77 Test Idea Triggers

Rikard Edgren
TIBCO Spotfire
SAST Öresund 2011-05-04
There are many definitions of quality:
“guess it’s that people like it”
“quality is value to some person”

doesn’t help me so much if these persons can’t tell me more

quality is more than the subjective sum of relevant quality attributes like capability, reliability, usability, charisma, security, performance, IT-bility, compatibility”
e.g. a stable product is better than an unstable

A long definition like this is of course practically useless.
Quality characteristics describe attributes that most software benefit from. They can be used on the whole product or for details.

The whole is made by the details. The quality of a detail is defined by the whole.

This is a thorough extension in the same spirit as Bach’s CRUSSPIC STMPL.


Does anyone have an example application we can use today?
At my company there are a dozen that always are fruitful.

IMPORTANT!!!

what’s most important differs; so remove items, add specifics, and transform the list to your own
### Capability

*Can the product perform valuable functions?*

- **Completeness**: all important functions wanted by end users are available.
- **Correctness**: any output from the product is correct.
- **Efficiency**: performs its actions in an efficient manner (without doing what it’s not supposed to do.)
- **Interoperability**: different features interact with each other in the best way.
- **Concurrency**: ability to perform multiple parallel tasks, and run at the same time as other processes.
- **Data agnosticism**: supports all possible data formats, and handles noise.
- **Extensibility**: ability for customers or 3rd parties to add features or change behavior.

Many test efforts stop with capability.
Reliability
Can you trust the product in many and difficult situations?

Stability
Robustness
Recoverability
Resource Usage
Data Integrity
Safety
Disaster Recovery
Trustworthiness

Reliability. Can you trust the product in many and difficult situations?
- Stability: the product shouldn’t cause crashes, unhandled exceptions or script errors.
- Robustness: the product handles foreseen and unforeseen errors gracefully.
- Recoverability: it is possible to recover and continue using the product after a fatal error.
- Resource Usage: appropriate usage of memory, storage and other resources.
- Data Integrity: all types of data remain intact throughout the product.
- Safety: the product will not be part of damaging people or possessions.
- Disaster Recovery: what if something really, really bad happens?
- Trustworthiness: is the product’s behavior consistent, predictable, and trustworthy?
Ongoing Test Ideas

in the back of your head
executed for free

reveals more information the more you test

things you have in the back of your head all the time
Characteristics (e.g. Stability, Performance) that are executed for free all the time
# Usability

*Is the product easy to use?*

<table>
<thead>
<tr>
<th>Affordance</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intuitiveness</td>
<td>Clarity</td>
</tr>
<tr>
<td>Minimalism</td>
<td>Errors</td>
</tr>
<tr>
<td>Learnability</td>
<td>Consistency</td>
</tr>
<tr>
<td>Memorability</td>
<td>Tailorability</td>
</tr>
<tr>
<td>Discoverability</td>
<td>Accessibility</td>
</tr>
<tr>
<td>Operability</td>
<td>Documentation</td>
</tr>
<tr>
<td>Interactivity</td>
<td></td>
</tr>
</tbody>
</table>

**Usability. Is the product easy to use?**
- **Affordance:** product invites to discover possibilities of the product.
- **Intuitiveness:** it is easy to understand and explain what the product can do.
- **Minimalism:** there is nothing redundant about the product’s content and appearance.
- **Learnability:** it is fast and easy to learn and remember how to use the product.
- **Memorability:** once you have learnt to do something you don’t forget it.
- **Discoverability:** the product’s capabilities can be discovered by systematic exploration of the user interface.
- **Operability:** an experienced user can perform common actions very fast.
- **Interactivity:** the product has easy-to-understand states and possibilities of interacting with the application (via GUI or API).
- **Control:** the user should feel in control over the proceedings of the software.
- **Clarity:** is everything stated explicitly and in detail, with a language that can be understood, leaving no room for doubt?
- **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 accessibility standards.
- **Documentation:** there is a Help that helps, and matches the functionality.
Charisma. Does the product have “it”?

- **Uniqueness**: the product is distinguishable and has something no one else has.
- **Sex appeal**: you just can’t stop looking at or using the product.
- **Satisfaction**: how does it feel after using the product?
- **Professionalism**: does the product have the appropriate flair of professionalism and feel fit for purpose?
- **Attractiveness**: are all types of aspects of the product “good-looking”?
- **Curiosity**: will users get interested and try out what they can do with the product?
- **Entrancement**: do users get hooked, have fun, in a flow, and fully engaged when using the product?
- **Hype**: does the product use too much or too little of the latest and greatest technologies/ideas?
- **Expectancy**: the product exceeds expectations and meets the needs you didn’t know you had.
- **Attitude**: do the product and its information have the right attitude and speak to you with the right language and style?
- **Directness**: are (first) impressions impressive?
- **Story**: are there compelling stories about the product’s inception, construction or usage?

I’m confident you agree that many of these are important aspects for the users’ perception of your product.

Then how come you don’t test charisma??
Unorthodox Test Ideas

awareness of violations
(deep) user interviews
testers’ subjectivity
uncontrolled users/environments

free-form, open-ended

Things like Charisma aren’t easily captured in tests.
You have to take alternative actions.

free-form: E.g. think about Charisma for a feature, and brainstorm around what can be done testing-wise.
Get rid of Pass/Fail-thinking, we can communicate noteworthy information.
Security, Does the product protect against unwanted usage?

- Authentication: the product’s identifications of the users.
- Authorization: the product’s handling of what an authenticated user can see and do.
- Privacy: ability to not disclose data that is protected to unauthorized users.
- Security holes: product should not invite to social engineering vulnerabilities.
- Secrecy: the ability to under no circumstances disclose information about the underlying systems.
- Invulnerability: ability to withstand penetration attempts.
- Virus-free: product will not transport virus, or appear as one.
- Piracy Resistance: no possibility to illegally copy and distribute the software or code.
- Compliance: security standards the product adheres to.
Performance

*Is the product fast enough?*

- **Capacity**: the many limits of the product, also under load, stress or slow network.
- **Responsiveness**: the speed of which an action is (perceived as) performed.
- **Availability**: the system is available for use when it should be.
- **Throughput**: the product’s ability to process many, many things.
- **Feedback**: is the feedback from the system on user actions appropriate?
- **Scalability**: how well does the product scale up, out or down?
What’s Important?

all characteristics are always important...
...but the OK zone is often easy to reach

conflicting characteristics...
...are solved by judgment, for each case

Judgment needs a lot of knowledge, collaboration and feelings.
Critical Usability

The new scalpel is ready for launch!

Great, tell me about it.

It has 22% better Accuracy, 40% better Throughput, and is unbreakable!

Wow, any drawbacks?

Nah. Nothing at all?

Maybe some minor deviations.

OK, just as long as there are no surprises on launch.

Well, it weighs 20 pounds...
**IT-ility**

*Is the product easy to install, maintain and support?*

- **System requirements**: ability to run on supported configurations, and handle different environments or missing components.
- **Installability**: product can be installed on intended platforms with appropriate footprint.
- **Upgrades**: ease of upgrading to a newer version without loss of configuration and settings.
- **Uninstallation**: are all files (except user’s or system files) and other resources should be removed when uninstalling?
- **Configuration**: can the installation be configured in various ways or places to support customer’s usage?
- **Deployability**: product can be rolled-out by IT department to different types of (restricted) users and environments.
- **Maintainability**: are the product and its artifacts easy to maintain and support for customers?
- **Testability**: how effectively can the deployed product be tested by the customer?
Compatibility. How well does the product interact with software and environments?

- **Hardware Compatibility**: the product can be used with applicable configurations of hardware components.

- **Operating System Compatibility**: the product can run on intended operating systems, and follows typical behavior.

- **Application Compatibility**: the product, and its data, works with other applications customers are likely to use.

- **Configuration Compatibility**: product’s ability to blend in with configurations of the environment.

- **Backward Compatibility**: can the product do everything the last version could?

- **Forward Compatibility**: will the product be able to use artifacts or interfaces of future versions?

- **Sustainability**: effects on the environment, e.g. energy efficiency, switch-offs, power-saving modes, support work from home.

- **Standards Conformance**: the product conforms to applicable standards, regulations, laws or ethics.
Internal Quality Characteristics

Supportability
Testability
Maintainability
Portability
Localization

These are also important!

...but we will not go into details today.
Some, but not all, of these characteristics can be quantified and made regressionable as automated tests.
Some are better tested manually and subjectively.
For most, both automated and manual approaches are recommended, because repetitive automated tests and changing manual tests are complementary, not antagonistic. If they feel in conflict, re-check your thinking.

Computers are faster, humans are better judges of importance.

A perfect scenario could be where developers create automated tests for the “checking”, and system testers can use diverse approaches to do the necessary testing (without being bothered by “easy” bugs.)

If you’re doing regression testing, automation is closer. When testing something new, manual is a better fit. Tools can always be used!
It is fast to write twenty guiding statements about the most important characteristics for your system.

You can use the list to create your own, shorter, better definition of quality characteristics.
Closing Notes

Many characteristics are considered during requirement, design, implementation
= Must Test
Some are not considered = Must Test

1 new idea to try?
1 ongoing test idea from now on?

A lot of these have been thought about during requirements, design and implementation; so we must test them.
Those that haven’t been considered, yet still important, we really must test.
Isn’t it better to use the list in the beginning of the project?
- One way doesn’t rule out the other. But I’m a tester, and testers are my main audience.

I counted to 8 categories and 73 sub-characteristics; does that add up to 77?
- I have no comments on that remark ;)

www.thetesteye.com

redgren@tibco.com