


Software Testing as a Social Science

*Cem Kaner, J.D., Ph.D.
Professor of Software Engineering
Florida Institute of Technology*

*Canadian Undergraduate Software Engineering Conference
Montreal, PQ
January 19, 2006*

**Copies of these slides: www.testingeducation.org/articles/CUSECstss.pdf
Course materials (video lectures, etc): www.testingeducation.org/BBST**






Greetings

My career focus:

*Satisfaction & Safety of
Software Users and Software Developers*

This doesn't fit in traditional pigeonholes, so I get to see several disciplines from the perspective of an insider and of an outsider.



Social Science?

- *Social sciences study humans, especially humans in society.*
 - *A natural question is, What will the impact of X be on people?*
 - *Work with qualitative and quantitative research methods*
 - *Higher tolerance for ambiguity, partial answers, situationally specific results*
 - *Ethics / values issues are relevant*
 - *Diversity of values / interpretations is normal*
 - *Observer bias is a fact of life, managed explicitly in well-designed research.*

Let's consider some definitions

(not necessarily the ones you find in software engineering textbooks)

- *What's a computer program?*

Let's consider some definitions

- *What's a computer program?*
 - *A communication*
 - *among several humans and computers*
 - *who are distributed over space and time,*
 - *that contains instructions that can be executed by a computer.*

Definitions

- *What is quality?*

Definitions

- *What is quality?*
 - *Quality is value to some person*

– *Jerry Weinberg*

Definitions

- *What is a stakeholder?*

Definitions

- *What is a stakeholder?*
 - *A person who is affected by*
 - *the success or failure of a project*
 - *or the actions or inactions of a product*
 - *or the effects of a service.*
- *Lawyers often limit scope to investment-backed stakeholders (people who have invested in the project).*
- *I prefer to instead follow Don Gause & Jerry Weinberg's approach to scope, and distinguish between*
 - *Favored stakeholders*
 - *Neutral stakeholders*
 - *Disfavored stakeholders*

Definitions

- *What is a software error?*

- *Hint:*

A bug is something that bugs somebody.

James Bach

Definitions

- *What is a software error?*
 - *An attribute of a software product*
 - *that reduces its value to a favored stakeholder*
 - *or increases its value to a disfavored stakeholder*
 - *without a sufficiently large countervailing benefit.*

Definitions

- *What is a requirement?*

Definitions

- *What is a requirement?*
 - *A (potential) attribute of a product (or service) that is desired by a favored stakeholder*
- OR
- *A contract term that obligates a supplier to deliver a product (or service) with a specified attribute.*

Definitions

- *What is a specification?*

Definitions

- *What is a specification?*
 - *A communication*
 - *that describes a requirement*
 - *or a design / implementation decision made in the service of meeting some requirements.*

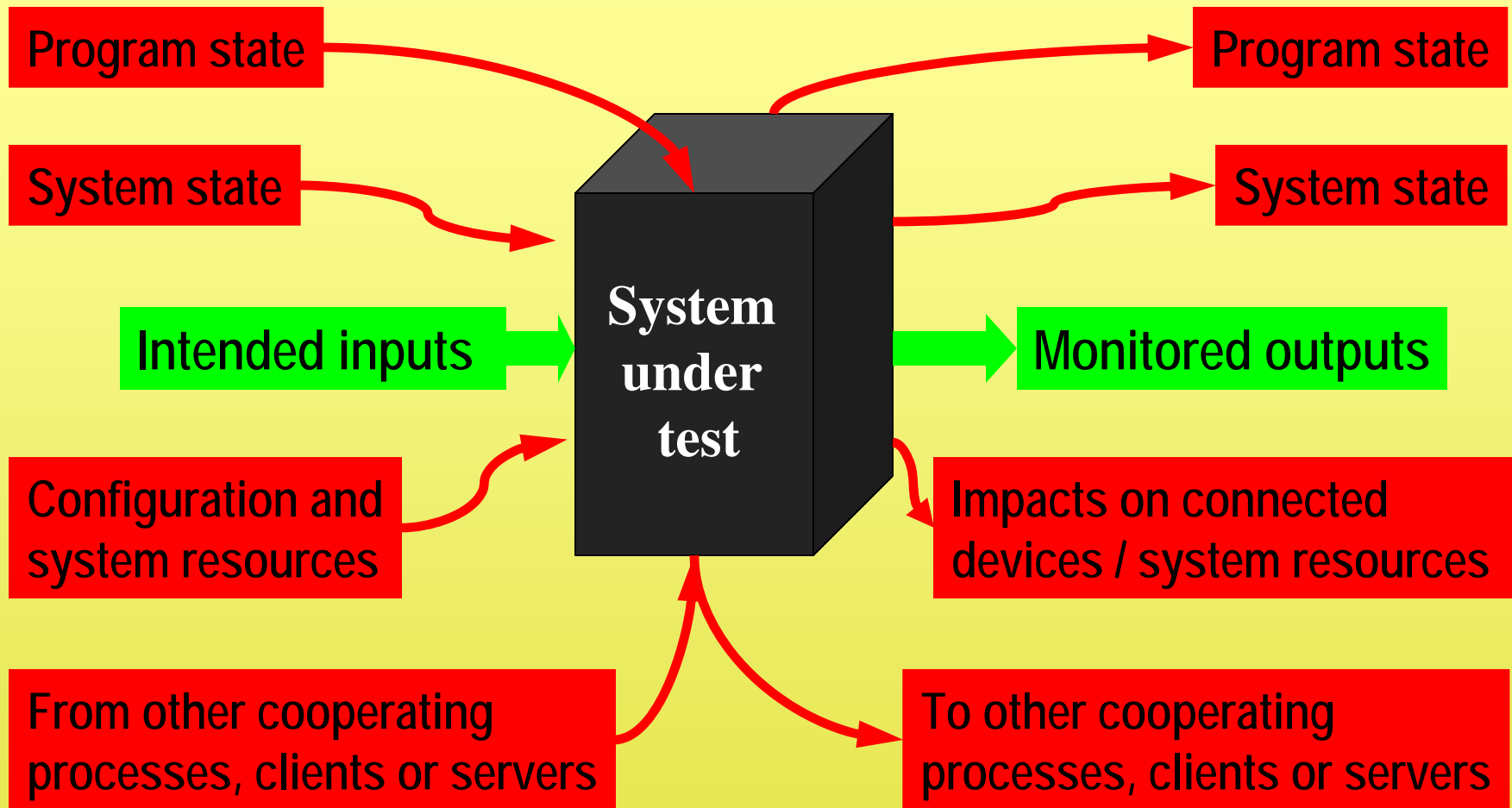
Requirements & Specifications

- *May be incomplete*
 - *The individual attribute may be incompletely considered or specified*
 - *The set of attributes may be (always is) incomplete*
- *May be infeasible*
- *May be incompatible with other requirements or specifications*
- *May vary across stakeholders*
- *May change over time*
- *May not be authoritative*

Definitions

- *What is software testing?*

Recognizing Failure is Challenging



Based on notes from Doug Hoffman

Definitions

- *What is software testing?*
 - *A technical investigation*
 - *conducted to provide quality-related information*
 - *about a software product*
 - *to a stakeholder.*

Information Objectives

- *Find important bugs, to get them fixed*
- *Assess the quality of the product*
- *Help managers make release decisions*
- *Block premature product releases*
- *Help predict and control costs of product support*
- *Check interoperability with other products*
- *Find safe scenarios for use of the product*
- *Assess conformance to specifications*
- *Certify the product meets a particular standard*
- *Ensure the testing process meets accountability standards*
- *Minimize the risk of safety-related lawsuits*
- *Help clients improve product quality & testability*
- *Help clients improve their processes*
- *Evaluate the product for a third party*

Different objectives drive you toward different:

- **Testing techniques**
- **Testing-project management styles**
- **Results reporting methods**
- **Politics on the project**

Test Techniques

- *Essentially, a test technique is a recipe for generating tests*
- *There might be as many as 150 named techniques*
- *Different techniques are useful to different degrees in different contexts*

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.*

Sample Technique: Scenario Testing

The ideal *scenario* has several characteristics:

- The test is **based on a story** about how the program is used, including information about the motivations of the people involved.
- The story is **motivating**. A stakeholder with influence would push to fix a program that failed this test.
- The story is **credible**. It not only could happen in the real world; stakeholders would believe that something like it probably will happen.
- The story involves a **complex use** of the program or a complex environment or a complex set of data.
- The test results are **easy to evaluate**. This is valuable for all tests, but is especially important for scenarios because they are complex.

Note how different this is from the use-case-based scenario.

Use cases abstract out the human issues, such as motivation, and focus on sequence instead.

See John Carroll's work on scenario-based design.

16 Vectors for Creating Scenarios

- Write life histories for objects in the system. How was the object created, what happens to it, how is it used or modified, what does it interact with, when is it destroyed or discarded?
- List possible users, analyze their interests and objectives.
- Consider disfavored users: how do they want to abuse your system?
- List system events. How does the system handle them?
- List special events. What accommodations does the system make for these?
- List benefits and create end-to-end tasks to check them.
- Look at specific transactions that people try to complete, such as opening a bank account or sending a message. List all the steps, data items, outputs, displays, etc.?
- What forms do the users work with? Work with them (read, write, modify, etc.)
- Interview users about famous challenges and failures of the old system.
- Work alongside users to see how they work and what they do.
- Read about what systems like this are supposed to do. Play with competing systems.
- Study complaints about the predecessor to this system or its competitors.
- Create a mock business. Treat it as real and process its data.
- Try converting real-life data from a competing or predecessor application.
- Look at the output that competing applications can create. How would you create these reports / objects / whatever in your application?
- Look for sequences: People (or the system) typically do task X in an order. What are the most common orders (sequences) of subtasks in achieving X?

Let's Sum Up

- *Is testing ONLY concerned with the human issues associated with product development and product use?*
 - *Of course not*
 - *But thinking in terms of the human issues leads us into interesting questions about what tests we are running (and why), what risks we are anticipating, how/why they are important, and what we can do to help our clients get the information they need to manage the project, use the product, or interface with other professionals.*