Growing From a Reckless Bughunter to a Stakeholder Conversationalist

EuroSTAR | Software Testing Conference | #esconf5

rikard.edgren@learningwell.se
Earning Respect

find valuable information
testing is never better than
the communication of the results
Story: My Biggest Mistake

• 1st opportunity to test customer-specific add-on
• 30 bugs!
understand your testing mission
The Seven Basic Principles of the Context-Driven School

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.
James Bach’s Implicit Principles

- **Context Primacy:** Context is not inert scenery, it embodies vital information, resources, constraints, and other agents that must inform all competent work.
- **Scientific Aspiration:** Folklore is not a basis for a respectable craft. Our work is informed by evidence, cleaned and tempered by skepticism and vigorous debate. Community status is accumulated through demonstrated and demonstrable merit. We avoid groundless and exaggerated claims.
- **Systems Non-Linearity:** Our systems are not practically predictable or reducible in terms of linear or statistical equations. We must use non-linear, cybernetic control methods, and learn to live with uncertainty.
- **Testing as Investigation:** Testing is not just fact checking and it is not quality improvement. It is an open-ended investigation and learning process focused on discovering problems.
- **Humanist Sensibility:** Technical workers are not interchangeable resources. All technical work is done by unique, unreliable people, and to be good at technical work we must develop as people.
- **Tester Autonomy:** We are not robots or slaves: we have agency. We manage the value of our time and bear responsibility for doing ethical work. We must cultivate the courage to do that.
- **Tester Responsibility:** We are not alone. We work within a social network in which value is constructed and responsibility is shared. This happens on project, corporate, professional, and societal levels.
- **Methodology Authorship:** Ignorantly mimicking behavior is not competent work. Competent testers must design (or adapt) and test their own practices and heuristics.
- **Skill Development:** Technical work is not brute labor. Methodology skill, in both tacit and explicit form, is absolutely required to fulfill our mission, and development of such skill is an ongoing obligation.
The Poster Story

- 2009 - 2010
- Rikard Edgren, Martin Jansson and Henrik Emilsson
- A very long list with software quality characteristics
## Software Quality Characteristics

<table>
<thead>
<tr>
<th>Capability</th>
<th>Can the product perform valuable functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Can you trust the product in many and difficult situations?</td>
</tr>
<tr>
<td>Usability</td>
<td>Is the product easy to use?</td>
</tr>
<tr>
<td>Charisma</td>
<td>Does the product have &quot;it&quot;?</td>
</tr>
<tr>
<td>Security</td>
<td>Does the product protect against unwanted usage?</td>
</tr>
<tr>
<td>Portability</td>
<td>Is transfer of the product to other environments and languages enabled?</td>
</tr>
<tr>
<td>Performance</td>
<td>Is the product fast enough?</td>
</tr>
</tbody>
</table>

### Definitions
- **Capability**: Can the product perform valuable functions?
- **Reliability**: Can you trust the product in many and difficult situations?
- **Usability**: Is the product easy to use?
- **Charisma**: Does the product have "it"?
- **Security**: Does the product protect against unwanted usage?
- **Portability**: Is transfer of the product to other environments and languages enabled?
- **Performance**: Is the product fast enough?
- **Compatibility**: How well does the product interact with software and environments?
- **Supportability**: Can customers' usage and problems be supported?
- **Testability**: Is it easy to check and test the product?
The Poster Story

- 2009 - 2010
- Rikard Edgren, Martin Jansson and Henrik Emilsson
- A very long list with software quality characteristics
Working with Quality Characteristics

• very easy to get something good-looking
• happily accepted, but not anchored
• driven by only a part of context, me...
• Still a good list though, especially for generating test ideas

• Now: start with blank paper; quality in customer’s words
Story: The Conversationalist

- more talking than testing nowadays
- information pull over information push
- get heard by adjusting the language

context-driven
A Few Tips

• most people are very occupied, make them important
• understand the information objectives, by listening
• explore what is important

• ask follow-up questions
• act on answers!
testing is simple: you understand what is important, and you test it
Explaining the testing

• why are we testing?
• why is the test strategy good?
• your stakeholders are decision-makers
the communication of the test results are seldom better than the anchoring of the test strategy
Exercise: 30 seconds

• Team up in pairs.
• Explain the benefits of your test strategy in 30 seconds.
it’s not only the testing,
it’s how you talk about it
Conclusions

• understand your testing mission
• find out what is important
• communicate with good words
Questions

- who should you talk to?
- what will you tell?
- what will you ask?