Learning Styles and Exploratory Testing

Andy Tinkham
Florida Institute of Technology
andy@tinkham.org

Cem Kaner, J.D., Ph.D
Florida Institute of Technology
kaner@kaner.com

Abstract

Exploratory testing is widely done in software testing. However, it is performed in many different ways. This paper discusses our initial work into examining a tester's learning style as an indication of the types of actions she might use while doing exploratory testing. We use the Felder-Silverman learning styles model as a basis for presenting our hypotheses about how this aspect of one's personality affects how she performs exploration.

Bios

Andy Tinkham is a graduate student at Florida Tech, pursuing a Ph.D. in computer science. The main focus of his research is on exploratory software testing and the ways in which people can be trained to be good explorers. He has 8 years of experience in the field of software testing, where he worked both as a tester for companies whose main business was the production of software and as a consultant, helping companies set up successful automated testing efforts. Mr. Tinkham holds a B.S. in Computer Engineering.

Cem Kaner, J.D., Ph.D., is Professor of Software Engineering at the Florida Institute of Technology. His primary test-related interests lie in developing curricular materials for software testing, integrating good software testing practices with agile development, and forming a professional society for software testing. Before joining Florida Tech, Dr. Kaner worked in Silicon Valley for 17 years, doing and managing programming, user interface design, testing, and user documentation. He is the senior author of *Lessons Learned in Software Testing* (with James Bach and Bret Pettichord) and *Testing Computer Software* (2nd Edition) (with Jack Falk and Hung Quoc Nguyen) and of *Bad Software: What To Do When Software Fails* (with David Pels).

Dr. Kaner is also an attorney whose practice is focused on the law of software quality. Dr. Kaner holds a B.A. in Arts & Sciences (primarily Math & Philosophy), a Ph.D. in Experimental Psychology (Human Perception & Performance: Psychophysics), and a J.D. (law degree).

This paper was originally prepared for and presented at the Pacific Northwest Software Quality Conference (PNSQC) 2003

This research was partially supported by NSF grant EIA-0113539 ITR/SY+PE: "Improving the Education of Software Testers"

Introduction

Exploratory testing has attracted a great deal of interest in the software testing community[1]. This approach to testing typically involves testing, learning, and designing new tests as interacting activities, all occurring simultaneously. Almost all testers explore at times as they perform their jobs, whether they acknowledge it or not. For example, consider regression testing of bugs. To check whether a bug was fixed, the tester might *start* with the exact steps listed in the bug report, but after the program passes this simple test, the tester will probably try additional tests to check if the bug is completely fixed. These tests might vary the conditions that initially exposed the problem, or they might explore possible side effects of the fix. The details of these additional tests are not recorded in any test plan. The tester invents them as she works. In this case, she is being an explorer – she is exploring.

One of the puzzles of exploratory testing is that there appear to be different exploration styles. That is, it appears to us that there are many different available strategies for creating and using exploratory tests, that these are not mutually exclusive, and that individual testers (or explorers) seem to adopt a subset of these strategies.[2] This makes it difficult to characterize exploratory testing in terms of a narrow set of techniques, or a set of mental models or behaviors shared by all explorers.

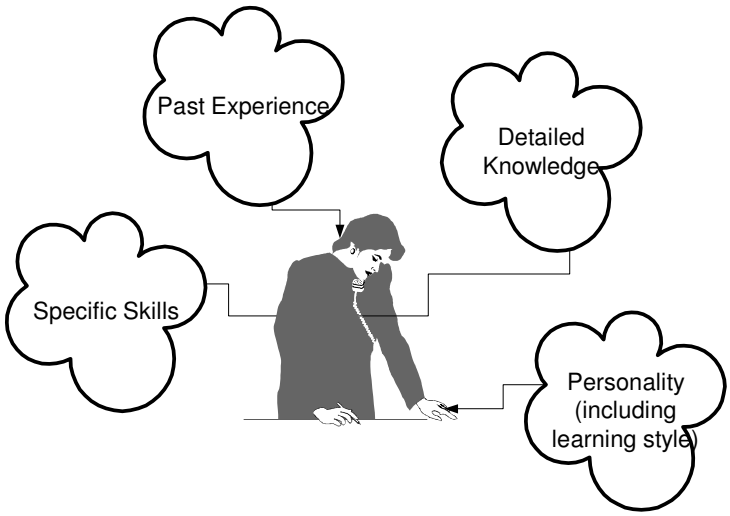
While doing exploratory testing, the tester generally has an overall purpose in mind that he is trying to achieve. Bach makes these purposes explicit and calls them charters[3]. An example charter might be to "check the UI against Windows interface standards". This charter serves to narrow the tester's focus and help him to better test the application by removing distractions and tangents that he might otherwise encounter. The charter does not, however, specify how he should accomplish the task. The specific actions required to satisfy the charter are left to the discretion of the tester, who determines them on the fly as he tests the application.

---

[1] For descriptions of exploratory testing, see (Tinkham & Kaner, 2003), (Bach, 2003), (Kaner, Bach & Pettichord, 2001), and (Agruss & Johnson, 2000)

[2] Kaner presented this idea in a set of 56 slides at the 7th Los Altos Workshop on Software Testing. His course slides (Kaner, 2002) are a more widely available, slightly updated source for this material.

[3] (Bach, 2003)



Many factors contribute to a tester's choice of exploration style

There is thus a large creative element to exploratory testing. Two different testers given the same charter may take two entirely different paths to achieve the stated goal. Neither path is necessarily better or worse than the other (though either one may find bugs that the other did not). How each tester develops her style (her pattern of adoption of exploratory test types) is probably based on several factors, including past experience, specific skills and detailed knowledge, and perhaps also aspects of her personality, such as learning style (the preferences the tester has for the ways in which she learns information).

This paper looks at how testers' learning styles could influence how they perform exploratory testing. Learning style seems like a relevant variable to us because a core activity of exploratory testing is learning about the software, including its weaknesses, potential failure modes, potential applications, market, configuration variability, and so on.

Several models of learning styles have been proposed. In this paper, we examine the Felder-Silverman model for insights that it might provide into the exploration process. We believe this model can provide guidance both in how an individual explorer approaches testing and how that explorer can expand his repertoire of techniques and thereby be more successful in exploration. We start by describing exploratory testing, and then discuss the Felder-Silverman model in more detail and apply its continua to exploratory testing. We have performed an informal survey of a small number of testers. While this survey was not statistically representative enough for us to be able to draw specific conclusions, it does indicate that there is a strong likelihood of patterns of correspondence between learning style tendencies and exploration technique choices occurring in a larger sample of the testing population, as well.

The research described in this paper is still in the early stages. While we believe the learning styles model makes sense and the points we make apply, we have not yet had the chance to validate the ideas. We are *very* interested in comments from the field to either support or refute the points we make. We are designing empirical research to study our hypotheses: we will make

our research results available at our lab's web site[4]. This research will involve observing and interviewing testers, in-depth study of the various learning style models, and the development and testing of exercises (each geared towards a particular learning style or combination of learning styles) to be used in training exploratory testers.

We present the hypotheses we have at this point because we believe the insights provided by learning style models can be immediately helpful to those doing exploratory testing. Our work might have implications for some other approaches to testing, but we are most interested in applying this to exploration because all of the investigative choices are under the explorer's control. Approaches centered on a specific technique, such as domain testing, are more constrained and so their application is less influenced by learning style.

Exploratory Testing

Exploratory testing is one of the most widely used approaches in the field of software testing[5]. It is also the least understood approach despite being used by most testers as part of their day-to-day activities. For example, any tester who deviates from a scripted process to characterize a defect he found or to verify a bug fix is doing exploratory testing. However, some testers are reluctant to describe their work as exploratory because they think the term connotes random work or that it is merely an excuse for testers to avoid the documentation and planning called for by traditional testing[6].

Competently done, exploratory testing is neither an excuse to avoid work nor a waste of time due to randomness. The essence of exploration is an active, risk-focused investigation of the product. Rather than designing tests early, when he is just beginning to learn what the product is, how it can fail, and who will use it to do what, on what platforms, the exploratory tester continually seeks out more information about the product and its market, platform, and risks – and designs tests to exploit the knowledge he has just gained. The explorer might use any test technique, and will probably use several different ones, in his quest to learn more about the product and its weaknesses.

Exploration Styles

Given a specific charter, individual testers will take substantially different approaches to fulfill that charter. One tester may start by creating models of the application. These models might be bare bones, quickly sketched on a piece of paper. Another tester might begin by brainstorming a list of different tests that could be executed and then work his way through the lists. A third tester might work from a list of failure modes (ways that we can imagine the product *might* fail), designing tests to determine whether the program actually does fail in those ways. While each approach could meet the charter of the testing being done, they will probably yield different bugs and different information about the product. Choosing a style of testing to achieve a specific

---

[4] http://www.testingeducation.org

[5] Those readers who may have less knowledge of what exploratory testing is are encouraged to read our earlier paper "Exploring Exploratory Testing" (Tinkham & Kaner, 2003) and James Bach's paper, "Exploratory Testing Explained" (Bach, 2003) for a more extensive explanation of the subject.

[6] While most articles do not explicitly describe exploratory testing in these terms, the author's attitudes can be inferred. See for example, the treatment of exploratory testing in the Software Engineering Body of Knowledge (SWEBOK, 2003) and papers extolling testing as a "discipline" (like (Drabick, 1999)).

charter and then developing effective tests within the style illustrates the creative demands of the exploratory testing process.

At the core of each style is the idea of questioning[7]. We think of each test case as a question asked of the application under test. Test design is a matter of developing good questions and asking them well. Exploratory test design is informed by the answers to previous questions.

Kaner identified nine different styles of exploration (most of which have several subsets) that he had seen exploring testers use[8]. Each style involves asking a different set of questions. The differences stem from the focus of the questions, and the skills and knowledge needed to ask the questions and interpret the answers. These styles involve using

- "hunches" (past bug experiences, recent changes),
- models (architecture diagrams, bubble diagrams, state tables, failure models, etc.),
- examples (use cases, feature walkthroughs, scenarios, soap operas, etc.),
- invariances (tests that change things that should have no impact on the application),
- interference (finding ways to interrupt or divert the program's path),
- error handling (checking that errors are handled correctly),
- troubleshooting (bug analysis (such as simplifying, clarifying, or strengthening a bug report), test variance when checking that a bug was fixed),
- group insights (brainstorming, group discussions of related components, paired testing), and
- specifications (active reading, comparing against the user manual, using heuristics (such as Bach's consistency heuristics[9])).

Why does one explorer rely on one (or a few) of these approaches while another adopts a different one? We think that part of the answer lies in the explorer's learning style.

The Felder-Silverman Learning Styles Model

A learning style is a person's "characteristic strengths and preferences in the ways they take in and process information"[10] These characteristics vary from person to person, and "may be strong, moderate, or almost nonexistent, may change with time, and may vary from one subject or learning environment to another"[11] for a given student. In their 1988 paper[12], Felder and Silverman proposed a model of learning styles that indicates a person's predilections on five continua: Sensory/Intuitive, Visual/Verbal, Inductive/Deductive, Active/Reflective, and Sequential/Global.[13]

---

[7] Kaner talks more about questioning strategies in his black-box testing course notes (Kaner, 2002)
[8] (Kaner, 2002)
[9] See (Bach, 1999)
[10] (Felder, 1996)
[11] (Felder, 1993)
[12] (Felder & Silverman, 1988)
[13] Readers who wish to determine their own learning style can visit http://www.ncsu.edu/felder-public/ILSpage.html for web-based and pencil-and-paper versions of Felder's *Instrument of Learning Styles*

The 5 continua of the Felder-Silverman model

A person's placement on the five continua might help her gain insight about why and when learning has been more pleasant and perhaps more effective, what blind spots she might have, and what techniques she might be underutilizing, that could help her learn more in situations that don't match her preferred environments.

Please note that a person's placement on the various continua is descriptive, not normative. There is no best pattern of results, no best learning style, no inherent superiority of any placement. Do not feel that you have to "be" a particular learning style. A person who prefers visual descriptions to verbal ones might still learn (if less enthusiastically) how to work through a strictly verbal specification. As Felder points out in his "Matters of Style" article[14], the model "provides clues, not infallible labels".

Felder presents a list of five questions that can be used to define (in part) a student's learning style:

1. "What type of information does the student preferentially perceive: *sensory*—sights, sounds, physical sensations, or *intuitive*—memories, ideas, insights?
2. Through which modality is sensory information most effectively perceived: *visual*—pictures, diagrams, graphs, demonstrations, or *verbal*—sounds, written and spoken words and formulas?
3. With which organization of information is the student most comfortable: *inductive*—facts and observations are given, underlying principles are inferred, or *deductive*—principles are given, consequences and applications are deduced?

[14] (Felder, 1996)

4. How does the student prefer to process information: *actively*—through engagement in physical activity or discussion, or *reflectively*—through introspection?
5. How does the student progress toward understanding: *sequentially*—in a logical progression of small incremental steps, or *globally*—in large jumps, holistically?"[15]

In the next sections, we explore the continua addressed by each question.

Sensory/Intuitive[16]

Felder defines a person who preferentially perceives sensory information as one who relies more on the information he receives through his external senses, while a person with a preference for intuitive information relies on his internal information (generated from memory, conjecture, and interpretation) and intuition. These preferences are equivalent to the sensing and intuiting types on the Myers-Briggs Type Indicator[17], and are derived from Carl Jung's theory of personality types.

According to Felder, strongly sensory learners are generally attentive to details. They are usually observant, and tend to favor facts and observable phenomena. They are apt to prefer problems with well-defined standard solutions and dislike surprises and complications that make them deviate from these solutions. Sensory learners can be patient with detail and are normally good at memorizing. They are generally good experimentalists.

Strongly intuitive learners, on the other hand, may be bored by details. They can easily handle abstraction, and are good at grasping new concepts. Often, intuitors strongly dislike repetition and they may be careless when performing repetitive tasks. Instead, they like innovation and are often imaginative and insightful. They respond best to thought problems, and like to emphasize fundamental principles and mathematical models. Intuitive learners often make good theoreticians, designers and inventors.

Visual/Verbal

Visual and verbal learners differ in how they best receive information. Visual learners retain more information they get from visual images such as pictures, movies, diagrams or demonstrations, and may have problems remembering information they simply hear. Verbal learners retain more information they hear (or read) such as lectures, written words, and mathematical formulas. "Most people (at least in western cultures) … are visual learners."[18]

Inductive/Deductive

The inductive/deductive dimension deals with how a learner organizes information. An inductive learner prefers to work from specifics and derive the generalities, while a deductive learner starts with the generalities and applies them to the specific situations they encounter. Thus, an

---

[15] (Felder, 1993)
[16] For examples of engineering students illustrating the sensory and intuitive aspects, see Felder's character sketch of two students, one of whom (Stan) is a sensory learner and the other (Nathan) is an intuitive learner, in his "Meet Your Students: 1. Stan and Nathan" article (Felder, 1989) available at http://www.ncsu.edu/felder-public/Columns/Stannathan.html
[17] (Felder & Silverman, 1988)
[18] (Felder, 1993)

inductive learner generally likes to be given a set of facts, observations, or an example and then to tease out the fundamental principles that support the example. Deductive learners prefer to learn the basic principles and then determine how best to apply these principles to situations they encounter.

Induction is described by Felder and Silverman as "the natural human learning style. Babies don't come into life with a set of general principles but rather observe the world around them and draw inferences… Most of what we learn on our own (as opposed to in class) originates in a real situation or problem that needs to be addressed or solved, not in a general principle…"[19] Inductive learners will often need to see the motivation for learning a piece of information before they can learn it and they need to see an event before they can understand the underlying theory about it.

At the other end of the continuum, Felder and Silverman describe deduction as "the natural human teaching style, at least for technical subjects at the college level. Stating the governing principles and working down to the applications is an efficient and elegant way to organize and present material *that is already understood*." Deductive learners learn best by starting at the fundamental principles and then learning the applications of these principles to real life and the problems they encounter.

Active/Reflective

People differ in how they process information once they have received it. Some people need to use the information right away for it to stick in their memories, while others need to think about the information and figure out how it fits into their mental framework before they can use it. The Active/Reflective dimension in the Felder-Silverman model covers this difference.

Active learners want to do something with information as soon as they get it. They might discuss it with others, either as peers or by explaining it to someone else, or they might experiment with the information they have received. They tend to like to work in groups and like to find solutions that work and in general are the people who design and carry out the experiments. If active learners had a trademark phrase, it could well be "Let's try it out and see what happens."

Reflective learners prefer to think about information before they use it. They prefer to work alone or with at most one other person who they trust. They need time to mentally manipulate the information to see what they can get from it. In general, reflective learners are the people who define the problems that need to be solved. The trademark phrase for reflective learners could be "Let's think it through first".

Sequential/Global[20]

The final dimension in the Felder-Silverman model is that of sequential versus global learning. This dimension deals with how learners "get" the information they are learning. Sequential

---

[19] (Felder & Silverman, 1988)

[20] For examples of engineering students illustrating the sequential and global aspects, see Felder's character sketch of two students, one of whom (Susan) is a sequential learner and the other (Glenda) is an intuitive learner, in his "Meet Your Students: 2. Susan and Glenda" article (Felder, 1990) available at http://www.ncsu.edu/felder-public/Columns/Susanglenda.html

learners learn material in a logically ordered progression, learning little bits as they go, and incrementally building on the knowledge they have already learned. Global learners, however, tend to learn in chunks. They will spend some time being lost, then suddenly everything will come together and they will understand the concept.

For sequential learners, each piece of information builds logically on the previous ones. Sequential learners are strong in convergent thinking and analysis, bringing ideas in together. They follow "linear reasoning processes" when they solve problems, and their solutions are often the sort that make sense to other people. Sequential learners often have little trouble in school, as they learn best when material is presented with increasing complexity and difficulty and they can work with material that they only partially or superficially understand.

Global learners instead tend to see the big picture. They spend a period of time not understanding the material, but then a critical piece of information arrives and everything falls together for them. Global learners tend to be more apt to see connections beyond those presented (often to completely different disciplines than the one they are learning in at the moment). When a global learner is solving a problem, she may seem to leap directly to the solution (possibly skipping intermediate steps) and be unable to explain how she got there to other people. Global learners tend to need to be able to fully understand the material before they can work with it, however, and this can lead to problems in school. Once they have this understanding, they can very quickly assimilate additional related information and often are strong in divergent analysis and synthesis.

Since the publication of their original paper in 1988, Felder has made two changes to the model. The first change was the dropping of the "Inductive/Deductive" continuum. We believe that this continuum provides insight for exploratory testers and so will treat the model as if it still contained this continuum for our purposes. The second change Felder made to the model was a word change for the Visual/Verbal dimension. Originally, this continuum was called Visual/Auditory. We will use the current terminology (Visual/Verbal) throughout the rest of this paper.[21]

Applications to Exploratory Testing

Now that we have reviewed the Felder-Silverman model, we can apply this to exploratory testing, looking at the potential exploratory styles of someone who has a strong preference for a given aspect (and ignoring, for now, the interaction of aspects from different continua). Again, it should be stressed that there is no superiority or inferiority implied in a specific aspect or the lack thereof. Each aspect brings strengths and weaknesses to exploratory testing, and the well-balanced test team will have members whose learning styles complement each other.

Sensory/Intuitive

So, how would a person who was strongly sensory-based or strongly intuitive-based approach exploratory testing? The sensory-based person (who you will remember likes details and well-defined solutions to problems, while preferring information gained from his senses) might focus on his actual observations of the software. The intuitor (with her preference for internally

---

[21] Readers interested in the reasoning behind these changes should read the preface to (Felder & Silverman, 1988)

generated information) might then focus instead on her internal model of the software she is testing.

The two testers will probably also vary in their approach to performing their testing. The stereotypic sensor will generally apply "rules and tools" – solutions that have worked in the past for specific bugs that he can apply in his current testing to determine whether a particular bug exists. His testing would take the form of a series of experiments on the application. These experiments will tend to be of the form "Does this specific bug exist". A strongly sensing tester may also be more likely to begin testing the product before he creates any models of the software (mentally or otherwise). He is more likely to consult the specification and other reference material, and to experiment with the conformance of the documentation and the product. The learning done by a strong sensor is apt to be more based on experiencing the product. Given the sensor's preference for well-defined standard solutions, he is probably going to be more inclined to develop a standard pattern for approaching exploratory testing. This pattern can develop into a mental script, shifting the tester's focus from exploratory testing to scripted testing.

The stereotypic intuitor is more likely to approach the problem instead by applying different theories of error to the software. She will take a risk-based approach to her testing, thinking of ways in which the software can fail and then thinking of tests which will show whether the software actually does fail in that manner. This is different from the sensor's testing for specific bugs in that the intuitor is not focusing on bugs she has seen in the past. Instead, she is using her experience and understanding of the application she is testing to think about all the different failure modes of the application. It is a subtle difference between the two—a difference primarily of level of thought. The sensor is taking specific examples of bugs and checking for them. The intuitor is looking for more general possibilities of failure (each of which may have multiple bugs associated with it) and then deriving tests that could trigger a particular failure mode. She will usually like it when her mental model of the software is proven to be incorrect, often viewing the act of bringing her model back in line with reality as a challenge to be tackled with great relish.

The intuitor is also more likely to begin testing by building a model of the software. This model could be a state-chart, a mapping of the software to its market, or some other representation of some portion of the system. While the sensor is perhaps doing research designed to predict the behavior of the system, the intuitor might instead be doing research to define and then refine her models, with the intention of then evaluating the model against the product.

Finally, the two testers will most likely find different exploration styles. The sensor could find the various attacks described by Alan Jorgensen[22] and James Whittaker[23] appealing, while the intuitor might be more drawn to the modeling techniques described by Elisabeth Hendrickson[24].

---

[22] (Jorgensen, 1999)
[23] (Whittaker, 2002)
[24] (Hendrickson, 2002)

Visual/Verbal

The major difference between exploratory testers with a strong preference for visual learning and those with a preference for verbal learning might reflect the internal mental model that the testers use. Visual learners will tend to work off an internal model that is picture-based. This model could be a set of UML diagrams, flowcharts, or even mental screenshots. They will also tend to work off visual portrayals of the steps in the tests they are executing. These portrayals may run like a movie in the visual learner's head. Alternatively, visual learners may make diagrams and pictures for their notes as they explore.

Verbal learners would instead use a textual model for their testing. These models might take the form of a textual description of the system (perhaps a textual use case format) or they may take the form of a remembered conversation. The model will be based around words – the tester will use words to describe the system to themselves and words to describe each step in the process.

In addition to their internal models, these testers may also differ in the types of specification documents they try to get from their analysts and developers. Visual testers probably will be more comfortable working with visual models of the system – the state charts, UML diagrams, flowcharts and other representations the people designing and building the software use to help clarify things for themselves. Verbal learners, on the other hand, are likely to be happier taking the textual specification for the system and wading through it, learning as they go.

Inductive/Deductive

An inductive learner may adopt an approach to testing where she gathers as many specifics (such as techniques, potential defects, changes made to the application, and application history) as possible and generalize them to the application. A deductive learner might instead approach testing by keeping a collection of general principles and heuristics and find ways to specifically apply these generalities.

The inductive learner will likely take advantage of historical data – looking at the available defect reports, the technical support database, published articles about the software being tested and about similar programs, and any other historical documents that she can get a hold of. From these documents, the inductor will derive a set of specific guidelines that she then can use to guide their testing. For example, the application being tested may have a history of defects in one particular area. An inductor would take the specific fact of the large number of defect reports and generalize it to show that there could still be a large number of defects remaining in that application area, and thus focus more attention on that area than on another area which has had no defects reported historically. While an inductive learner is using heuristics for this approach, it is less apt to be a deliberate usage than we believe the deductive learner will have.

The deductive learner starts with a collection of general heuristics and guidelines and then consciously applies them to the application. Many of the traditional techniques of software testing are deductive – the tester learns the basic skill (such as equivalence partitioning) and then determines how to apply it in the specific situation of her testing.

We expect there will also be differences in how deductive and inductive testers use bug taxonomies[25] and risk lists. We expect the deductive tester to gain familiarity with the categories and then be able to come up with new examples within those categories. The inductive tester, on the other hand, is expected to gain an understanding of the various list elements and then be able to see new ways of categorizing them.

Active/Reflective

Active and reflective testers differ most in how they execute tests. An active tester will usually do very hands-on testing. She often will perform many test cases rapidly and will view each test case as an experiment, asking, "What happens if I do this?" each time. An active tester will also tend to be more visibly a part of a testing group, often bouncing ideas and results off other members of the group to solicit their feedback.

A reflective tester, on the other hand, is apt to do far fewer tests. A thought process will precede each test case where the tester is thinking through the test. Reflective testers make up for their lack of speed in test execution by executing the "good" tests that are most likely to find bugs, however. Most reflective testers will probably tend to prefer to work alone or with at most one other person, and so may seem anti-social or outside the group. This isolation and thinking should give them the time to develop more complex tests and scenarios to apply to the application, and thus they should be encouraged to take the time they need.

Sequential/Global

The last pair of aspects we have to consider is the sequential and global aspects. A sequential learner builds information and knowledge in a logical progression, while a global learner needs critical pieces of information in order to get the understanding of the subject.

The sequential tester will seem to get off to a faster start. He will build test plans as he goes, step by step. Not having a piece of information will not normally prove to be a problem for a sequential tester, as he will work with the information that he does have. He also will be able to explain his tests clearly to people after he has performed them. In general, a sequential tester's test cases will grow in complexity over time as he builds a deeper understanding of the system.

A global tester will get off to a slower start. He may have problems understanding the point of the application (or their area within it) and need to be shown how to use the application in order to have any idea how to test it. Once he gets the piece of information that brings it all together for him, however, he quickly becomes able to create detailed, complex tests that often draw on connections that other people on the testing team have not seen.

Wrap-up

We are interested in the Felder-Silverman model of learning styles because it gives us hints about why different testers adopt different exploratory strategies. If those hints are validated, we will be better able to create effective sets of lesson plans and exercises to train explorers. Testers with exploratory testing experience might be more able to discover their blind spots or identify

---

[25] A taxonomy is a listing with the elements broken up into categories. For more information, see (Vijayaraghavan, 2002, 2003)

techniques that would not otherwise occur to them. Test managers will be more able to determine where there may be gaps in their testing teams and what kinds of skills they need to bring in to balance the team.

Much work remains to be done before the impact of learning styles on exploratory testing is completely understood. Each individual aspect needs to be explored in more detail for connections. The aspects must be looked at in combination (for example, how is someone who is strongly intuitive AND strongly verbal going to differ from someone who is just strong in one of those two aspects?). Experimentation must be performed to validate the claims. We believe it will be an exciting process; one that we hope will attract the interest of many people in the field. As we further our research, we will be documenting our research in two places – papers on the lab web site (http://www.testingeducation.org) and in our weblogs (at http://blackbox.cs.fit.edu/blog/andy and http://blackbox.cs.fit.edu/blog/kaner).

References

Agruss, C., & Johnson, B. (2000). Ad Hoc Software Testing: A perspective on exploration and improvisation. (*http://www.testingcraft.com/ad_hoc_testing.pdf*)

Bach, J. (1999, August 26, 1999). *General Functionality and Stability Test Procedure*. Retrieved July 29, 2003, from http://www.satisfice.com/tools/procedure.pdf

Bach, J. (2003). Exploratory Testing Explained. (*http://www.satisfice.com/articles/et-article.pdf*)

Drabick, R. (1999, October 27, 1999). *Growth of maturity in the testing process*. Retrieved July 29, 2003, from http://www.softtest.org/articles/rdrabick3.htm

Felder, R. M. (1989). Meet Your Students: 1. Stan and Nathan. *Chemical Engineering Education, 23*(2), 68-69 (*http://www.ncsu.edu/felder-public/Columns/Stannathan.html*)

Felder, R. M. (1990). Meet Your Students: 2. Susan and Glenda. *Chemical Engineering Education, 24*(1), 7-8 (*http://www.ncsu.edu/felder-public/Columns/Susanglenda.html*)

Felder, R. M. (1993). Reaching the Second Tier: Learning and Teaching Styles in College Science Education. *Journal of College Science Teaching, 23*(5), 286-290 (*http://www.ncsu.edu/felder-public/Papers/Secondtier.html*)

Felder, R. M. (1996). Matters of Style. *ASEE Prism, 6*(4), 18-23 (*http://www.ncsu.edu/felder-public/Papers/LS-Prism.htm*)

Felder, R. M., & Silverman, L. K. (1988). Learning and Teaching Styles in Engineering Education. *Engineering Education, 78*(7), 674-681 (*http://www.ncsu.edu/felder-public/Papers/LS-1988.pdf*)

Hendrickson, E. (2002). A Picture's Worth a Thousand Words: How to create and use diagrams and maps to improve your testing. *STQE, 4*(5), 26-32

Jorgensen, A. A. (1999). *Software design based on operational modes.* Unpublished PhD., Florida Institute of Technology, Melbourne, FL.

Kaner, C. (2002, Summer). *A Course in Black Box Software Testing (Professional Version)*, 2003, from http://www.testingeducation.org/coursenotes/kaner_cem/cm_200204_blackboxtesting/

SWEBOK. (2003, May). *Software Engineering Book of Knowledge*. Retrieved July 29, 2003, from http://www.swebok.org

Tinkham, A., & Kaner, C. (2003, May 15). *Exploring Exploratory Testing.* Paper presented at the STAR East 2003 Conference, Orlando, FL, USA (*http://www.testingeducation.org/articles/exploring_exploratory_testing_star_east_2003_paper.pdf*)

Vijayaraghavan, G. (2002). *Bugs in Your Shopping Cart: A Taxonomy*. Retrieved July 30, 2003, from http://www.testingeducation.org/articles/bugs_in_your_shopping_cart_a_taxonomy_paper_isqc_sep_2002_paper.pdf

Vijayaraghavan, G. (2003). *Bug Taxonomies: Use Them to Generate Better Tests*. Retrieved July 30, 2003, from http://www.testingeducation.org/articles/bug_taxonomies_use_them_to_generate_better_tests_star_east_2003_paper.pdf

Whittaker, J. A. (2002). *How to Break Software*. Boston: Addison Wesley.

Acknowledgements

The authors would like to thank the following people for their assistance with this paper:

- James Bach
- Kit Bradley
- Sue Carlson
- Elisabeth Hendrickson
- James Lyndsay
- Jean Marchant
- Brian Marick
- Max McNaughton
- Barb Nolan
- Erik Petersen
- Alan Richardson
- Dianne Runnels
- Melissa Strader
- Ruku Tekchandani
- Heather Tinkham