Maxims for Malfeasant Software Testing

Johnny Klonaris
Senior Software QA Engineer
There’s more to software testing...

Much like an iceberg, where the portion below water is unseen, by far larger and the source of most of the danger, I contend that most of software testing is not about testing software...
There’s more to software testing...

Much like an iceberg, where the portion below water is unseen, by far larger and the source of most of the danger, I contend that most of software testing is not about testing software...
• There’s more to software testing than testing software
• You’re never done: Iteration as a way of life
• Perspective: The Big Picture vs. Devilish Details

“testing” can include software install, hardware infrastructure, interfacing with build and engineering groups, and lots more...
About your presenter...

Johnny Klonaris
Senior Software Quality Engineer

- ~30 years experience
- 17+ years software QA
- 7+ years at MontaVista
- “Senior” by design
“Maxims for Malfeasant Software Testing”

-or-

“How to Design a Software Testing Lab to Fail”

Based on the idea:
“Maxims for Malfeasant Designers, or how to design languages to make programming as difficult as possible”
by Richard L. Wexelblat, 1976

http://portal.acm.org/citation.cfm?id=807695&coll=portal&dl=ACM
Maxim: Focus on Testing

- Much more than testing:
  - Software Build
  - Product Install
  - Hardware Infrastructure
    - Power
    - Networking
    - Air Conditioning
  - Target Bring-up
  - Software Infrastructure
Maxim: Focus on Testing

It’s much more than the software:
- Installation and Tools
- Customer Experience
- Supportability
- Interfacing with other groups
- Documentation
Maxim: Minimize Moves / Disruptions

“Three Removes are as bad as a Fire”
-Benjamin Franklin

- Physical moves can be an opportunity
  - Identify and toss junk / store historically interesting hardware
  - Reorganize
- Space set aside for cable management is not a waste

Story about cabling.
The full quote is:
“I never saw an oft removed Tree, Nor yet an oft removed Family, That throve so well, as those that settled be. And again, Three Removes are as bad as a Fire.”
[1758 B. Franklin Poor Richard's Almanack (Preface)]
Ball of cables pulled out in once piece from behind the cabinets.

The way things are now.
Maxim: RS-232 is a standard

- So what?
- Methods for solving serial connection issues
- RS-232 - DE-9 to RJ-45

RJ-45 is becoming a new standard for serial connections. It’s more reliable and through the use of patch bays, allows for rerouting / reconnecting. Problem: nearly every serial port wired directly to an RJ-45 connector is different from the standard. The solution is to build the connection into the hood.
Dealing with RS-232

We need to know how to wire these hoods. Manually built tables worked for a while - until they became too complicated to manage by hand. What was needed was a script to turn a simple data file into a readable table.
We need to know how to wire these hoods. Manually built tables worked for a while - until they became too complicated to manage by hand. What was needed was a script to turn a simple data file into a readable table.
Maxim: Use off the shelf tools

- Sometimes: yes
- Available tools may not be enough
- Problem: Symbolically represent the problem / solution
- Best solution may be the simplest
Sure you can...
Maxim: You can’t test documentation

- Sure you can...

2. At the `CFE>` prompt, run the following command to configure the target to use BOOTP:

   `setenv -p LINUX_CMDLINE "ip=bootp"

   Note: You only need to run this command on the target once.

3. Set the IP address on the target. Enter the following command:

   `ifconfig eth0 -auto`

4. Boot the kernel. Enter the following command:

   `boot -elf <host IP address>:vmlinux-broadcom-bcm91250-mips2_fp_be`

Obviously one can “test” documentation by reviewing it. But it’s also possible to test by making sure you structure your automation to use the same procedures as are called out in the manual. This shows that our documentation, our scripting and ultimately what comes out the boot screen match.
Sure you can...

2. At the CFE prompt, run the following command to configure the target to use BOOTP:
   ```
   setenv -p LINUX_CMDLINE "ip=bootp"
   
   Note: You only need to run this command on the target once.
   ```

3. Set the IP address on the target. Enter the following command:
   ```
   ifconfig eth0 -auto
   ```

4. Boot the kernel. Enter the following command:
   ```
   boot -elf <host IP address>:vmlinux-bcm91250-mips2_fp_be
   ```

   send_user "Waiting for CFE>
   expect {
     "CFE>"
   }
   sleep 1
   send "setenv -p LINUX_CMDLINE "ip=bootp"
   expect {
     "CFE>"
   }
   sleep 1
   send "ifconfig eth0 -auto"
   expect {
     "CFE>"
   }
   sleep 1
   send "boot -elf -tftp $HostIP:/tftpboot/$Alias"
   # End of converted script...

Obviously one can “test” documentation by reviewing it. But it’s also possible to test by making sure you structure your automation to use the same procedures as are called out in the manual. This shows that our documentation, our scripting and ultimately what comes out the boot screen match.
Sure you can...

send_user "Waiting for CFE>\n"  
expect {  
  "CFE>"  
}  
sleep 1  
send "setenv -p LINUX_CMDLINE \"ip=bootp\""  
expect {  
  "CFE>"  
}  
sleep 1  
send "ifconfig eth0 -auto\r"  
expect {  
  "CFE>"  
}  
sleep 1  
send "boot -elf tftp $HostIP:/tftpboot/$A
# End of converted script..."

Obviously one can “test” documentation by reviewing it. But it’s also possible to test by making sure you structure your automation to use the same procedures as are called out in the manual. This shows that our documentation, our scripting and ultimately what comes out the boot screen match.
Reality can force some unusual solutions. We needed a “robot” to push the button on our hardware. This little guy does the trick.
Maxim: Software engineers can’t do hardware

- Maybe...
- Necessity, etc...

Reality can force some unusual solutions. We needed a “robot” to push the button on our hardware. This little guy does the trick.
Reality can force some unusual solutions. We needed a “robot” to push the button on our hardware. This little guy does the trick.
Maxim: Get it right the first time

- No way to know
- Idiot proof solutions inspire the universe to be more bizarre
- Solution:
  - Flexibility in design / layout / coding
  - Willingness **not** to leave well enough

Example of the universe becoming more bizarre: An early dhcp automation process built the dhcpd.conf file from a target database. Vastly improved booting reliability. Then along came serial boot. This required some small changes, but then along came: the SH family, which booted by first mounting the rootfs and then copying the kernel from the NFS mount. Who knew?
Maxim: Never change your testing software during a release cycle

- Sometimes you just have to
- A good time to be thorough and careful
  - Compatibility
  - Backups
  - Backout strategy

What fool would swap in a completely rewritten results reporting subsystem two weeks before a release was due to ship?
Maxim: Information is a good thing

- Too much of a good thing...
- Management doesn’t want details
- Management doesn’t want overviews
- Management wants:

  “SHIP IT!”

Overheard a manager being shown our automation results tables “That’s great, when did you do that?”. We’d had it for over a year and had included the link in dozens of emails. Point being: you can’t assume folks will read – esp. if it’s not clear it’s what they need.
Then again, you also can’t expect anyone to use an "overview" that looks like this. This represents our feature test results from several years ago on the platforms for that release.
Maxim: Management doesn’t want details

- Sure they do, if they make sense
- But mostly they want an overview
- Engineers need overviews too

A more modern / current overview. The bottom line shows a complete install failure – probably a build issue. The top line shows that all 24 tests built – but at least one indicated a warning.
Maxim: Management doesn’t want details

A more modern / current overview. The bottom line shows a complete install failure – probably a build issue. The top line shows that all 24 tests built – but at least one indicated a warning.
Maxim: Management doesn’t want details

A more modern / current overview. The bottom line shows a complete install failure – probably a build issue. The top line shows that all 24 tests built – but at least one indicated a warning.
This shows some detail on that warning (the yellow square). We also see that some kernel build failures on xtensa.
• Structure is needed
• Flexibility can be planned for...

Talk about pre-allocation of some of the patch panel jacks for cross patching. A close up of one of the “remote” panels. The blue tape designates networking connections.
Maxim: Structure precludes flexibility

- Structure is needed
- Flexibility can be planned for...

Talk about pre-allocating some of the patch panel jacks for cross patching. A close up of one of the “remote” panels. The blue tape designates networking connections.
Maxim: Neatness counts

- Actually, it does!
- Neatness:
  - Provides traceability
  - Implies you know what you’re doing

A collection of serial port connections. Note: all cables are labeled – this was not “prepped”.
Maxim: Neatness counts

- Actually, it does!
- Neatness:
  - Provides traceability
  - Implies you know what you’re doing

A collection of serial port connections. Note: all cables are labeled – this was not “prepped”.
Maxim: Neatness counts

Actually, it does!

Neatness:
Provides traceability
Implies you know what you're doing

A collection of serial port connections. Note: all cables are labeled – this was not “prepped”.
Reliability is the product of the accumulated failures.

The devil can be in the details: what are these?

Talk about the complexity of making a serial connection from the web interface, to starting a Java telnet applet, to a telnet connection to a host, then to a serial server... Many things can go wrong - there's no sense taking a chance that a bumped connector could be the source when 5 cents and 20 seconds can secure the connection.
Maxim: Focus on the details

- Mapping the structure of the automation to the tasks at hand is critical
- A broad view of testing means that all parts of the product are fair game

Just as the previous slides talked about the importance of details – this “anti-maxim” implies we need to also keep the big picture in mind. We need to map the structure to the problem, while still maintaining enough flexibility. Assuring Quality can include a very wide range of topics.
Everything you know is wrong?

- No, but some assumptions are worth challenging
- Software QA can be a lot more than just testing software
- A QA engineer’s job is never done
- Nothing succeeds like success
Extra Slides
Extra “slides” to go into some detail on how we can look into the logs for failures, if there’s interest.
Extra “slides” to go into some detail on how we can look into the logs for failures, if there’s interest.
Extra “slides” to go into some detail on how we can look into the logs for failures, if there’s interest.
Extra “slides” to go into some detail on how we can look into the logs for failures, if there’s interest.
Extra “slides” to go into some detail on how we can look into the logs for failures, if there’s interest.