Software Testing Tutorial Critique

To: Dr. Dan Jones

From: Allison Young

Date: 10/25/2014

Re: Tutorial Critique

Introduction

For this critique, I considered analyzing several more technical guides before settling on the Software Testing Tutorial. I wanted to pick a guide that contained a balance of content I understood and new material so I could more easily identify which features were effective and which detracted from the user's ability to learn. I work alongside engineers who conduct software testing; therefore, I know much of the terminology, but not the details of how the testing processes are determined. This tutorial is beneficial for me, and I have the background knowledge to judge its coverage of the topic.

The Software Testing Tutorial is overall an effective document, although loose grammar and inconsistent design – in this case symptomatic of a lack of a focused topic – detract from the user's ability to reference this tutorial in getting started with software testing. This memo report will cover successful and unsuccessful elements in the tutorial using the seven design principles outlined by Dr. Jones, as well as instances of rambling content structure. This tutorial will be viewed from the critical perspectives expressed in Janice Redish's book, *Letting Go of the Words*, and critiqued on its ability to hold a conversation with the user.

Content

What stands out most in this tutorial are its prevalent grammatical errors. Several sentences are rendered incoherent by mistakes in verb tense and punctuation. One such instance of this occurs with the phrase, "Although they are interrelated and at some level they can be considered as the same activities, but there is indeed a difference between them." This sentence is difficult to read, as the conjunctions "Although" and "but" are used together unnecessarily. Punctuation errors include missing commas and misused semicolons, such as in the sentence, "It increases the test coverage; improve accuracy, saves time and money in comparison to manual testing". Issues such as these represent careless oversights that prevent the reader from comprehending the tutorial and can easily be fixed.

Redish states that important topics should be addressed first, and then content should be cut as much as possible to give users only the information that they need. This tutorial gives a relatively skim guide of the basics of software testing, and so succeeds at this. "Furthermore", "and hence", and "on the other hand" seem to inflate the document, however, and this is not helped by their missing the appropriate contextual punctuation. While these transitions can be used, eliminating them would strengthen the voice of the tutorial and better engage the user. A user reading about software testing is most likely not doing it for leisure, and it would be better for this tutorial to "cut the fluff", as Redish puts it. These slips in conversational language show a lack of attention to the audience of the tutorial. Ideally, the ideas would be presented in shorter sentences and paragraphs to optimize the way that the reader sees the information. This tutorial has, for the most part, reasonably worded sentences. It could increase its effectiveness by reducing the density of text further.

Software Testing Tutorial Critique

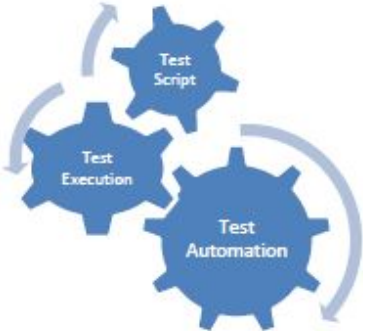
Illustrations/Visuals

to 

Figure -This woman seems confrontational

The opening image on the first page of the tutorial is a good element for drawing in the audience. By showing a professional woman, the document is made more welcoming to a broad audience of potential readers. However, the creators of this tutorial did not consider whether the image "engender[s] good feelings about the… site", a detail that Redish highlights as being important matching illustrations to message. The woman on the front page appears bored, even tired, setting the mood to one of frustration with a topic that is, by most standards, already overwhelming.

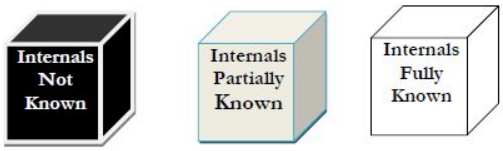
Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not. This activity results in the actual, expected and difference between their results. In simple words testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.



Figures .1 and 2.2-The text and image do not correlate

The triangle of Unit-Integration-System that is shown in Chapter 1 is a welcoming graphic for its color and simplicity. These work well in setting an easy tone for the tutorial. I found it confusing upon further inspection, although readers would probably not question its presence within the text. It is not discussed in the introduction to Chapter 1, and that causes it to float outside of the content, without offering any reinforcement to the page. Chapter 2 contains its own misleading graphic – a seemingly simplistic gear representation of the relationship between automation testing, test execution, and a test script. I found the chapter information difficult to understand not only because I was attempting to understand its relation to the graphic, but also because the graphic depicts a motor relationship that cannot physically work. The arrows indicate that the gears are turning against each other, as opposed to in synchrony. Redish mentions that illustrations that do not serve a purpose are "distracting" to the reader, who takes too "much time and effort trying to figure out what it represents". In this scenario, even more time is spent trying to understand a mechanical process that goes against logic. These visuals are effective in their size and simple esthetic, but frustrating when they do not offer clarification into the content of the surrounding text.

Software Testing Tutorial Critique



Comparison between the Three Testing Types

	Black Box Testing	Grey Box Testing	White Box Testing
1.	The Internal Workings of an application are not required to be known	Somewhat knowledge of the internal workings are known	Tester has full knowledge of the Internal workings of the application
2.	Also known as closed box testing, data driven testing and functional testing	Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application	Also known as clear box testing, structural testing or code based testing
3.	Performed by end users and also by testers and developers	Performed by end users and also by testers and developers	Normally done by testers and developers
4.	-Testing is based on external expectations -Internal behavior of the application is unknown	Testing is done on the basis of high level database diagrams and data flow diagrams	Internal workings are fully known and the tester can design test data accordingly
5.	This is the least time consuming and exhaustive	Partly time consuming and exhaustive	The most exhaustive and time consuming type of testing
6.	Not suited to algorithm testing	Not suited to algorithm testing	Suited for algorithm testing
7.	This can only be done by trial and error method	Data domains and Internal boundaries can be tested, if known	Data domains and Internal boundaries can be better tested

Figure 3-an effective visual: simple to detailed

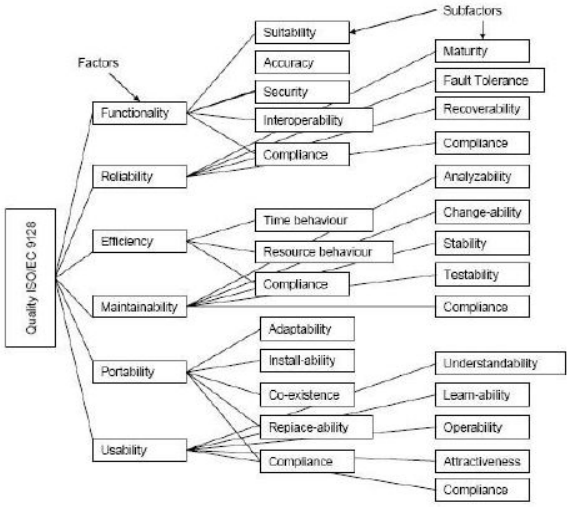


Figure 4-an overly confusing visual

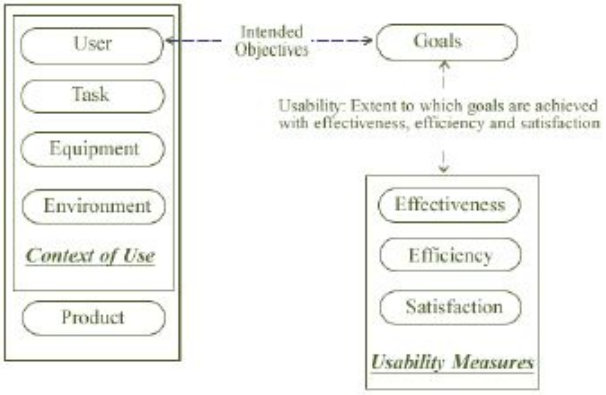
The most effective visual reference employed is the white, gray, and black box comparison shown in Chapter 3. This illustration makes only one point: that the color naming convention of the boxes refers to the amount that internal application workings are known by the tester. As a reader, I found it easier to remember this detail as it was presented in a basic visual format. The least effective visual reference is the sample traceability matrix. Because of the density of text, small font, and variation in shading, this image is difficult to see within the document. In order to improve this, I would have reduced the width of the columns in the matrix to increase font size and border thickness slightly. By giving the matrix a higher contrast appearance, it would become more visually appealing and easy to understand.

Design

As was mentioned earlier, critique of the design of the Software Testing Tutorial will be approached using the seven design principles provided by Dr. Jones. The first element is proximity, a design detail that this tutorial tends to implement successfully. Headings mostly sit near the text they address, although this distance is not consistent. The justified alignment of the text satisfies another design principle, keeping the page clean and visually organized. At one point, the text wrap around an image causes this to separate words so they are difficult to read. An example is shown, and the solution would be to resize the image in the body of the text to prevent this occurrence. The dynamic spacing between words and inconsistent proximity reduce the document's effectiveness, as it also reduces repetition, another important design principle. The tutorial maintains the overall font, text wrap, and table formats, but detracts from the impact of the document in the repetition inconsistencies that it overlooks. An instance of this that particularly bothered me was a bulleted list in which the bullet points and spacing varied from the rest of the document for no clear reason. These three design elements in particular were partially executed, but not enough to allow for full, distraction-free engagement by the user.

Software Testing Tutorial Critique

ISO/IEC 9241-11: Part 11 of this standard deals with the extent to which a product can be used by specified users to achieve specified goals with Effectiveness, Efficiency and Satisfaction in a specified context of use.
This standard proposed a framework which describes the usability components and relationship between them. In this standard the usability is considered in terms of user performance and satisfaction. According to ISO 9241-11 usability depends on context of use and the level of usability will change as the context changes.



ISO/IEC 25000: ISO/IEC 25000:2005 is commonly known as the standard which gives the guidelines for Software product Quality Requirements and Evaluation (SQuaRE). This standard helps in organizing and enhancing the process related to Software quality requirements and their evaluations. In reality, ISO-25000 replaces the two old ISO standards i.e. ISO-9126 and ISO-14598.

SQuaRE is divided into sub parts such as:

- ISO 2500n - Quality Management Division.
- ISO 2501n - Quality Model Division.
- ISO 2502n - Quality Measurement Division.
- ISO 2503n - Quality Requirements Division.
- ISO 2504n - Quality Evaluation Division.

The main contents of ***SQuaRE*** are:

- Terms and definitions.
- Reference Models.
- General guide.
- Individual division guides.
- Standard related to Requirement Engineering (i.e. specification, planning,

Figure 5-the justified text is obtrusive, but headings and bullet points are well-implemented

Contrast and similarity are well employed in the tutorial, as fonts and text sizes vary based on their section. The reader can clearly see the difference between the introduction to a chapter and the content within. By using sans serif fonts and headings with greater kerning than the body text, the tutorial is visually welcoming and sectioned off into manageable pieces. While paragraphs can be lengthy, they are for the most part only a few sentences in length: optimal for the user who may be skimming for the basic of software testing. The main contrast issue that arises occurs in the lists and tables employed by the document. The lists are lengthy, which goes against Redish's advice to reduce length for lists in which "people will not immediately recognize… the items". The tables contain large blocks of text as well, reducing their effectiveness in showing simple contrasts between concepts. Because of the poor execution, these oversights create obstructions in the content of the tutorial instead of aiding the user in understanding concepts.

The order of the concepts outlined is reasonable, as it is separated topically, with some sub-processes detailed chronologically. One instance of this is the separation of Levels of Testing and Testing Methods

from each other. They are explored separately, with Levels of Testing covering the order in which the level occurs within the Software Testing process. The tutorial is lacking visuals that demonstrate this. If there were more flowcharts or clear diagrams relating which levels contained which testing methods, the user would have a better sense of the order of information. In addition to strengthening the concepts, it would better guide the reader through reading the document. Overall, the tutorial does an acceptable job of ordering the design based on the overlapping nature of the topic.

Closing

The Software Testing Tutorial appears to be in its first draft, as the style is fully developed. It is inconsistent at points and rife with grammar errors, but these issues can be fixed with a basic copyedit. The major issue that needs to be remediated is the lack of connection of the visuals with the content of the text. While grammar is distracting immediately, ineffectual images create confusion upon further reference of the document or thorough reading. For one who has already begun learning the basics of software testing, this is fine, as it simply bolsters their knowledge of key terminology and relationships. However, a beginner might be put off in deciphering the message of the images, causing other design issues to become more aggravating as they search through the document.

As one who works around engineers every day, I found this tutorial to be reasonably easy to follow overall. It was certainly a less challenging topic to understand when explained in the document, as opposed to by someone in the field, who would be prone to using more jargon. I think that critiquing a tutorial in this fashion is useful to not only finding the problems, but understanding their relationship with the user. I learned that professional tutorials are not all held up to a standard that I would deem acceptable, as I would under no circumstances distribute this document in its current state. At the very least, I would copyedit it to remove the issues that create confusing sentences. This in-depth critique gave me a better idea of what types of issues in voice and design naturally occur when explaining technology-related processes.