On the Importance of Supporting Multiple Stakeholders Points of View for the Testing of Interactive Systems

Alexandre Canny[1], Elodie Bouzekri[1], Célia Martinie[1], Philippe Palanque[1]

[1] ICS-IRIT, University Paul Sabatier – Toulouse III
firstname.lastname@irit.fr

Abstract. Testing is the activity meant to demonstrate that systems are fit for purpose and to detect their defects. On interactive systems, checking the fitness for purpose requires proper knowledge of the users' activities and profiles as well as of the context of use. Moreover, defects may be present in software, input/output device hardware or in the way interaction techniques are handled. Comprehensively testing interactive systems thus requires a large set of skills provided by usability experts, software engineers, human-factor specialists, etc. So far, these stakeholders conduct testing activities using processes from their respective areas of expertise that do not take advantage of others stakeholders' expertise effectively. This paper discusses the contribution of each stakeholders in current testing activities and highlights that a common view of the interactive system under test can serve as a mediating tool for each stakeholder to share information and identify/execute more relevant test suites.

Keywords: Interactive-System Testing, Stakeholders in Testing, Testing Activities.

1 Introduction

The testing of interactive system is known to be a complex activity that cannot be exhaustive [4]. Indeed, testing requires finding the system's defects and demonstrating it is fit for purposes [9], which is made difficult by the nature of interactive systems that integrates hardware, software and humans. On such systems, defects may be found in the code of applications as well as in the way the input/output devices and interaction techniques are handled in changing context (e.g. when an aircraft enters an area of turbulences), etc. Moreover, demonstrating that interactive systems are fit for purpose requires the ability to demonstrate that they let the users accomplish their goals and also that they are compliant with domain-specific constraints (e.g. is a videogame matching constraints imposed by rating organizations such as ESRB and PEGI?).

Researchers and practitioners in fields such as Software Engineering and Human-Computer Interaction developed processes and tools for supporting the testing of interactive systems using coverage criterions relevant in their respective areas of expertise. Furthermore, authorities and rating organizations introduced documentations geared towards systems manufacturers to let them know how fitness to purpose is

checked for domain-specific aspects. Unfortunately, testing remains conducted by stakeholders focusing on their own areas of expertise who are not working in close collaboration with stakeholders from other areas. This may lead, for instance, to software engineers making some assumptions on the way the user will interact with the application. By doing so, they may design test cases/suites that do not properly take into account the human capabilities when searching for defects (e.g. the SteamVR motion tracking system was not tested with expert players in mind [5]) even though some exchanges with usability experts could have help identifying correct ones. We claim that by allowing the various stakeholder in the testing activities of an interactive system to collaborate, designing relevant test cases would be easier.

In this paper, we first present the stakeholders in generic process for testing usability and software. Then, we present the stakeholders in the testing and validation of three kinds of interactive systems. The third section discusses the testing problem with architect view in mind and highlights how integrated the testing of interactive system should be. The fourth section highlights the need for exchange of information between stakeholders and for associated processes. The fifth section concludes the paper.

2 Process View on the Testing of Interactive Systems

In the fields of HCI and of Software Engineering, the testing activities have different objectives and are thus organized by different processes.

2.1 Testing in HCI

In the field of HCI, testing is associated to user evaluation, which aims at ensuring that the interactive system fulfills user needs in terms of usability, user experience and learnability. A good level of usability is always required because users have to be able to accomplish their tasks in an efficient way [11]. User testing takes place at various stages of the design and development process. The alternation of prototyping and user testing phases aims to capture the maximum of user needs and to ensure that user tasks and user behavior are compatible with the interactive system presentation and behavior. The usability design process (**Fig. 1**) [7] presents a set of steps that aims at developing an usable interactive system.

The main characteristics of the usability design process (see **Fig. 1**) are: an early user involvement, an iterative and incremental set of design steps, empirical measurements, evaluation of the use in context and multi-disciplinary design teams. Users are involved since the beginning of the design process and are then regularly solicited for the evaluation of mock ups and for the testing of prototypes. Several stakeholders thus contribute to the testing activities:

- **Users**: formulate their needs, accomplish given actions with the prototypes and give their opinion on the prototypes in terms of the perceived usability,

- **Designer**: gather user needs and produce mock ups and prototypes, ensure that the mock ups and prototypes are legible and functional for user review and user testing,
- **Programmers**: program high-fidelity prototypes and/or deploy the interactive system, ensure that the prototypes are reliable to be tested and used by users,
- **Usability experts**: observe and interview users, produce experimental evaluation protocols, manage the experiments and analyze the results.

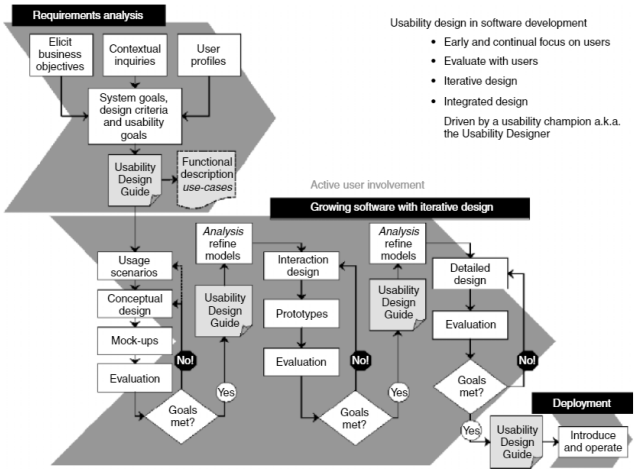

Fig. 1. The usability design process (from [7])

2.2 Testing in Software Engineering

In the field of software engineering, testing "consists of the dynamic verification that a program provides expected behaviors on a finite set of test cases, suitably selected from the usually infinite execution domain" [8]. During the software development process, several types of testing activities aim at ensuring that the produced software behaves as specified and is free of defects. **Fig. 2** depicts the ordering of the development and testing phases in the V software development process [2].

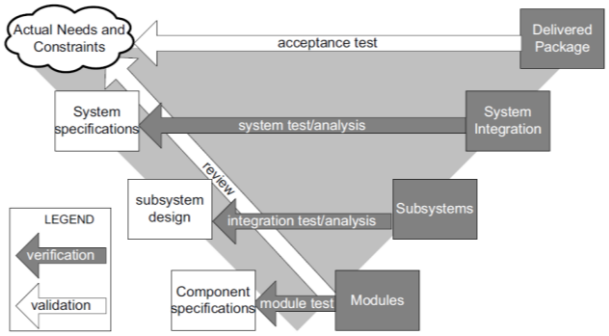

Fig. 2. Development and testing phases in the "V" development process (from [2])

Fig. 2 illustrates the different types of testing activities required to verify software systems. These activities are defined in the Software Engineering Body of Knowledge (SWEBOK) [8]. "Module test" (in **Fig. 2**) or unit test refers to the independent testing of each function and procedure. Integration test refers to the testing of several parts of the software that interact together. System test refers to the testing of the entire software. Acceptance test or validation test refers to the testing of the entire software in the context of use.

The testing of a software application involves different stakeholders [8] such as:

- **Software engineers**: produce specifications of requirements, specifications of (high level) software design and specification of system tests and integration tests. They also integrate the software components, perform the integration tests and build the entire software.
- **Programmers**: produce component (low level) software specifications, program the components and perform the unit tests for the components they have produced.
- **Testers**: execute the system tests, produce test reports and raise defects in case there are.

3 Application Domain View on Interactive System Testing

Beyond the generic nature of the processes presented in the previous section lies application domain-specific constraints and uses that may deeply influence the way to conduct the testing activities. Testing the compliance with regulatory obligations or guidelines are amongst the activity that may cause the involvement of specific stakeholders in the testing of an interactive system. In this section, we present some stakeholders involved in the testing of i) desktop application with GUI, ii) videogames and iii) safety critical systems.

3.1 Testing of GUI-Based Applications

Graphical User Interfaces (GUI) are known to be impossible to test exhaustively as the number of sequences of events that can be performed on their widgets is infinite [3]. Thus, the main challenge in testing GUIs is to identify the relevant event sequences to execute on GUI widgets [10]. Indeed, Banerjee et al. [3] define GUI testing as "solely by performing sequences of events (e.g. "click on button", "enter text", "open menu") on GUI widgets (e.g. "button", "text-field", "pull-down menu")". Banerjee et al. [3] present several types of GUI testing techniques (script-based testing, capture/replay testing and model-based testing). For each technique, different stakeholders are involved:

- **Programmer**: program the GUI application. In script-based testing approaches, the programmer additionally writes scripts describing the event sequence to execute and the expected state of the GUI either between each events or after the complete sequence.

- **Users**: accomplish given actions with the GUI applications. In capture/replay testing approaches, the **users'** interaction with the application are recorded. They are then used later for non-regression testing.
- **Software engineers**: execute the tests. The capture/replay approach allows to record relevant sequences (the ones that users actually performs), its main drawback is that these recordings become outdated as soon as a GUI element changes (e.g. while adding a tab in a settings window).
- **Test automation managers** and **test automation engineers**: are involved for model-based testing approaches. They select and apply techniques to build models of the GUI behavior from the results of the reverse engineering of the application [10] or from the requirements and specifications of the application [15]. They build models describing all the executable event sequences of up to a given length (as selected by test automation manager). The models are then used to generate relevant event sequences (e.g. all the sequences leading to the "Save as" dialog).

Besides the event-driven nature of the GUI, some organizations may want to verify that their GUI complies with specific guidelines such as accessibility one (e.g. [16]). While the automation of some of these tests is possible, usability experts may be involved in this process.

3.2 Testing of Games

Testing of games shares quality concerns with software applications. However, for the development of games, there is a common agreement in the community that successful games rely on an iterative development approach. Usability evaluation is an important aspect in games development: if a game is not usable (e.g. the interaction technology does not allow easy learning how to play the game), a game is typically not successful. Novak [12] makes a distinction between testing activities and quality assurance activities in game development. The game testing activities focus on the usability and user experience of the game. Whereas, the quality assurance activities include process monitoring, game evaluation and auditing according to the developer and publisher standards. Novak [12] identifies the following stakeholders involved in the testing of games:

- **Unit testing manager** is the responsible of the testing of multiple game projects.
- **Lead tester** is the testing team supervisor and manager. In addition, the lead tester must identify some types of errors (i.e. modeling or texturing errors)
- **Compatibility and format testers** work for a publisher. They focus on the cross-platform game compatibility.
- **Production (developer), quality assurance (publisher), regression testers** usually work together. They make suggestions to improve, to add or delete game features. They take into account prospective competing titles. Regression testers focus on severe bugs.

- **Playability, usability and beta testers** are involved during the Beta phase. The Beta Testers are volunteers who test the game in-house. They are members of game's target users.
- **Focus testers** are target users who test the game with the marketing department. These tests are similar to focus group [14].

Rating organizations (e.g. PEGI, ESRB, etc.) are also part of the validation process of a game. They are responsible for rating the game prior to their release and intervene at pre-production stage to attribute provisional ratings (found in game trailers), during the main production to adjust the rating to the game changes and in post-production to deal, for instance, with the rating of additional game content.

3.3 Testing of Safety Critical Systems

In safety critical systems, several quality factors deeply influence the development process such as reliability, fault-tolerance or security. The nature and high cost related to the evaluation of critical systems makes it necessary to test the whole system before its deployment, contrary to non-critical systems that can be patched. This constraint leads to plan certification very early in the development process of the system. To do so, the certification authority and the applicant commit an agreement as soon as a new project enters an active development phase. Then, each part of the system is tested and revised until it matches the certification requirements. In order to make the testing activities dependable, the principles of fault tolerance as detailed in [6] can be applied to the testing activity. For instance, assigning people of different organizations to the development and to the system testing covers the diversity and segregation principles. Hereinafter, we present the stakeholders involved in the testing and validation process of an aircraft, as listed by the Federal Aviation Administration [1] certification authority:

- **FAA (certification authority)**: authority supplies requirements (regulation and policy) and associated means of compliance to the applicant, determine conformity and airworthiness.
- **Applicant's inspectors** and **designees**: must demonstrate the compliance of the system to be certified with these requirements.
- **Applicant's flight test pilots**: conduct flight tests to show compliance.
- **FAA (certification authority) aircraft evaluation group** and **designees** evaluate conformance to operations and maintenance requirements.

Because safety critical systems are large-scale systems with multiple components, the testing process needs some automation. Moreover, the applicant's engineers may use formal model-based methods to model the system with automated property checking by using model-checkers as described in the DO-178C supplement 333 [13]. To avoid unnecessary tests, if a part of the already certified system is reused and unchanged in a new system to be certified, the already certified system part does not need to be tested.

4 Architectural View on Interactive-System Testing

In the previous sections, we highlighted that testers work with various considerations in mind. Thus, a key to support multiple stakeholder points of view in the testing of an interactive system is to benefit from a mediating view that bridges the gaps between those considerations. As architectures are meant to describe the conceptual structure and logical organization of a system, they are prime candidate to serve as mediating tools. While most architectures are domain specific (i.e. network architecture, software architecture, etc.), the H-MIODMIT architecture [4] (**Fig. 3**) highlights the presence of the human (left of **Fig. 3**) and the software (right part of **Fig. 3**). Moreover, this architecture considers hardware by explicitly mentioning "Input Devices" and "Output Devices".

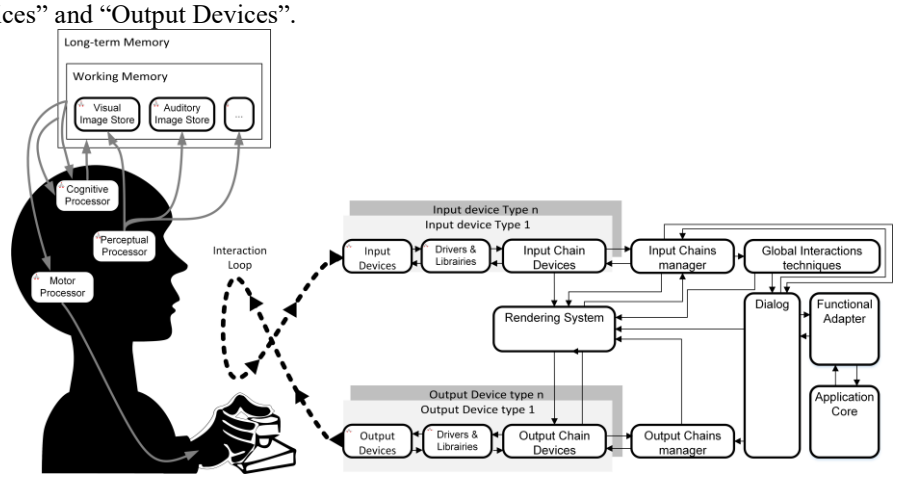

Fig. 3. The H-MIODMIT architecture (from [4])

Thanks to such architecture, it is possible to reason at a higher level of abstraction than with any domain-specific architecture. Thus, a usability expert (bringing knowledge about the human capabilities) may state that "a properly motivated human using a light enough controller can turn their wrist at up to 3600 degrees/sec" in a Virtual Reality experience [5]. Looking at this statement over the H-MIODMIT architecture, we identify that it relies on the knowledge of the "Motor Processor" (leftmost component in **Fig. 3**) and serves as an input knowledge for the testing of the "Input Devices" (i.e. the controllers motion sensors must be capable of handling rotation speed of up to 3600 degrees/sec). Moreover, this means that "Drivers and Libraries" must be able to produce relevant high-level events from the controller data (e.g. considering the way the controller samples information, is a "byte" sufficient to convey the delta angle?). Ultimately, such Usability Expert statement will translates into test specifications for components throughout H-MIODMIT. This architecture remains however insufficient to distribute all the testing requirements as it does not highlights, for instance, the existence of the context of use and its impact on the various systems' components.

5 Conclusion

Designing reliable and usable interactive systems is complex and involves multiple stakeholders. This position paper presents some of the stakeholders involved in interactive system testing. It highlights that the stakeholders from different areas of expertise may benefit from the knowledge of each other during the testing activities. This backs our claim that processes and tools supporting multiple stakeholders' points of view in the testing of interactive systems are required. Such processes and tools should provide ways for each stakeholders to understand the high-level test requirements defined in other areas of expertise and ways to trace-back them from refined requirements to propagate changes if the architecture or purpose of the interactive system evolves. Furthermore, they should be able to cope with application domain-specific requirements to design as comprehensive as possible test suites.

References

1. AIA, AEA, GAMA, and the FAA Aircraft Certification Service and Flight Standards Services: The FAA and Industry Guide to Product Certification (Third Edition) (2017).
2. Baresi, L., Pezzè, M.: An Introduction to Software Testing. Electronic Notes in Theoretical Computer Science. 148, 89–111 (2006). https://doi.org/10.1016/j.entcs.2005.12.014.
3. Banerjee, I., Nguyen, B., Garousi, V., Memon, A.M.: Graphical user interface (GUI) testing: Systematic mapping and repository. Information and Software Technology. 55, 1679–1694 (2013). https://doi.org/10.1016/j.infsof.2013.03.004.
4. Canny, A., Bouzekri, E., Martinie, C., Palanque, P.: Rationalizing the Need of Architecture-Driven Testing of Interactive Systems. In: Human-Centered and Error-Resilient Systems Development. Springer, Cham (2018).
5. Dent, S.: "Beat Saber" players were so fast that they broke Steam VR. https://www.engadget.com/2019/02/12/beat-saber-players-too-fast-for-steam-vr/
6. Fayollas, C., Martinie, C., Navarre, D., Palanque, P., Fahssi, R.: Fault-Tolerant User Interfaces for Critical Systems: Duplication, Redundancy and Diversity as New Dimensions of Distributed User Interfaces. Presented at the Proceedings of the 2014 Workshop on Distributed User Interfaces and Multimodal Interaction January 7 (2014). https://doi.org/10.1145/2677356.2677662.
7. Göransson, B., Gulliksen, J., Boivie, I.: The usability design process – integrating user-centered systems design in the software development process. Software Process: Improvement and Practice. 8, 111–131 (2003). https://doi.org/10.1002/spip.174.
8. IEEE Computer Society, Bourque, P., Fairley, R.E.: Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0. IEEE Computer Society Press, Los Alamitos, CA, USA (2014).
9. International Software Testing Qualification Board: ISTQB Glossary, https://glossary.istqb.org/search/.
10. Nguyen, B.N., Robbins, B., Banerjee, I., Memon, A.: GUITAR: an innovative tool for automated testing of GUI-driven software. Autom Softw Eng. 21, 65–105 (2014). https://doi.org/10.1007/s10515-013-0128-9.
11. Nielsen, J.: Usability Engineering. Elsevier (1994).
12. Novak, J.: Game Development Essentials: An Introduction. Cengage Learning (2011).

13. RTCA. DO-178C Software Considerations in Airborne Systems and Equipment Certification. (2011).
14. Stewart, D.W., Shamdasani, P.N.: Focus Groups: Theory and Practice. SAGE Publications (2014).
15. Utting, M., Pretschner, A., Legeard, B.: A taxonomy of model-based testing approaches. Softw. Test. Verif. Reliab. 22, 297–312 (2012). https://doi.org/10.1002/stvr.456.
16. W3C Web Accessibility Initiative: Web Content Accessibility Guidelines (WCAG) Overview, https://www.w3.org/WAI/standards-guidelines/wcag/.