Deriving Testing Strategies from User Experience Design Practices

Venkat Moncompu
venkatams@yahoo.com

Abstract: User-experience-design (UXD) and Quality-by-Design (QBD) both involve a shared tool-set. Both methodologies use conceptual, behavioral and interaction models to facilitate understanding the user needs and the context of use. This paper suggests that an exploratory tester can exploit the synergies implied by the shared tool-set.

Using a case study, this article attempts to demonstrate how the insights gained through the analysis and review of the UX design artifacts aids the discovery and learning during exploratory testing. It is compelling to see the usefulness of scenario-based contextual design in providing a foundation for unscripted exploratory testing. The case study further shows the gap that exists in these teams rich in engineering culture that is at odds with the exploratory testing which is more of an art than science.

Biography: *Venkat Moncompu is a manager with West Monroe Partners, LLC where he is advocating quality by design in the software development projects that he engages in. Venkat has 15 years of experience in the software industry and interests include quality management, agile test automation, user experience and design thinking. West Monroe Partners, LLC is an international, full-service business and technology consulting firm focused on guiding organizations through projects that fundamentally transform their business.*

Venkat has a master's degree in Engineering from Arizona State University. Venkat has published articles and spoken at various forums on Quality Assurance in Information Technology

Copyright. Venkat Moncompu, 2012.

1. Introduction

Quality-by-design is based on the premise that quality is built-in and verifying quality early in the design reinforces the conformance to quality as the solution is built. UX design methodology starts by reviewing the value proposition and identifying the user segments. The user research then leads to a conceptual understanding of the scenarios and contexts of use in support of the value proposition. The behavioral and interaction design that emerges is a model of the system in the context of its use. The elements that constitute the model are validated by usability tests. These models also drive the user acceptance functional testing when supplemented by task analysis and user goals. Evaluation of the design through usability tests for an understanding of the user interactions are very valuable in discovering new scenarios and gaps in how the system would be used. This is a very powerful and profound recognition of the user acceptance behavior. The findings of the usability tests and insights gained on the user behavior provide guidance to exploratory tests which are a supplement to requirements driven testing.

2. Acceptance Testing

The acceptance testing validates the fitness for use of a system. The factors that influence these usability tests are

- The environments and operating conditions of use,
- The prioritized user scenarios that provide highest value to the user and
- The attributes of quality that satisfy the specific user's needs in the context of their objectives

The motivations, user goals, user roles, functional understandings of the domain and usability tests guide a user centric design process. By iterating through prototypes and high fidelity models, the navigation flows through the system forms the basis of the acceptance tests. User stories build the relationship between user tasks and the goals by mapping user roles and personas to them.

The software engineering team creates elaborate documentation in waterfall methodology that fail to capture these user perspectives at all. We run into limitations when we try to capture these behavioral characteristics in any number of use cases and requirements. However, guided by the UX design these limitations are overcome by actively engaging the testers with the design teams up-front during the evolution of the design (co-location, pair-programming). The requirements in agile development methodology are almost identical to the UX design work items. They are merely presented in a slightly different manner. Therefore, it seems logical to engage the testers with the UX design team earlier in the design process to identify these acceptance conditions, criteria and tests for the definition of what "Done" means.

One of the best practices of agile development is the adoption of acceptance test driven development (ATDD). Unless these are designed as "intended," automated acceptance tests run concurrently as the development team implements the user stories, adds little value to the development effort. The opportunity to design tests efficiently presents itself during the evolution of the UX design itself (lockstep) rather than forced to trail the development as in traditional approaches. The scripted tests rely on the documented requirements which can only specify the systemic view for acceptance. Reviewing the emerging design with the insights gained from user research into the context of their use, the testers bring the acceptance tests to life. The testing team can derive objectives for exploratory tests with a better understanding of the personas, usage scenarios and the interaction model.

During this user research process, additional details of user behavior emerge such as preferences by a certain persona to perform a task and navigation towards a goal which might not otherwise be obvious. The interaction models provide Meta model information that improves the testing efficiency by uncovering operating conditions, user data needs and motives for navigation flows that are not documented. With unscripted exploratory tests, learning and adapting tests based on the system behavior in the context of its use become intrinsic to the *charter-test-retrospective-adapt* cycle.

Exploratory testing inherently deals with discovering these experiences as the users navigate through the application based on the scenarios and tasks. Given the tangible benefits of front-loading agile development with a robust UX design process, it is ironic that the agile development community has not adopted the practice of UXD in a more embracing manner than they currently do. Part of the problem is that management and the developers focus not on designing creative user experiences but on delivering software engineering service. In the few instances where they do have a UX design practice, there seems to be a dysfunctional relationship between user centric design that is user oriented and solution design that is customer focused where the user and customer are not the same. It then becomes more happenstance than the consequence of a well planned strategy. Secondly, the problem is the inability of the engineering culture of the teams to acknowledge that exploratory testing is as much an art as it is science.

For the benefit of the reader of this article, before visiting the mechanics of exploratory testing to show this correlation in detail and the patterns that emerges therefrom, let me first review the techniques and tools used in UXD in more detail towards deriving acceptance test conditions and criteria.

3. UX Design & Acceptance Testing

The UX designer starts at the very inception of the project by identifying the business value proposition which talks to the uniqueness of the offering and the differentiating attributes of the solution from the competition. The high level objectives of the business problem are evident from the analysis of the value proposition.

These objectives then become the focus areas of higher priority when designing the tasks and test steps around these. The value proposition also discusses the beneficiary i.e. the user base that the solution caters to (emerging market or existing customers). The user segmentations and identification of personas to fit the user segments are then identified during user research. The reasons and motivations of using the system and how the users interact with the system in their working environment are recognized. The pragmatic tester relies on a host of factors such as biases, observations, hunches, instincts, heuristics, and experience to supplement the approach to exploration. To keep these open-ended drivers focused irrespective of the approach (be it free-style, scenario-based, strategy-based or feedback-driven), the understanding of the user and the context of use in delivering value is paramount. For exploration, the tester leverages tools such as checklists, mind-maps, and cheat-sheets to build a framework to base her focus on.

The tester adopts the personas just as the business analyst captures the actors for the use cases. The user tasks lead to the exploration of layout and navigation design with low-fidelity prototypes to begin with even before higher fidelity working prototypes are built. At this juncture, the acceptance tests in concrete system interaction steps begin to emerge. Before fleshing out the details, the tester outlines a matrix of tasks against goals and user personas to form a dimensional matrix of test coverage.

User stories are the preferred tool for capturing requirements because they help describe the users' view of the system behavior rather than a system-centric view described in use cases. Whereas use cases serve as a contract between the various development teams and the stakeholders, user stories are flexible and serve as a negotiable framework for engaging the users to describe user-centric view of the system behavior. User stories are typically takes the form a construct such as:

"As a <role> I want to <goal> so that <benefit>"

What remains then is the acceptance criteria which are the objectives derived from task analysis. It is here that significant contributions have been made in the recent years to design by specification. For ATDD, tools that support Gherkin like domain specific language (DSL) are introduced and the acceptance tests created as shown in example 1.

Given an amateur bird watcher has launched the application
And has browsed to the bird she has spotted in the gallery
When the user begins to record her observations
Then the system should pre-populate all of the information presented to the user in the gallery

Example 1: Using gherkin DSL to capture acceptance test criteria

What is obvious from this example is that the construct is essentially in English language where the user role is clearly defined and the task she is trying to accomplish is also captured. The user research helps fill the role of the user with the appropriate persona for the task's objective. Then the conditions for acceptance of performing this task within the context are identified. The ingredients are essentially the same – a user goal, the task steps and the acceptance conditions in the form of expected behavior. This then becomes specification by test (or example). These acceptance tests are scripted with the steps known *a priori*.

Exploratory testing on the other hand is the unscripted testing of the system by continuous learning and discovery of the system behavior under conditions of use. The factors that differentiate it from *random* testing are the insights testers learn to keep the tester focused on the specific navigation patterns and flows that users take while interacting with the system. The testers rely on heuristics, biases, observations, hunch, intuition, conjectures to comprehend the user's perception of the interface to accomplish their stated goal. The different types of exploratory tests are freestyle, scenario-based, strategy-based and feedback-driven. To provide a structure and framework around exploration, it is very important for the testers and developers to engage with the designers early even as the user experience team begins the user research process. By studying the emerging user personas whose task analysis drive the interaction design, testers derive testing strategies and put the system through its paces in the context of their use. A few different time-boxing techniques have been proposed such as session-based testing where the testing charter is chosen so as to fit into each session of a specific time interval to do the exploration. In the case study that follows, I attempt to derive information based on user research and supporting user experience design artifacts such as wireframes to gain insights into how the system would be acceptable for use.

User stories cross-referenced in this way with the matrix and supporting document such as checklists and mind map help the tester to test scenarios of use by the different user profiles exhaustively. Focusing on the goals, the tester can adapt the tests with the feedback from the system behavior. Tests are then vectored to be more effective but within the realm of possible acceptance of use.

4. Case study

The mobile application under consideration is the bird spotter application that caters to at least 3 different user archetypes. An amateur who is not familiar with bird species, a researcher who knows bird species by name and characteristics, and an avid bird watcher interested in the seasonal migratory patterns and tracks bird migration through social networking channels with others in the spotter community.

A closer look at these personas reveals quite a lot about the individual user needs and the way the application might be used. Based on the personas captured by the UX design team, the environment in which they might be using the application can be inferred.

	Amateur bird watcher likes to feed birds for example while out and about with his son. Knows little about birds and therefore has to use the browsing features in the application to help identify birds by their visual and audio characteristics. He would use the application in an urban setting or residential neighborhoods and therefore would only be interested in birds that are commonly seen.
	The **avid bird watcher** wants to spot exotic birds that are seasonal and migratory. He wants to share the spotting information with others in the community. Looking to use the application while on the road or outdoors. He would like to report his observations to the conservatory once he returns home or office and so caching the recorded observations to the device storage would be important.
	The **research student** wants to complete the research on birds and complete the study with spotter data and information. She will be using the application in camp sites and remote areas. She would like to collate all the information she has captured over time and map patterns by geographic location. And so, she would like to navigate the application in such as manner as to derive geographical information to the observations and sightings.

Table 1: Personas for the bird spotter application

In order to understand the application states better, the analysis of the user personas reveals additional requirements: presenting such information as seasonal migratory patterns and a sampling of their bird song and images of the bird to help user in identification.

The avid bird watcher wants to get real-time data feeds of spotting information in her vicinity through a social media channel. She would want to control access to her friends and filter by attributes such as distance or relevance. Here also, we see ample opportunity to explore the application based on the user needs.

The research student persona likes to record detailed information as the lower bound on the spotter count, specific species of interest, direction of migration, geo-tagging of location to compare with a seasonal database such as those maintained by the Audubon Society, for instance.

Based on the layout and interaction design, additional tests can be explored given these additional contexts of use aided by the personas. These additional features increases complexity and the exploratory tester has many opportunities to improve testing efficiency.

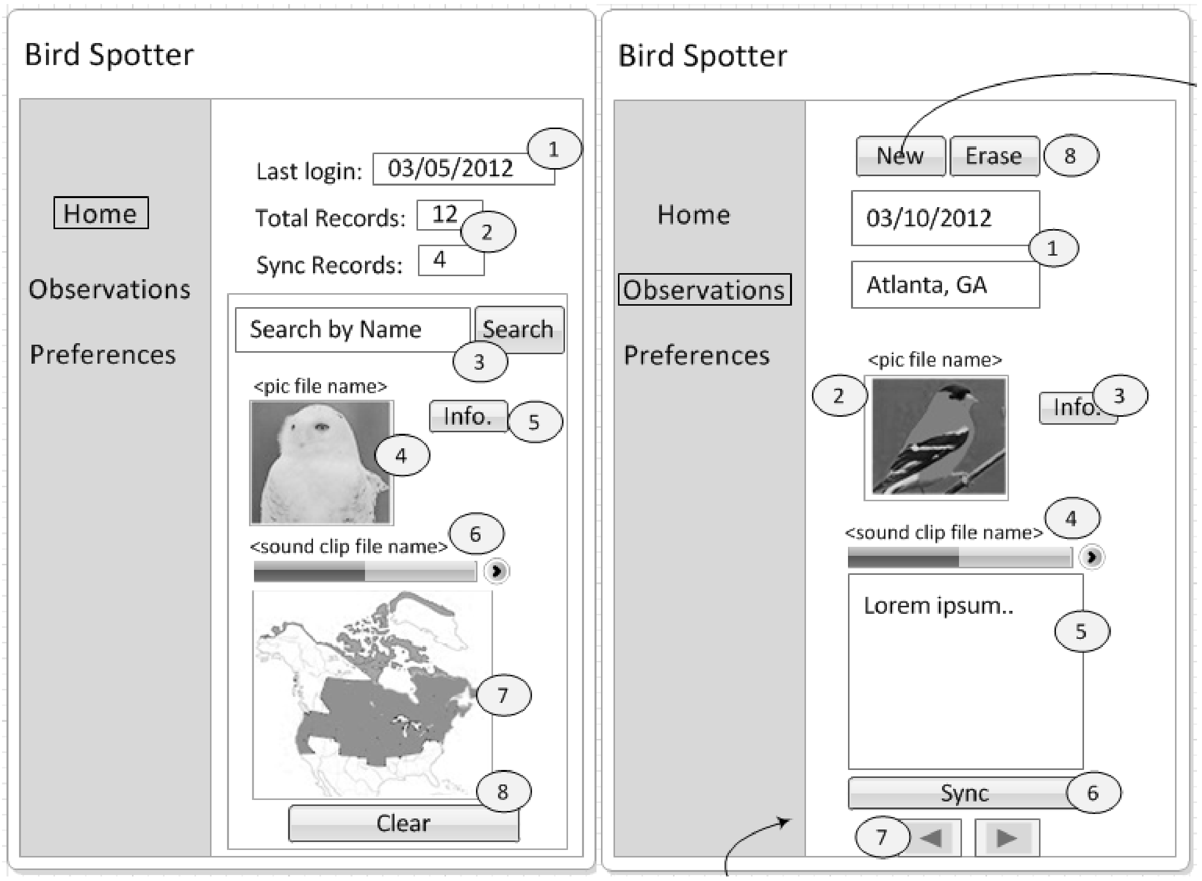

Figure 1: Mock-up of bird spotter application

Given that these users cannot be stationary and tethered, the mobile platform becomes the primary platform of choice. The user interface (UI) and layout design has to factor in such constraints as:

1. The smaller screen has limitations of real-estate for the controls
2. Consider the type of controls from an affordance perspective
3. Evaluate navigation and find-ability based on ease of use – kids, for e.g., would like to use and may not have very adept fingers or retention capability

The UI wireframe for the above user scenarios may look like that shown in figure 1. The UI mock-up is expected to support the user behavior stemming from user research. Combined with the user tasks, these navigations and interactions result in exploratory test flows. As the design iterates to the optimal, the tester can understand the user motivations to drive the testing strategy. The layout and structure of content corresponds to user's task goals and can be validated using acceptance tests. The tester seeks to identify gaps in the tasks specific user roles perform that are missed and tries to relate the user story to the navigation flows that they map to. For instance, a set of administrative tasks to *manage* the amount of content that can be downloaded to the mobile device given its constraints for space was a gap that was identified by exploration.

Another example of the detecting issues from exploration was in the "home" screen. The tester found that the interface did not support the amateur user's need to "browse" through the database of birds to match the observed features such as color; habitat and bird calls to identify a certain species that was spotted.

As the tester begins to understand the context of use, she uncovers specific navigation flows and system behavior resulting in undocumented application states and test conditions. The user characteristics such as domain knowledge also give insights into the way the application would be used. In the case of the amateur bird spotter she is not expected to know the species by their attributes or zoological name. The amateur needs to be able to look up birds through the browse feature that provides information about the species from the database. So the acceptance criteria might be the expectation of being presented with such collateral information (details on habitat and seasonal migratory patterns) that increases the user's confidence in identification.

The navigation flows around ease of finding this information forms the basis of usability testing but it is not as exhaustive as the opportunity presented to the exploratory tester to vary the test steps. The exploratory tester not only relies on feedback but also on such intangibles as experience and intuition. And so, the tester might want to test "what-if" and edge-scenarios around the flow being discussed here. The tester also relies on deductive reasoning from the system behavior she observes in using the system within the confines of the usability model described by the UX design artifacts.

The more experienced spotter, on the other hand, knows the bird species well enough that she will be searching for them not just by name but also by other characteristics. The application should be able to support these two sets of search-and-browse capability. For these specific and different types of user goals, the tester can explore ways to accomplish the task of locating specific birds in the application by changing the application states that affect the searchable set. These search factors are be affected by varying geographic locale and vicinity of the observer. The tester vectors tests by varying these controls from intuition and system response.

The research student in contrast is merely using the statistical capability for reporting her observations collected from the field. She is looking to get a quick record of the observation without having to browse through or search for the type of species spotted. The focus of this exploratory tester impersonating this persona is around the ease of reporting and being able to "tag" her observations appropriately for statistical compilation later. Though the system behavior forms the basis of exploration, the context and goal of ease-of-use testing would not be recognized without the details surrounding the persona.

Given the different ways of using the application, the UX designers provide guidance for such capabilities as configuring the user preferences. This may be in the form of

A. Specifying a location where the mobile user moves to provide geographically relevant bird collection or
B. Filtering instant messages from other spotters in the network by radial distance and ranking those by list of "friends" in the etc.

Given the concern about privacy and security, user research would also uncover the user's need to be able to grant or deny friends' invitation to share instant messages. The tester then sets out to test these possible different behavior permutations of the system. The ability to test these conditions using scripted tests will not be effective no matter how well the documented requirements are. A combination of structured automated and unscripted exploratory testing approach will likely uncover more potential issues. In addition, the cost of creating documented test assets for increasingly complex systems become prohibitively expensive. Besides, the value these scripted tests provide is little to none when it comes to detecting bugs efficiently. The cost of maintenance of these tests due to changes from big-design-up-front syndrome (BDUF) also increases.

Repeatedly running the same tests to uncover new defects is ineffective just like walking on the footsteps of another in a minefield which is unlikely to set off any explosions. The spirit of exploration in uncovering new bugs and potential defects is therefore very compelling for reducing inefficiencies in the testing process.

Another gap uncovered by exploratory testing is a scenario when a novice user moved to another geographic location and the application did not trigger update to the list of birds that are common in the new location. The documented requirements only called out the ability of the user to specify the location by using the preferences screen. It is only when the scenarios were reviewed based on tester experience and the application was already in use that the need for prompting the user to change the location was uncovered as an issue.

5. Conclusion

A key component of UX design is user research that results in segmentation of users into user personas. Personas are very useful in the discovery and understanding of the way the system would be used. Quality by design focuses on this aspect of user behavior illustrated by scenarios in the context of their use.

Scenario-based tests that are adaptive to different aspects of system behavior are more effective than any combination of scripted tests planned *a priori*. The personas and user scenarios provide the framework for this form of test adaptation to be contextual and relevant. Providing visibility to the test results from the perspective of user acceptance helps the team evaluate whether the development has achieved the minimally shippable feature set objectively.

For sprint planning and deriving test strategies, the lessons learned from exploration helps not only to develop test charters but also to adapt tests based on feedback. Overlaying the contextual information to assess the impact of defects to specific personas helps to evaluate the quality of the system from a fitness-for-use perspective.

In agile software engineering projects, the testing team can collaborate with the UX design team early in the development phase to converge to the goal of fitness-for-use efficiently. The solution team thus builds-in quality and improves testing efficiency. With pair programming as described above, the testing team also gets to engage early and test often. Adapting tests based on the feedback from system behavior is more realistic when personas are the basis for adaptation. Testing within the context of use in this manner makes the solution more successful.

References

1. Juha Itkonen, Mika V. Mantyla and Casper Lassenius, *Defect Detection Efficiency: Test Case Based vs. Exploratory Testing*, First Intl. Symposium on Empirical software Eng. and Meas. IEEE Comp. Soc., 2007.
2. James Bach, *To repeat tests or not to repeat,* http://www.satisfice.com/blog/archives/24.
3. Adam Goucher, *Quality Through Innovation*, http://adam.goucher.ca/?p=479.
4. James Bach, *Session Based Test Management,* http://www.satisfice.com/sbtm/.
5. Lisa Crispin and Janet Gregory, *Agile Testing: A Practical Guide for Testers and Agile Teams*, Addison-Wesley Professional, 2009.
6. Jakob Nielsen and Rolf Molich, *Heuristic evaluation of user interfaces*, CHI '90 Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people, pp. 249 – 256.
7. Jonathon Kohl, *Demystifying Exploratory Testing*, http://www.stickyminds.com/BetterSoftware/magazine.asp?fn=cifea&ac=439.