Acceptance

Thanks go to William Arms for allowing incorporation of his materials

Unit Objective

- Understand what acceptance testing is

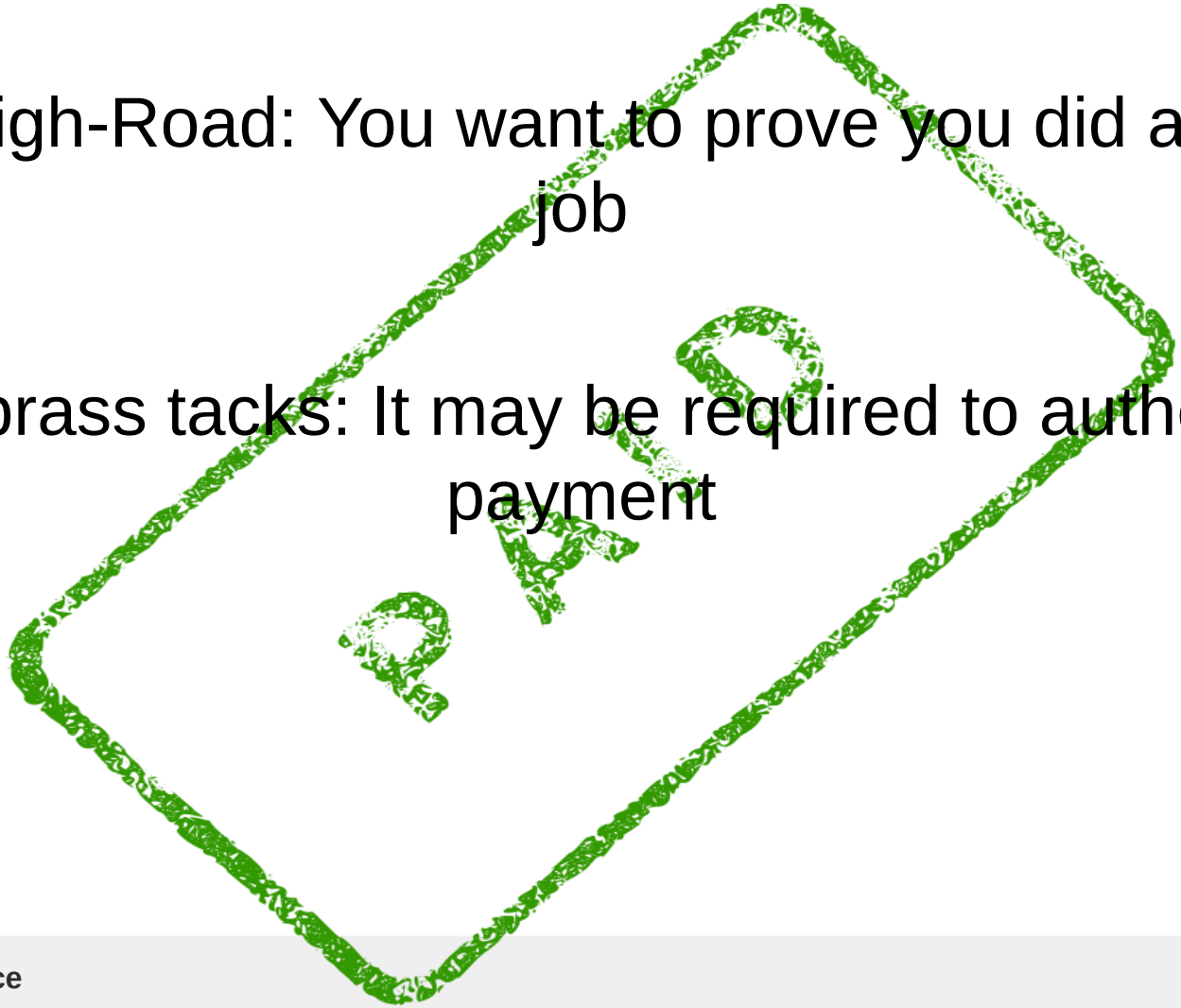
Acceptance Testing

The customers of the system assess whether the delivered work meets the objectives and can be deployed

Why is This Important?

The High-Road: You want to prove you did a good job

The brass tacks: It may be required to authorize payment



Two Types of Testing

User Acceptance Testing

- Assesses whether the system meeting the user's expectations and needs
- Does it do what it needs to do so the customers can use it?

Two Types of Testing

Operational Readiness Testing

- Assesses whether the system is stable enough and performs well enough to meet the operational expectations and needs
- Can the system be supported and can it fit into the production environment?

UAT by Alpha / Beta Testing

Testing pre-release versions of code with groups of users

Alpha — Small groups “testing” very early releases.
No promise of code stability.

Beta — Larger groups testing/evaluating co
Ideally a release candidate (soft launch)

One Way To Conduct Alpha or Beta Testing

THROW THE SYSTEM OUT THERE AND SEE WHAT HAPPENS


Sorry, wrong answer.

The Point is to LEARN SOMETHING About the System

First challenge

- Finding users who will address the questions to which you want answers

Who will test the system?

- What qualifies a candidate user?
- How to attract/recruit users? (or, why would they want to use your test code?)
- Will they use everything or just a subset?
- Will they actually use it at all?

The Point is to LEARN SOMETHING About the System

First challenge

- Finding users who will address the questions to which you want answers

Second challenge

- Getting these users into proper position
  - What training or support will these users need?

The Point is to LEARN SOMETHING About the System

Third challenge

- Collecting the data

- How to collect the results?
  - How to capture and deliver the data from the user
- How to collect the *right* data in a timely manner
  - Users will not be systematic
  - Users may not be prompt

The Point is to LEARN SOMETHING About the System

Third challenge

- Collecting the data

Fourth challenge

- Actually doing something with the data

Rigorous View of Acceptance

The Client tests the COMPLETE SYSTEM

Testing is against the requirements and is often scenario-driven.

1. System
2. Documentation
   - Operational
   - Training
3. Procedures
   - Installation
   - Fault Identification and Mitigation
   - Backup and Recovery

Emphasis is on

- Failure recovery
- Error handling
- Restarts
- Stability

Acceptance Testing

Is *Black Box* testing

Involves the entire system

- No piece-parts

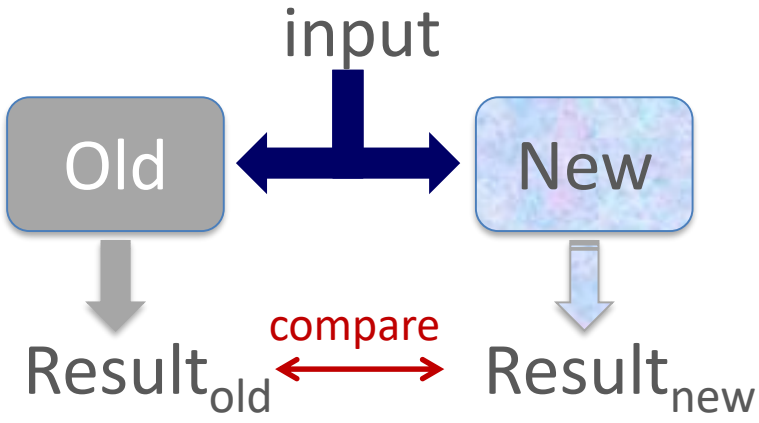
The client assesses/decides whether the SUT (System Under Test) meets the requirements

Acceptance Testing

Everything should be REAL

1. *Real Data*
2. *Realistic Situations*
3. *Real Users*
4. *Real Administrators*
5. *Real Operators*

Acceptance Test When Replacing a System - Parallel Testing



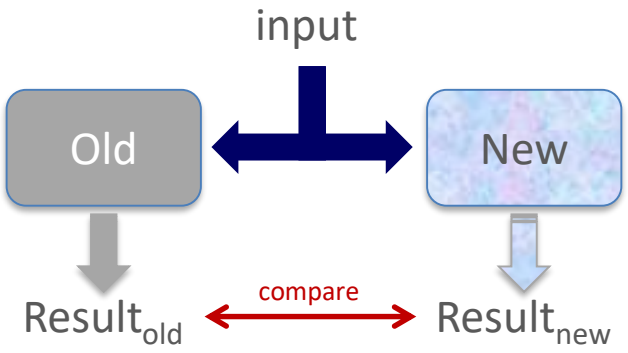
If the results are the same (or improved),

Then Accept!

At end, simply turn on one side (and remember which results were inhibited)

Impacts of Using Parallel Testing

- Requires twice the infrastructure
- Need a mechanism for comparing results
- Often run for a very long time
- Some results shouldn't be doubled
  - E.g. sending a bill



At end, simply turn on one side (and remember which results were inhibited)

The Bottom Line

Does the system that you built satisfy the client

Does it meet the requirements?

Is this the system envisioned?

Chaos Report 1994-20

this?!

19

-
- i
  c
  a
  w
  tir

- Wh
  yea
  nur
  sim

h_fil

Source: Survey
– 350

Could Agile Help?

Each sprint is a complete dev cycle

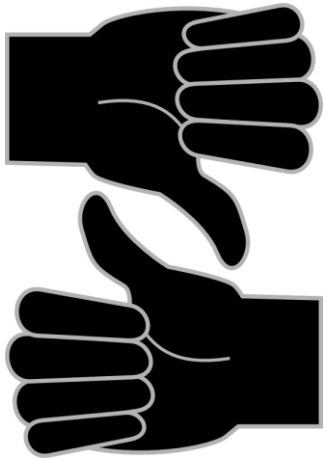
Put user acceptance of sprint's results as the last step of the sprint

Does not eliminate UA risks,
but reduces the risk to the *scope of one sprint*

If there is a significant amount of new module interactions, acceptance testing may require running all (or most all) of the previous sprints' acceptance tests

Reality Check

Acceptance is positioned as a binary, clear-cut decision



Reality Check

Acceptance is positioned as a binary, clear-cut decision

Rarely is anything complex 100% functional and operation

So acceptance is often a satisficing decision, with conditional acceptance substituting for final acceptance

Why Would Anyone Accept Partial Work?

Client View

- Can reap (some) value now
  - The iterative vs. waterfall notion
- Often has made capital investments and needs to show use
- Process changes may already be in place
- Bottom line: the costs of waiting for a better system out-weigh the system's shortcomings

Supplier View

- Allows the supplier to collect
  - Enables some revenue
  - Offsets accrued debt
- Willing to commit to (fast) fixes perhaps at reduced or no cost
  - Has a vested interest in seeing something deployed to protect own name in the market

Four Conditions Affecting How Contentious Acceptance Is - Regardless Of Process

1. What expectations for the overall system were set at the beginning?
   - How close is the final product to those expectations?
   - How much rework happened?
     - How much was client driven versus supplier caused?

2. How involved was the client?

3. What promises did the client make to management –
   and how close is the final product to those commitments?

4. How much depends on this one decision?
   - Financially: *lump sum* payments versus *partial* payments
   - Functionally: *only* release versus *first* release