

White Paper Submitted for STC 2010

Testing Heuristics & Cognitive Approach to Testing

Author:

Neha Thakur

ISTQB Certified Professional

QA Lead

neha.thakur@diaspark.com

Abstract:

“Human beings, viewed as behaving systems, are quite simple. The apparent complexity of our behavior over time is largely a reflection of the complexity of the environment in which we find ourselves.”

~ **Herbert Simon** (Winner Nobel Prize in economics 1978 mainly for research on decision making in organizations)

Have you ever thought that testing is so very much related to the solutions of everyday life, we just need to have a good common sense and accumulation of the expertise in a rational way gained over years of performing testing? Sometimes the solutions are right in front of our eyes in a relatively simple manner and yet we try to establish a complicated procedure laced with the processes with the help of copious analysis done by a dedicated team just to showcase the expertise.

In this paper, I will be showcasing the cognitive approach, different heuristic analysis, some practical demonstration to solve problems, why is there a growing need to use this approach, various advantages of using cognitive thinking and heuristic based testing.

In today's dynamic and competitive business scenario, the software functional demand is spiraling up while the timing of an application release is crucial and tight and overall don't forget the continuously changing requirements. Thus, looking at this scenario this paper will focus on following ideas/key points:

- Why we need software testing at all, various heuristic models and cognitive approach?
- Basic Rules to Optimize the Chance for Discovery of defects?
- How to assess risk and prioritize the testing efforts, testing the Results?

The paper will also disseminate knowledge about the benefits and challenges of implementing the cognitive testing as compared to the traditional testing.

Learning Objectives for the Audience:

- How to build and use the cognitive approach to test, it always works!!!
- Using Heuristic Analysis and creating a checklist for the purpose of testing.

Lessons to be learnt from the presentation:

A simple, strong and yet powerful approach to enjoy and enhance the different ways to perform software testing.

Contents

Abstract.....	2
1.0 Introduction.....	4
2.0 Testers Ten Pointer.....	5
2.1 Usability Heuristics.....	6
2.2 Defect predictions and Exploratory Testing.....	7
2.3 Usability Methods and Approaches.....	8
3.0 Conclusion.....	11
4.0 References.....	12
5.0 Acknowledgments - Biography of the author.....	13

1.0 Introduction:

Think of it like this: If you see someone walking down your street wearing a mask and carrying an automatic weapon, you might get suspicious. Our heuristic software is designed to recognize suspicious code. ~ Carey Nachenberg

Heuristic, in simple definition “is an adjective for experience-based techniques that help in problem solving, learning and discovery.” In more precise terms, heuristics stand for strategies using readily accessible, though loosely applicable, information to control problem solving. This definition implies that we learn majority times by trial-and-error in other terms by personal experiences; however, heuristic knowledge can be applied to complex as well as simple everyday problems. Human chess players use a heuristic approach.

We apply heuristic in our everyday problems, for e.g. if we are stuck in traffic we apply heuristics, which is just like solving a puzzle or finding the best and shortest path possible. We apply heuristics in our learning as well, we discover during the course of our education that which method is most appropriate and suitable for us, sometimes we use of combination of the strategies depending on the problem and the situation, for example some people learn by diagrams/ figures, some use mnemonics, some can memorize facts and data easily, some can memorize what they read, what they hear, what they see.

In similar fashion we apply heuristic in developing and testing a product/software. The most popular of all the heuristic application in designing an application/product/software is Object-oriented Design Heuristics and User-interface design heuristics. In applications that consist of an object-oriented model interacting with a user interface, the model should never be dependent on the interface. The interface should be dependent on the model. Some more sophisticated antivirus software uses heuristic analysis to identify new malware or variants of known malware.



2.0 Testers Ten Pointer:

A heuristic evaluation is a discount usability inspection method for computer software that helps to identify usability problems in the user interface (UI) design. It specifically involves evaluators examining the interface and judging its compliance with recognized usability principles (the “heuristics”). In object-oriented programming, a God object is an object that knows too much or does too much. The God object is an example of an anti-pattern. [Source: Wikipedia]

Below are few pointers that I have jotted down based on my experience which a good tester should possess, you can call it Ten Pointer Criteria, I have observed that it really works wonder when you have characteristics you need in front of you to analyse the testers qualities, abilities, skills.

Ten Pointer Criteria: PRM-Q-IAI-ESL

- Heuristic 1: Be Passionate, have zeal to exceed expectations and consistently delivering a better performance every time.
- Heuristic 2: Requirements docs are just a means to test the product/software based on the specs, trust your instincts, don't completely rely on them. Read “Testing Without a Map” by Michel Bolton
- Heuristic 3: Use [mind map](#) to brainstorm the ideas (A mind map is a diagram used to represent words, ideas, tasks, or other items linked to and arranged around a central key word or idea. Mind maps are used to generate, visualize, structure, and classify ideas, and as an aid to studying and organizing information, solving problems, making decisions, and writing.)
- Heuristic 4: Ask questions as much as you can, this applies to everyone, document the answers; create knowledge base/FAQs it helps every time. You can question the product which doesn't even exist functional.
- Heuristic 5: Innovation, creating innovative solutions and convert them into positive results.
- Heuristic 6: Accountability, taking ownership of the product/software, this is your product you are the stakeholder.
- Heuristic 7: Personal Integrity, all actions should be based on openness, trust and integrity/honesty.
- Heuristic 8: Think and Act like an Expert, set a MISSION and a VISION for you.
- Heuristic 9: Create a Story out of your observations; it helps in understanding the problem, performing analysis and reporting.
- Heuristic 10: Learning & Sharing Curve, Read More, Learn various methods, diversify, and Share your experiences with community.

Way to Remember: Passionately Require Mind Questions, Innovating Accounts of Integrity, Experts Story Learning!!

Various mnemonics have been discovered to help testers like [FCC CUTS VIDS \(test reporting heuristic\)](#) by Mike Kelly, “[cidtestdsfdpotcrusspicstmplfdsfscura](#)” is James Bach and Michael Bolton’s mnemonic to remember 36 important heuristics that helps a tester to ask questions that solves the problem, work on new test ideas, designing better tests. [HICCUPPS](#) by James Bach, a handy mnemonic for oracles. Karen N. Johnson devised a heuristic to plan regression testing, it’s called: [RCRCRC](#).

2.1 Usability Heuristics:

These are ten general principles for user interface design. They are called “heuristics” because they are more in the nature of rules of thumb than specific usability guidelines. A very good example is amazon.com

1. Visibility of system status:

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

2. Match between system and the real world

The system should speak the users’ language, with words, phrases and concepts familiar to the user, rather than system-oriented terms.

3. User control and freedom

Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue.

4. Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

6. Recognition rather than recall

Minimize the user’s memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7. Flexibility and efficiency of use

The system should cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed.

9. Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation.

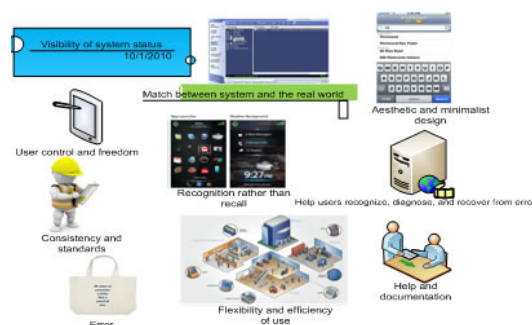


Figure 1: Ten Usability Heuristics

2.2: Defect predictions and Exploratory Testing:

Quality is often measured by number of defects that has got into the final product. Minimizing the number of defects and maximizing software quality requires a thorough testing of the software in question. On the other hand, testing phase requires approximately 50% of the whole project schedule. Testing is one of the most expensive, time and resource consuming phase of the software development lifecycle. An effective test strategy should therefore consider minimizing the number of defects while using resources efficiently.

Various approaches have been used in determining estimates of defective modules which will help in reduced testing cycle and management will be able to use limited resources effectively in a more optimized manner. Defect predictors based on linear regression, discriminant analysis, decision trees, neural networks and Naïve Bayes classification have been analysed in previous research.

Exploratory testing in structured way is one of the approaches based on the cognitive thinking and using heuristics. For this a random approach which most of us generally follows for finding exploratory defects has to be structured and scientifically thought of. In today's competitive business scenarios where timing of a product to the market are extremely critical and requirements are continuously changing, the complexities ever increasing James Bach through his Rapid Software Testing techniques follows a scientific method. Below describes the outline of the methods and approaches he follows:

- The Idea is think it in your mind and think scientifically, watch out for the mental traps, use questions, explain the behavior, and based on the experiences gained from testing use predictions, create a checklist.
- Complexity is in mind, don't be afraid of it! Observation is the key to a successful tester, observe and draw inferences, from observation you can see what is missing and what is not required, termed as extras this can be done from heuristic approaches.
- Consistently enhance your heuristics and list down the quality criteria, oracles and find out the way of coping difficult problems.
- Few Knows introduced are, Know the type of coverage for the testing you are performing on application, your strategies, approaches, procedures, innovate the process of testing everytime.

Few of the critical software practices for improving the performance are depicted below:



Figure 2: Critical software practices for improving the performance

2.3: Usability Methods and Approaches:

1. Cognitive Walkthroughs:

Cognitive walkthroughs are often conducted to resolve obvious problems before conducting performance tests. The cognitive walkthrough appears to detect far more potential problems than actually exist, when compared with performance usability testing results. Several studies have shown that only about twenty-five percent of the potential problems predicted by the cognitive walkthrough were found to be actual problems in a performance test. About thirteen percent of actual problems in the performance test were missed altogether in the cognitive walkthrough. Cognitive walkthroughs may best be used to identify potential usability issues to evaluate during usability testing.

2. Contextual Task Analysis:

A contextual task analysis, or contextual inquiry, is a user research method that applies ethnographic observation and one-on-one interviewing to understand the task procedures that users follow to reach their goals. The researcher silently observes the user at work in his or her natural work environment and notes any tools and people that support the user as they work toward task goals

Different models to help performing Contextual Task Analysis are:

- a. Sequence Model
- b. Artifact Model
- c. Physical Model
- d. Cultural Model
- e. Flow Model

3. Facilitated Brainstorming is much different than simply gathering in a small group and sharing ideas. Facilitated brainstorming is a generative exercise, which involves a moderator who keeps the group focused on the exercise and tracks all of the participants' ideas.

4. Focus Groups are best utilized as an evaluative tool, rather than a generative one (such as Facilitate Brainstorming and Charrettes). A Moderator facilitates a small group of 4 to 8 participants by demonstrating or showing them a product or concept. The participants are encouraged to freely give their honest opinions about the product, including suggestions to make it better.

5. Charrettes is an intense generative exercise that takes place over multiple days, and involves a multi-disciplinary group of participants. The purpose of a Charrettes is to efficiently generate design solutions for a project while considering diverse viewpoints of the stakeholders involved.

6. Structured, one-on-one interviews help researchers learn about users' attitudes and beliefs surrounding a website or application and specific tasks that the website or application supports.

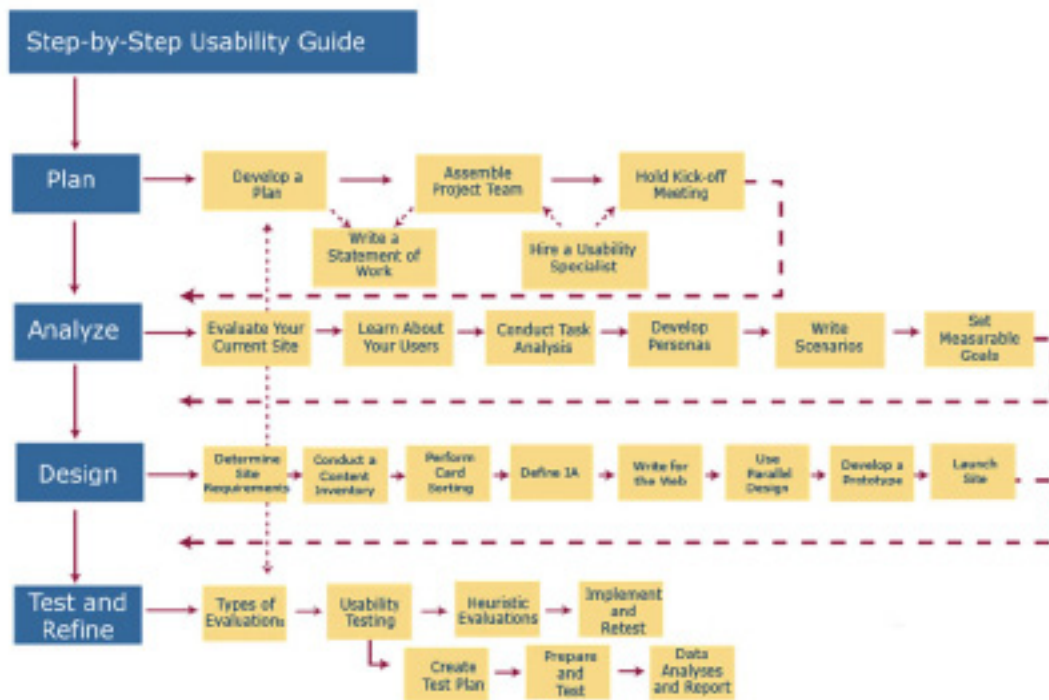
7. Participatory design engages stakeholders and end users in the process of solving a design problem.

8. Surveys are a good way to collect quantitative data about users' opinions about an application or website. Traditionally, surveys have been mailed to consumers through the postal service, and today, the internet is a great resource for distributing and collecting survey feedback from hundreds or thousands of participants in a short period of time.

9. Use Heuristic evaluation as mentioned in section 2.2

	Analyze	Design	Test
Card Sorting	✓	✓	✓
Contextual Interviews	✓	--	--
Focus Groups	✓	✓	✓
Heuristic Evaluation	✓	--	--
Individual Interviews	✓	✓	✓
Parallel Design	--	✓	--
Personas	✓	--	--
Prototyping	--	✓	✓
Surveys (Online)	✓	✓	✓
Task Analysis	✓	--	--
Usability Testing	✓	✓	✓
Use Cases	--	✓	--
Writing for the Web	--	✓	--

Figure 3: Usability Methods Vs Phases Source: Usability.gov]



4: Visual Map of Step By Step Usability Guide [Source: Usability.gov]

2.4: Assess Risk and Prioritize Testing Efforts:

“If there’s a magic to risk-based testing, it’s the magic of noticing the signs and clues all around you, about where the problems lie.” — James Bach

Risk based testing is defined as the prioritization of testing efforts in general and also the specific features and functions to be tested, based on the identified risks. Risk can be associated with various facets like the Product, Business, Customers, technology, Project, Process, Operational, Competition, security, Performance, Regulation, Compliance, Legal etc.

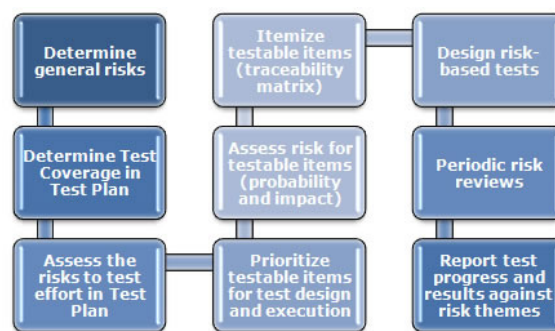


Figure 5: Risk Based Testing Cycle

Test Type	Extensive	Normal	Minimal	None	N/A	Comments
Installation Testing					X	
Data Integrity Testing		X				
Functional Testing	X					
Destructive Testing				X		To be addressed in next quarter's cycle
Security Testing	X					
Compatibility Testing	X					
Performance Testing	X					
User Interface Testing		X				
Data Conversion Testing					X	
Regression Testing		X				
Usability Testing	X					
User Docs Testing			X			

Figure 5: Planned test coverage for standard test types based on assessed risk

3.0 Conclusion

In today's fiercely competitive world, customers want high-quality software, but they want it quickly and at low cost. Faced with these pressures, some organizations try to rush testing, which endangers quality and increases long-term support costs. Other organizations over-test in hopes of catching every defect, which wastes money and can delay software's delivery to the market. In this paper we evaluated the characteristics and analyse the tester's qualities, abilities, skills. We listed and diagrammatically represented the usability heuristics followed across the world, after that we highlighted different ways of predicting defects, Critical software practices for improving the performance, exploratory testing strategies and approaches. We thereby worked on putting in one place all the USABILITY METHODS AND APPROACHES, depicting the Usability Methods Vs Phases, Visual Map of Step By Step Usability Guide. Then in the end we showed the Risk based Testing Cycle and the planned test coverage for standard test types based on assessed risk. By researching and experiencing, observing the need for today's tester where thinking rationale and using cognitive as well heuristic approaches will open up a brave new world of testing where limits will be defined by the tester.

4.0 References

Articles Sources	Author
http://www.developsense.com/articles/2005-01-TestingWithoutAMap.pdf	Michael Bolton
http://www.scribd.com/doc/17277488/Heuristic-Learning	Milena Tzakova
Object-Oriented Design Heuristics	Arthur Riel
Heuristic Framework for Evaluating Web 2.0 Applications	--
USING WORK-CENTERED SPECIFICATIONS TO INTEGRATE COGNITIVE REQUIREMENTS INTO SOFTWARE DEVELOPMENT	Jeffrey Wampler, Emilie Roth, Randall Whitaker, Kendall Conrad, Mona Stilson, Gina Thomas-Meyers and Ronald Scott
Ten Usability Heuristics	Jakob Nielsen
https://wiki.mozilla.org/Bugzilla:CMU_HCI_Research_2008 SOFTWARE DEFECT PREDICTION: HEURISTICS FOR WEIGHTED NAÏVE BAYES	Burak Turhan, Ayse Bener
Test Heuristics Cheat Sheet Heuristics & Frameworks	Elisabeth Hendrickson, James Lyndsay, and Dale Emery
http://www.spmn.com/critical_software_practices.html	--
http://www.usabilityfirst.com/usability-methods/card-sorting/	--
http://usability.gov/basics/uc_design_testing/index.html	--
Suffering from Release Remorse?	Betty Schaar

5.0 Acknowledgements

Name	Description
Kirti Joshi	Provided technical guidance, support and inspiration to write this paper, for reviewing the paper.

Biography of the author

Neha Thakur is passionate experienced software testing professional whose written articles and presentations have received awards and she recently spoke at the CAST 2009 conference in Colorado on stakeholder management and in STC 2009 on Future of Software Testing. She is an engineering honors graduate and is currently working as QA Lead at Diaspark INC, Indore, India and has previously worked as Business Tech Analyst with Deloitte Consulting. Her interests and expertise include test automation, agile testing and creative problem solving techniques. Neha is an active speaker on any topic on this earth related to testing and has presented papers on various topics at STC in last few years.

Software & Services

Microsoft office

Microsoft office SharePoint Server 2007
Microsoft Office 2007

Server

Windows Server 2003
SQL Server 2005

Development

Microsoft Visual Studio 2008
Microsoft SharePoint Designer

Technologies

Microsoft .NET Framework 3.0
InfoPath and Excel Services



USA - Edison - Corporate Office
200 Metroplex Drive, Edison, NJ, 08817
Tele: +1 732 248 8333 Ext. 6740
Fax: +1 732 248 8334

USA - New York Office
333, 7th Avenue, New York, NY, 10001
Tele: +1 212 972 2740

India - Indore - Global Delivery Centre
"Labh-Ganga" Building, Indore, 452003
Tele: +91 731 4273300
Fax: +91 731 2436613

© 2010 Diaspark Inc This white paper is for informational purposes only. WE MAKE NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT