Growing From a Reckless Bughunter to a Stakeholder Conversationalist

EuroSTAR Software Testing Conference | #esconfs

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!

context-unaware

understand your testing mission

www.context-driven-testing.com

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

IT-bility. *Is the product easy to install, maintain and support?*

Capability. *Can the product perform valuable functions?*

Compatibility. *How well does the product interact with software and environments?*

Reliability. *Can you trust the product in many and difficult situations?*

Supportability. *Can customers' usage and problems be supported?*

Usability. *Is the product easy to use?*

Testability. *Is it easy to check and test the product?*

Charisma. *Does the product have "it"?*

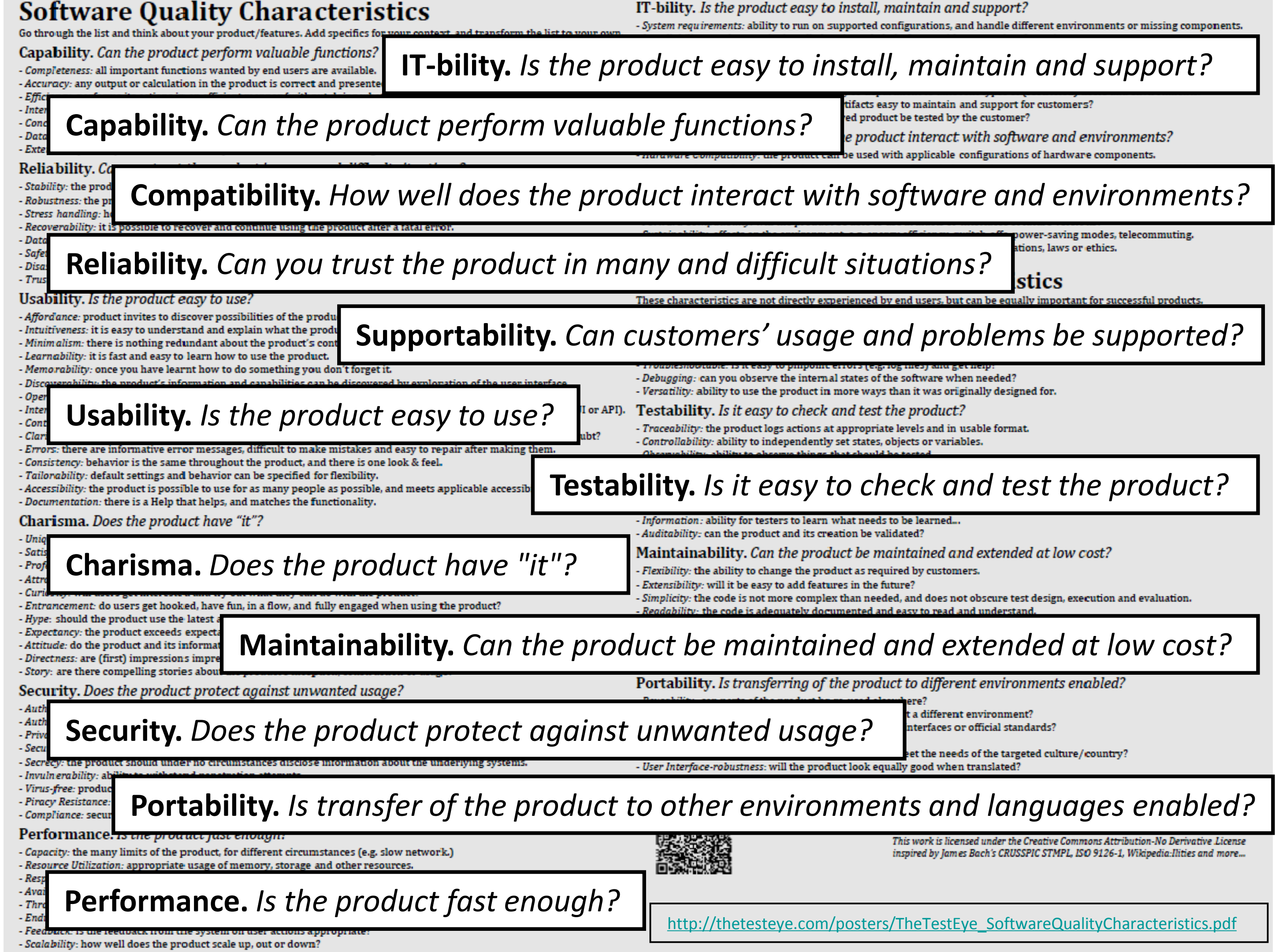
Maintainability. *Can the product be maintained and extended at low cost?*

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?*

http://thetesteye.com/posters/TheTestEye_SoftwareQualityCharacteristics.pdf



Software Quality Characteristics

Go through the list and think about your product/features. Add specifics for your context, and transform the list to your own.

Capability. *Can the product perform valuable functions?*
- *Completeness:* all important functions wanted by end users are available.
- *Accuracy:* any output or calculation in the product is correct and presente
- *Effic*
- *Inter*
- *Conc*
- *Data*
- *Exte*

Reliability. *Ca*
- *Stability:* the prod
- *Robustness:* the pr
- *Stress handling:* h
- *Recoverability:* it is possible to recover and continue using the product after a fatal error.
- *Data*
- *Safe*
- *Disa*
- *Trus*

Usability. *Is the product easy to use?*
- *Affordance:* product invites to discover possibilities of the produ
- *Intuitiveness:* it is easy to understand and explain what the produ
- *Minimalism:* there is nothing redundant about the product's cont
- *Learnability:* it is fast and easy to learn how to use the product.
- *Memorability:* once you have learnt how to do something you don't forget it.
- *Discoverability:* the product's information and capabilities can be discovered by exploration of the user interface.
- *Oper*
- *Inter* I or API).
- *Cont*
- *Clari* ubt?
- *Errors:* there are informative error messages, difficult to make mistakes and easy to repair after making them.
- *Consistency:* behavior is the same throughout the product, and there is one look & feel.
- *Tailorability:* default settings and behavior can be specified for flexibility.
- *Accessibility:* the product is possible to use for as many people as possible, and meets applicable accessib
- *Documentation:* there is a Help that helps, and matches the functionality.

Charisma. *Does the product have "it"?*
- *Uniq*
- *Satis*
- *Prof*
- *Attr*
- *Curi*
- *Entrancement:* do users get hooked, have fun, in a flow, and fully engaged when using the product?
- *Hype:* should the product use the latest a
- *Expectancy:* the product exceeds expecta
- *Attitude:* do the product and its informat
- *Directness:* are (first) impressions impre
- *Story:* are there compelling stories about

Security. *Does the product protect against unwanted usage?*
- *Auth*
- *Auth*
- *Priv*
- *Secu*
- *Secrecy:* the product should under no circumstances disclose information about the underlying systems.
- *Invulnerability:* a
- *Virus-free:* produc
- *Piracy Resistance:*
- *Compliance:* secur

Performance.
- *Capacity:* the many limits of the product, for different circumstances (e.g. slow network.)
- *Resource Utilization:* appropriate usage of memory, storage and other resources.
- *Resp*
- *Avai*
- *Thro*
- *Endu*
- *Feedback:* is the feedback from the system on user actions appropriate?
- *Scalability:* how well does the product scale up, out or down?

IT-bility. *Is the product easy to install, maintain and support?*
- *System requirements:* ability to run on supported configurations, and handle different environments or missing components.

tifacts easy to maintain and support for customers?
ed product be tested by the customer?

e product interact with software and environments?
- *Hardware Compatibility:* the product can be used with applicable configurations of hardware components.

power-saving modes, telecommuting.
ations, laws or ethics.

stics

These characteristics are not directly experienced by end users, but can be equally important for successful products.

- *Debugging:* can you observe the internal states of the software when needed?
- *Versatility:* ability to use the product in more ways than it was originally designed for.

Testability. *Is it easy to check and test the product?*
- *Traceability:* the product logs actions at appropriate levels and in usable format.
- *Controllability:* ability to independently set states, objects or variables.
- *Information:* ability for testers to learn what needs to be learned...
- *Auditability:* can the product and its creation be validated?

Maintainability. *Can the product be maintained and extended at low cost?*
- *Flexibility:* the ability to change the product as required by customers.
- *Extensibility:* will it be easy to add features in the future?
- *Simplicity:* the code is not more complex than needed, and does not obscure test design, execution and evaluation.
- *Readability:* the code is adequately documented and easy to read and understand.

Portability. *Is transferring of the product to different environments enabled?*

ere?
a different environment?
nterfaces or official standards?
et the needs of the targeted culture/country?
- *User Interface-robustness:* will the product look equally good when translated?

The Poster Story

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

context-hipster

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?