
- **Representative** of events most likely to be encountered by the user. (xref. Musa's *Software Reliability Engineering)*.
- **Non-redundant.** This test represents a larger group that address the same risk.
- **Motivating**. Your client will want to fix the problem exposed by this test.
- **Performable**. It can be performed as designed.
- **Maintainable**. Easy to revise in the face of product changes.
- **Repeatable**. It is easy and inexpensive to reuse the test.
- **Pop**. (*short for Karl Popper*) It reveal things about our basic or critical assumptions.
- **Coverage**. It exercises the product in a way that isn't already taken care of by other tests.
- **Easy to evaluate**.
- **Supports troubleshooting.** Provides useful information for the debugging programmer.
- **Appropriately complex.** As the program gets more stable, you can hit it with more complex tests and more closely simulate use by experienced users.
- **Accountable**. You can explain, justify, and prove you ran it.
- **Cost**. This includes time and effort, as well as direct costs.
- **Opportunity Cost**. Developing and performing this test prevents you from doing other work

 Copyright © 2003

http://www.testingeducation.org/BBST/BBST--IntroductiontoTestDesign.html

Contexts Vary Across Projects

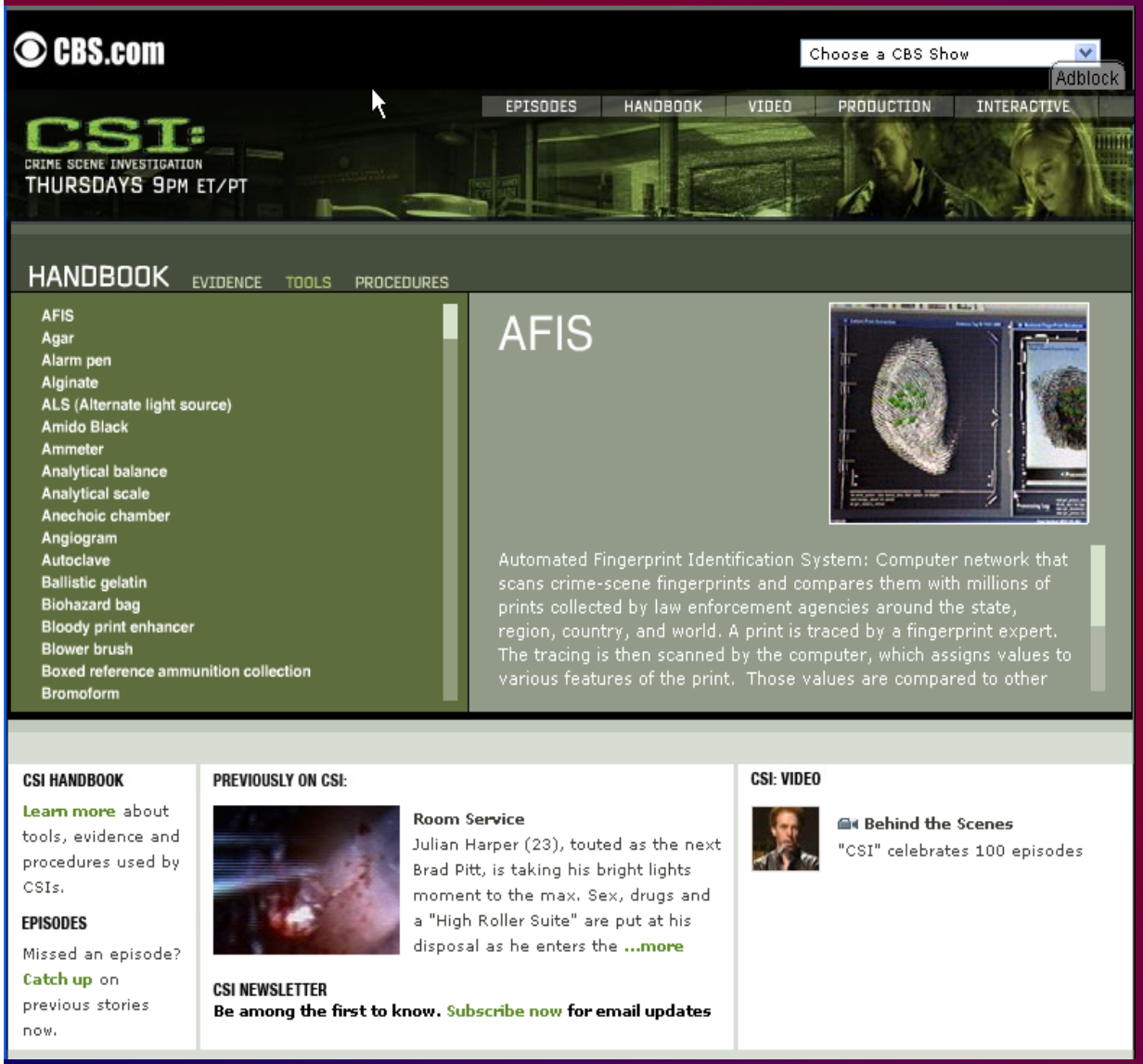
Testers must learn, for each new product:

- *What are the goals and quality criteria for the project*
- *What skills and resources are available to the project*
- *What is in the product*
- *How it could fail*
- *What the consequences of potential failures could be*
- *Who might care about which consequence of what failure*
- *How to trigger a fault that generates the failure we're seeking*
- *How to recognize failure*
- *How to decide what result variables to pay attention to*
- *How to decide what* other *result variables to pay attention to in the event of intermittent failure*
- *How to troubleshoot and simplify a failure, so as to better*

  (a) motivate a stakeholder who might advocate for a fix

  (b) enable a fixer to identify and stomp the bug more quickly
- *How to expose, and who to expose to, undelivered benefits, unsatisfied implications, traps, and missed opportunities.*

It's kind of like CSI



MANY tools, procedures, sources of evidence.

- Tools and procedures don't define an investigation or its goals.
- There is too much evidence to test, tools are often expensive, so investigators must exercise judgment.
- The investigator must pick what to study, and how, in order to reveal the most needed information.

Imagine ...

- *Imagine crime scene investigators*
  - *(real investigators of real crime scenes)*
  - *following a script.*
- *How effective do you think they would be?*

Exploratory Testing After 23 Years

Areas of agreement	*Areas of controversy*
Areas of progress	*Areas of ongoing concern*

Areas of Agreement*

- *Definitions*
- *Everyone does ET to some degree*
- *ET is an approach, not a technique*
- *ET is the response (the antithesis) to scripting*
  - *But a piece of work can be a blend, to some degree exploratory and to some degree scripted*

* *Agreement among the people who agree with me (many of whom are sources of my ideas). This is a subset of the population of ET-thinkers who I respect, and a smaller subset of the pool of testers who feel qualified to write about ET. (YMMV)*

Areas of Controversy

- *ET is not quicktesting*
  - *A quicktest (or an "attack") is a test technique that starts from a theory of error (how the program could be broken) and generates tests that are optimized for errors of that type.*
    - *Example: Boundary analysis (domain testing) is optimized for misclassification errors (IF A<5 miscoded as IF A<=5)*
  - *Quicktests (most) don't require much knowledge of the application under test. They are "ready" right away.*
  - *Quicktesting is more like scripted testing or more like ET depending on the mindset of the tester.*

Areas of Controversy

- *ET is not quicktesting*
- *ET is not only functional testing:*
  - *When programmers define testing, they often define it as functional testing*
    - *Agile ™ system testing is fashionably focused around stories written by customers, not a good vehicle for parafunctional attributes*
    - *Parafunctional work is dismissed as peripheral (e.g. Marick's assertion that it should be done by specialists who are not part of the long term team) (e.g. Beizer's "Usability is not testing")*
  - *If quality is value to the stakeholder*
    - *and if value is driven by usability, security, performance, aesthetics, trainability (etc.)*
    - *then testers should investigate these aspects of the product.*

Areas of Controversy

- *ET is not quicktesting*
- *ET is not only functional testing*
- *ET can involve tools of any kind and can be as computer-assisted as anything else we would call "automated"*
  - *Along with*
    - *traditional "test automation" tools,*
  - *we see emerging tool support for ET such as*
    - *Test Explorer*
    - *BBTest Assistant*
  - *and better thought support tools*
    - *Like mind manager and inspiration*
    - *And qualitative analysis tools like Atlas.ti*

Phone System: The Telenova Stack Failure

Telenova Station Set 1. Integrated voice and data.
108 voice features, 110 data features. 1984.

 Copyright © 2003-05

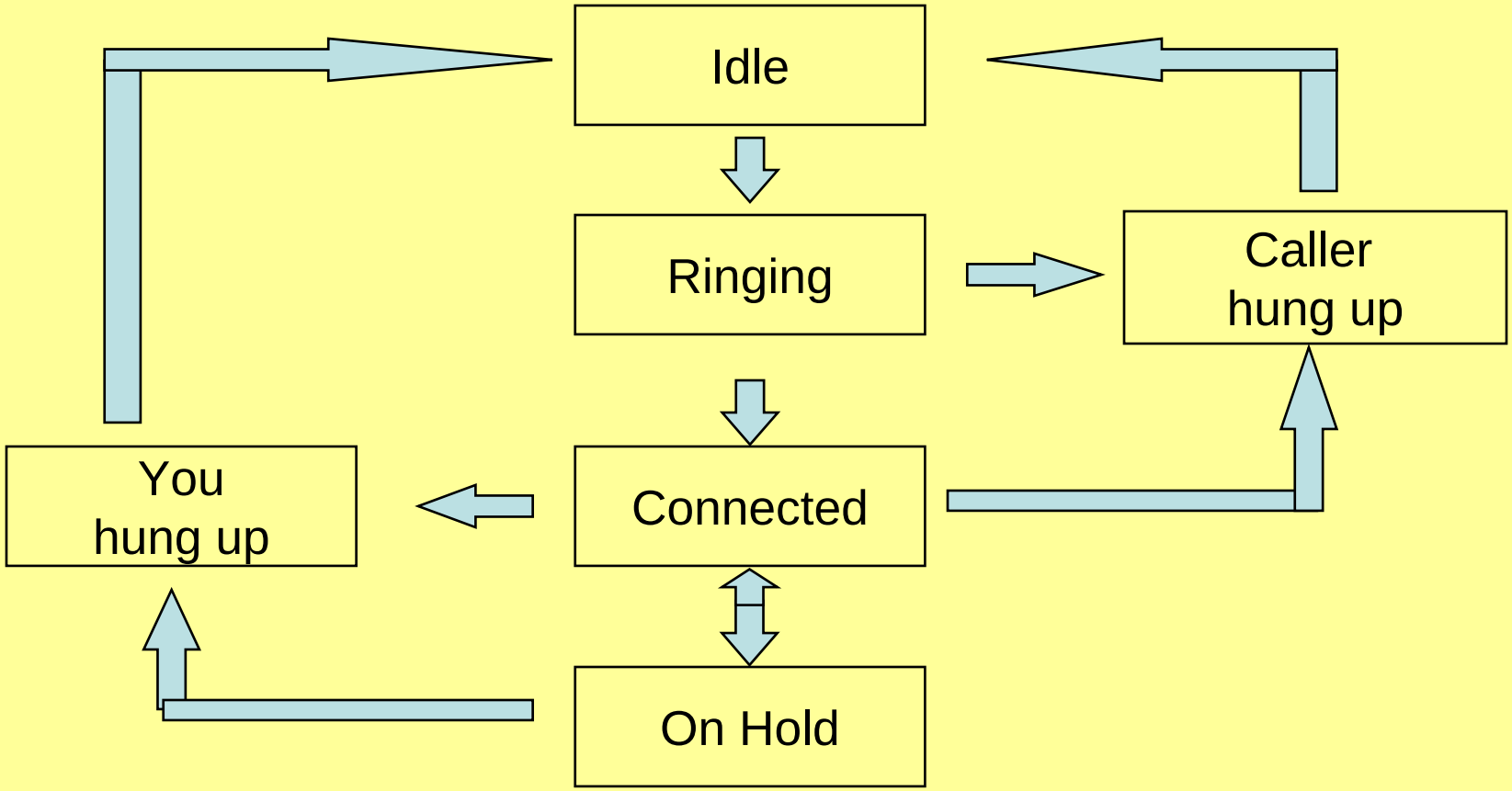
The Telenova Stack Failure

Context-sensitive display
10-deep hold queue
10-deep wait queue

 Copyright © 2003-05

The Telenova Stack Failure

A simplified state diagram showing the bug



 Copyright © 2003-05

The Telenova Stack Failure

The bug that triggered the simulation:

Beta customer (a stock broker) reported random failures

Could be frequent at peak times

An individual phone would crash and reboot, with other phones crashing while the first was rebooting

On a particularly busy day, service was disrupted all (East Coast) afternoon

We were mystified:

All individual functions worked

We had tested all lines and branches.

Ultimately, we found the bug in the hold queue

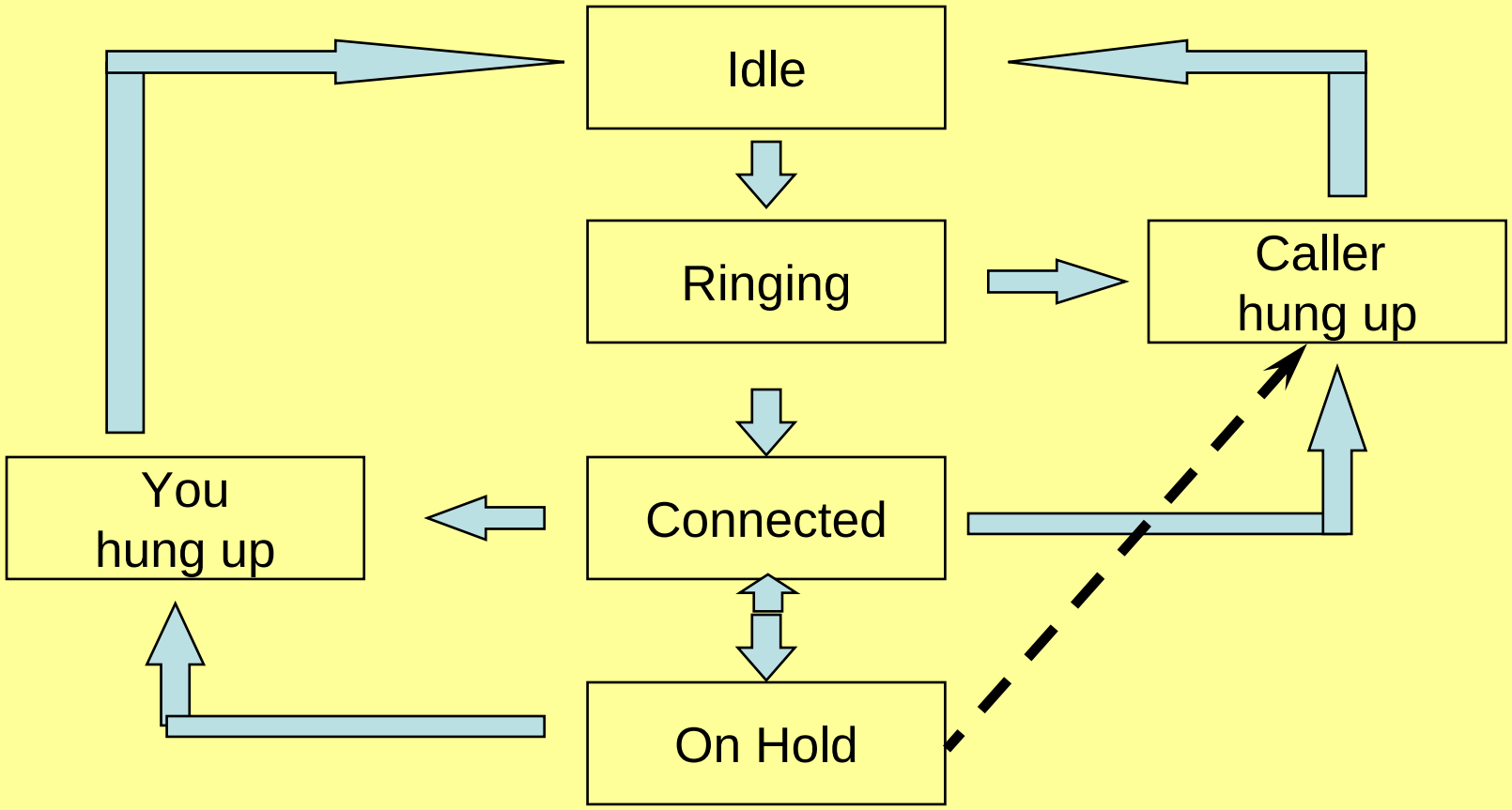
Up to 10 calls on hold, each adds record to the stack

Initially, the system checked stack whenever call was added or removed, but this took too much system time. So we dropped the checks and added these

- Stack has room for 20 calls (just in case)
- Stack reset (forced to zero) when we knew it *should* be empty

The error handling made it almost impossible for us to detect the problem in the lab. Because we couldn't put more than 10 calls on the stack (unless we knew the magic error), we couldn't get to 21 calls to cause the stack overflow.

Telenova Stack Failure



 Copyright © 2003-05

Telenova Stack Failure

Having found and fixed
the hold-stack bug,
should we assume
that we've taken care of the problem
or that if there is one long-sequence bug,
there will be more?

Hmmm…

If you kill a cockroach in your kitchen,
do you assume
you've killed the last bug?
Or do you call the exterminator?

Copyright © 2003-05

Simulator with Probes

- Telenova (*) created a simulator
  - generated long chains of random events, emulating input to the system's 100 phones
  - could be biased, to generate more holds, more forwards, more conferences, etc.

- Programmers added probes (non-crashing asserts that sent alerts to a printed log) selectively
  - can't probe everything b/c of timing impact

- After each run, programmers and testers tried to replicate failures, fix anything that triggered a message. After several runs, the logs ran almost clean.

- At that point, shift focus to next group of features.

- Exposed lots of bugs

This is a classic example of exploratory testing.

(*) By the time this was implemented, I had joined Electronic Arts.

 Copyright © 2003-05

Areas of Controversy

- *ET is not quicktesting*
- *ET is not only functional testing*
- *ET can involve tools of any kind and can be as computer-assisted as anything else we would call "automated"*
- *ET is not focused primarily around test execution*
  - *I helped create this confusion by initially talking about ET as a test technique.*

Controversy: ET as a Technique

- *In the 1980's and early 1990's, I distinguished between*
  - *The evolutionary approach to software testing*
  - *The exploratory testing technique(s), such as:*
    - *Guerilla raids*
    - *Taxonomy-based testing and auditing*
    - *Familiarization testing (e.g. user manual conformance tests)*
    - *Scenario tests*

Controversy: ET as a Technique

- *1999 Los Altos Workshop on Software Testing #7 on Exploratory Testing*
  - *James Tierney presents observations on MS "supertesters" indicating their strength is heavily correlated with social interactions in the development group (they learn from the team and translate the learning into tests)*
  - *Bob Johnson and I present a long list of "styles of exploration" (a categorization of what James Bach & I now call "quicktests," and James Whittaker calls "attacks")*
  - *James Bach shows off his heuristic test strategy model, various other models and heuristics relied on by testers*
  - *Elisabeth Hendrickson, Harry Robinson, and Melora Svoboda also give presentations that discuss the use of models to drive test design in the moment*

Controversy: How can ET be a Technique?

- *We were cataloging dozens of quicktests (essentially techniques) used by explorers. Is ET a family of techniques?*
- *At end of LAWST 7, Gelperin concludes that he doesn't understand what is unique about "exploratory" testing. Our presentations all described approaches to design and execution of tests that he considered normal testing. What was the difference?*
- *He had a point:*
  - *Can you do domain testing in an exploratory way?*
    - *Of course*
  - *Specification-based testing?*
    - *Sure*
  - *Stress testing? Scenario testing? Model-based testing?*
    - *Yes, yes, yes*
  - *Is there any test technique that you cannot do in an exploratory way?*

Controversy: ET is a Way of Testing

- *WHET #1 and #2 – James Bach convinced me that the activities we undertake to learn about the product (in order to test it) are exploratory too.*
  - *Of course they are*
  - *But this becomes the death knell for the idea of ET as a technique*
  - *ET is a way of testing*
    - *We learn about the product in its market and technological space (and keep learning until the end of the project)*
    - *We take advantage of what we learn to design better tests and interpret results more sagely*
    - *We run the tests, shifting our focus as we learn more, and learn from the results.*

Areas of Controversy

- *ET is not quicktesting*
- *ET is not only functional testing*
- *ET can involve tools of any kind and can be as computer-assisted as anything else we would call "automated"*
- *ET is not focused primarily around test execution*
- *ET can involve very complex tests that require significant preparation*
  - *Scenario testing is the classic example*
  - *To the extent that scenarios help us understand the design (and its value), we learn most of what we'll learn in the development and first execution. Why keep them?*

Areas of Controversy

- *ET is not quicktesting*
- *ET is not only functional testing*
- *ET can involve tools of any kind and can be as computer-assisted as anything else we would call "automated"*
- *ET is not focused primarily around test execution*
- *ET can involve very complex tests that require significant preparation*
- *Current testing certifications (and related training) appear to be worthless for exploration support and might be anti-productive*

The certification challenge, as I see it

Software testing

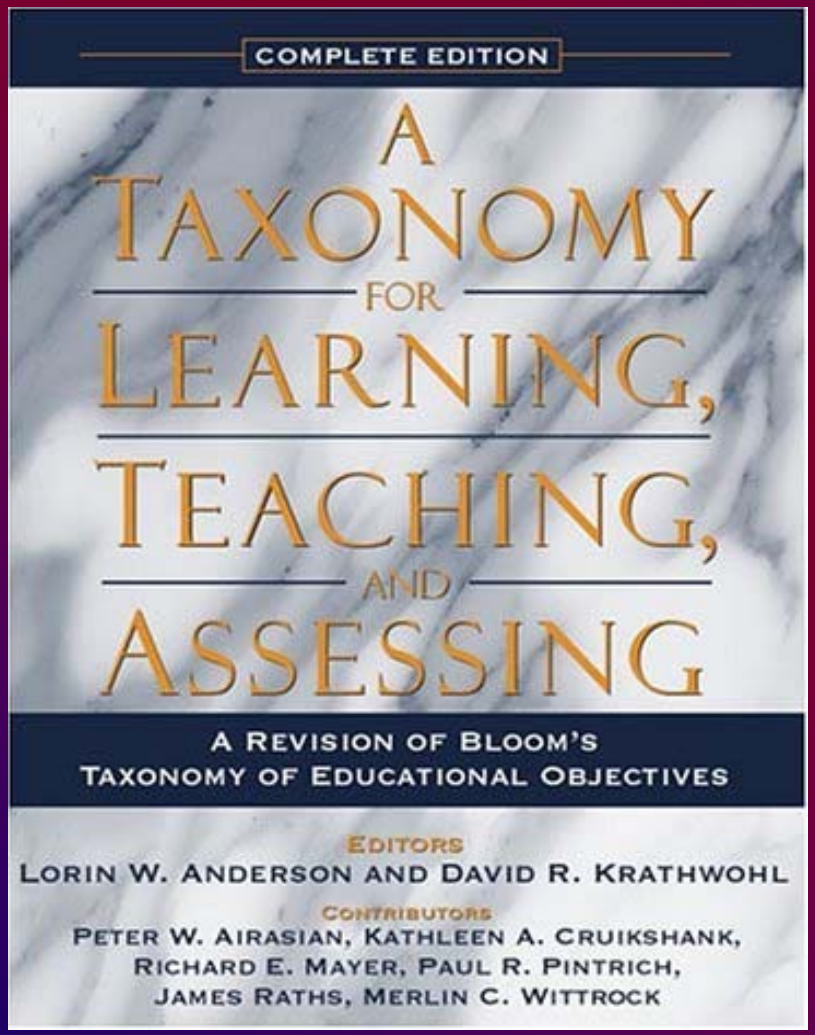
is cognitively complex,

requires critical thinking,

effective communication, and

rapid self-directed learning.

Characterizing Cognitive Complexity



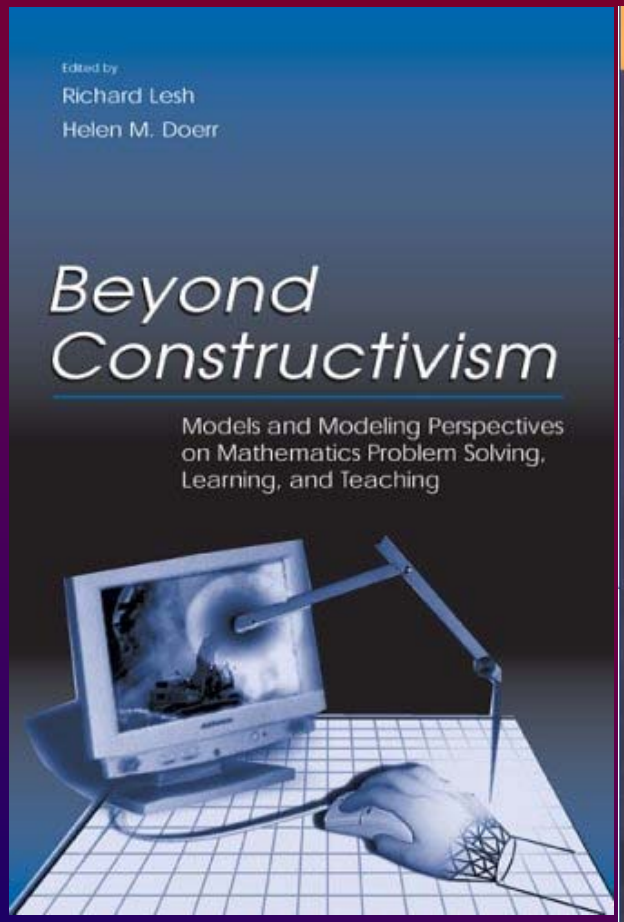
- *Anderson & Krathwohl (2001) provide a modern update to Bloom's (1956) taxonomy*

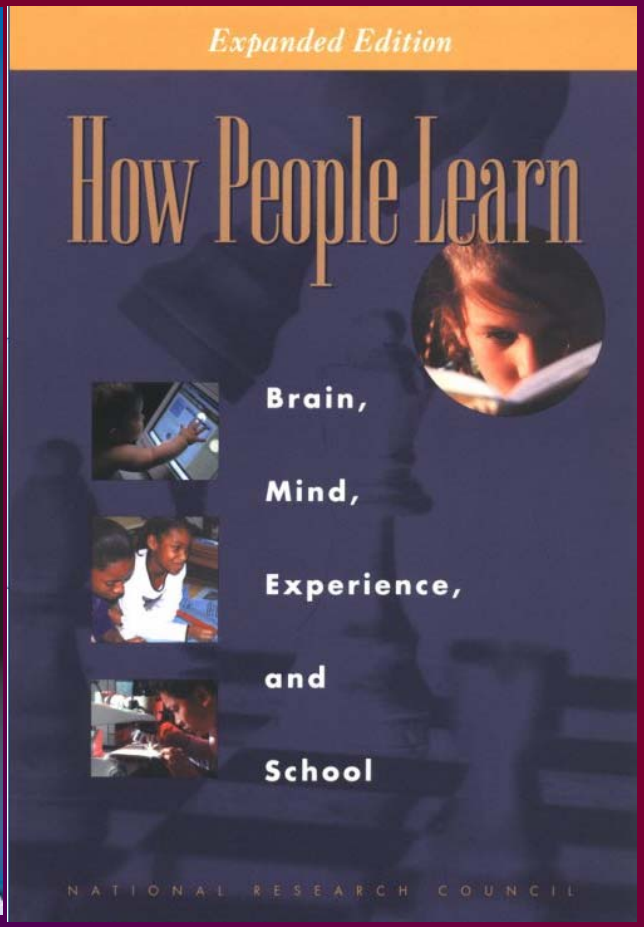
Characterizing Cognitive Complexity

Knowledge Dimension		Cognitive Process Dimension					
		Remember	Understand	Apply	Analyze	Evaluate	Create
	Factual	**lecture**	**lecture**				
	Conceptual	**lecture**	**lecture**				
	Procedural	**lecture**	**lecture**				
	Meta-Cognitive						

Anderson & Krathwohl, 2001

Transfer Problem





- In science / math education, the transfer problem is driving fundamental change in the classroom
- Students learn (and transfer) better when they discover concepts, rather than by being told them or memorizing them

Areas of Progress

- *We know a lot more about quicktests*
  - *Well documented examples from Whittaker's* How to Break... *series and Hendrickson's and Bach's courses*

Areas of Progress

- *We know a lot more about quicktests*
- *We have a better understanding of the oracle problem and oracle heuristics*

Areas of Progress

- *We know a lot more about quicktests*
- *We have a better understanding of the oracle problem and oracle heuristics*
- *We have growing understanding of ET in terms of theories of learning and cognition*
  - *Including benefits of paired testing*

Areas of Progress

- *We know a lot more about quicktests*
- *We have a better understanding of the oracle problem and oracle heuristics*
- *We have growing understanding of ET in terms of theories of learning and cognition*
- *We have several guiding models*
  - *Distinguishing between classification models and generative models*
  - *Satisfice heuristic test strategy model*
  - *Failure mode & effects analysis applied to bug catalogs*
  - *State models*
  - *Other ET-supporting models (see Hendrickson, Bach)*

Areas of Ongoing Concern

- *We are still early in our wrestling with modeling and implicit models*
  - *A model is*
    - *A simplified representation created to make something easier to understand, manipulate or predict some aspects of the modeled object or system.*
    - *Expression of something we don't understand in terms of something we (think we) understand.*

Areas of Ongoing Concern

- *We are still early in our wrestling with modeling and implicit models*
- *Testing is a more skilled and cognitively challenging area of work than popular myths expect*
- *Testing is more fundamentally multidisciplinary than popular myths expect*

Areas of Ongoing Concern

- *We are still early in our wrestling with modeling and implicit models*
- *Testing is a more skilled and cognitively challenging area of work than popular myths expect*
- *Testing is more fundamentally multidisciplinary than popular myths expect*
- *We are just learning how to track and report status*
  - *Session based testing*
  - *Workflow breakdowns*
  - *Dashboards*
    - *Construct validity is still an unknown concept in Computer Science*

Areas of Ongoing Concern

- *We are still early in our wrestling with modeling and implicit models*
- *Testing is a more skilled and cognitively challenging area of work than popular myths expect*
- *Testing is more fundamentally multidisciplinary than popular myths expect*
- *We are just learning how to track and report status*
- *We are just learning how to assess individual tester performance*

Areas of Ongoing Concern

- *We are still early in our wrestling with modeling and implicit models*
- *Testing is a more skilled and cognitively challenging area of work than popular myths expect*
- *Testing is more fundamentally multidisciplinary than popular myths expect*
- *We are just learning how to track and report status*
- *We are just learning how to assess individual tester performance*
- *We don't yet have a good standard tool suite*
  - *Tools guide thinking*
  - *Hendrickson, Bach, others have made lots of suggestions*
  - *Tinkham is working on this for his dissertation*

Closing Notes

- *If you want to attack any approach to testing as unskilled, attack scripted testing*
- *If you want to hammer any testing approach on coverage, look at the fools who think they have tested a spec or requirements document when they have one test case per spec item, or code with one test per statement / branch / basis path.*
- *Testing is a skilled, fundamentally multidisciplinary area of work.*
- *Exploratory testing brings to the fore the need to adapt to the changing project with the information available.*
- *ET is fundamentally agile, but maybe not very Agile ™.*