International Journal of Software Engineering
and Knowledge Engineering
Vol. 25, Nos. 9 & 10 (2015) 1771–1775

DOI: 10.1142/S0218194015710151

User Scenarios Through User Interaction Diagrams

Douglas Hiura Longo and Patricia Vilain

Informatics and Statistics Department
Federal University of Santa Catarina, Florianópolis
Santa Catarina, Brazil
douglashiura@posgrad.ufsc.br
patricia.vilain@ufsc.br
http://www.uid.inf.ufsc.br

This paper investigates the applicability of User Interaction Diagrams (UIDs) as user scenarios for the specification of software requirements by non-technical customers. Two methods for building user scenarios using UIDs were proposed: the progressive and the regressive methods. These two methods were applied in an experiment where the results demonstrated that the regressive method requires significantly less effort as compared to the progressive method. Furthermore, there was a significant difference in the quality of diagrams obtained from each of the two methods. In our experiment, the regressive method resulted in better quality factors.

Keywords: Requirements engineering; UID; User Scenarios; ATDD; Assert First; TDD.

1. Introduction

Unlike traditional methodologies, Acceptance Test Driven Development (ATDD) is characterized by acceptance tests being developed since the beginning of the development lifecycle [1]. In ATDD, these tests consist in documenting software requirements and customer expectations in a format that can be automatically and repeatedly tested [2]. Thus, test automation requires domain specific languages for representation of the requirements as tests and tools to create and run the tests. However, the tools for test automation are not specialized enough to have end users participating in the requirements specification process [3]. Also, there is no evidence that a non-technical customer would be able to express requirements in domain specific languages [4].

User Interaction Diagrams (UIDs) were proposed to document application requirements by focusing on the interaction between the user and the system [5]. The present study investigates the possibility of a non-technical customer to express system requirements by utilizing UIDs as user scenarios, which can later be turned

into acceptance tests. This study also investigates the creation of requirements specification using the proposed progressive and regressive methods.

To evaluate the proposal, it was considered an experiment with 21 non-technical participants, and the requirements specification of a game. The objective of the experiment is to demonstrate the use of UIDs as user scenarios to specify requirements in the agile development, and compare the outcomes from applying both the progressive and regressive methods [6].

2. Research Proposal

This study uses UIDs for representing software requirements. Each UID comprises a set of symbols (circles, squares, arrows) and information types. This proposal replaces the information types represented in UIDs by values of the user scenarios. Figure 1 shows two examples of an 8-Puzzle game and the representation of a User Scenario using UID (US-UID). Both examples address the *win* scenario of the 8-Puzzle game using the same symbols, but with different methods of construction. In both examples, the initial state is random; in the second state the user enters the values (->) to move the empty space; and the last state of the system displays the result.

Table 1 shows the symbols used in US-UIDs.

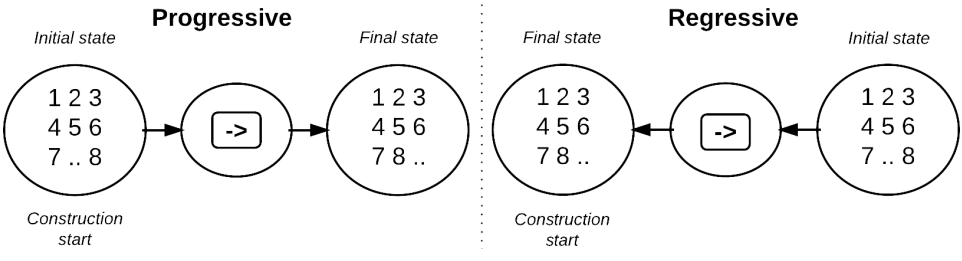

Fig. 1. Example of US-UID of 8-Puzzle game.

Table 1. Symbols for US-UIDs.

Symbol	Use
	**Ellipse** – it represents a **state of interaction**.
	Arrowed line – it represents the transition between interaction states.
	**Rectangle** – it represents the value of a **user input**.
Character sequence	**Value** – it represents the value of a **system output.**

For the representation of the requirements specification as user scenarios, the following methods are presented:

- *Progressive*: It indicates the expected result only at the end of the construction. Initially, the interaction states are built in order to achieve the expected result. Figure 1 (left) shows the progressive direction of the construction of a requirement specification, and its result is shown only at the end of the flow.
- *Regressive*: Similarly to the Assert-First technique [7], the regressive method starts the construction of the user scenario from the result, and adds other interaction states specifically to reach the outcome (initial state). Figure 1 (right) shows a scenario where the requirement is constructed with the regressive method.

3. Assessment of the Proposal

To assess the applicability and usefulness of the proposal in relation to completeness and consistency of requirements represented by 21 non-technical participants, we conducted an experiment. The experiment aimed at the construction of US-UIDs of the 8-Puzzle game. The following questions are investigated:

- RQ1: What quality factors (completeness and correctness) of the requirements are represented in US-UIDs?
- RQ2: Are such quality factors (completeness and correctness) associated with progressive or regressive methods?
- RQ3: Which of the proposed methods facilitates the construction of US-UIDs?

Regarding the overall assessment findings, the participants delivered 67% complete user scenarios where 90% of them used UIDs correctly (RQ1). A user scenario was considered complete when it presented the end result or state of victory, and at least one user input to the board configuration; and considered correct when the UID symbols were properly applied. Figure 2 displays the outcomes of completeness, correctness and time spent divided according to the scenario method of construction.

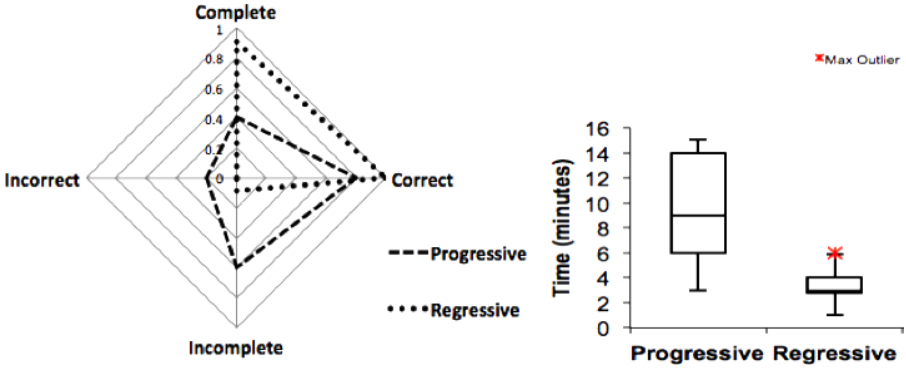

Fig. 2. Probability of correctness and completeness (left) and distribution of time spent (right), divided according to construction method.

The US-UIDs specified by the progressive method have a 40% probability of being complete, and 80% of being correct. The regressive method has 91% probability of being complete, and 100% of being correct (RQ1). The Fisher's statistical test obtained a p-value of 0.04928. Thus, for the confidence level $\alpha = 0.05$, $p\text{-}value < \alpha$, then the alternative hypothesis (H_1) is acceptable, allowing to conclude that the quality factors are different between the methods (RQ2). The time spent by the progressive group is between 3 and 15 minutes, and the regressive group is between 1 and 6 minutes. Thus, according to the time distribution analysis, the regressive method involves less effort (RQ3).

4. Threats to Validity

The main goal here is to determine whether and how participants can construct US-UIDs. Thus, in line with [8], the internal validity of this research relates in inferences made based on data from a specific experiment. The inferences are made following the use of appropriate measures and metrics for evaluation of US-UIDs. We specifically avoid the use of absolute measures of completeness and correctness as expressed in IEEE [9].

External validity refers to the ability to generalize the findings to other domains [8]. The external validity of this research contains two threats: the problem domain studied and the population of participants. The problem domain (8-puzzle) contains user interactions with the system. These user interactions are similar, such as: a calculator; authentication systems or web sites. Thus, the US-UIDs have a potential applicability to other areas of software engineering. The small population of participants is a weakness. However, it is possible to reply the methodology to increase the data [6].

5. Conclusions

In this study, UIDs were utilized to allow non-technical customers to represent user scenarios. The applicability of user scenarios (US-UIDs) is related to agile software development, where requirements are customer expectations and can also be used as acceptance tests. In this context, UIDs were quite suitable for creating user scenarios that specify software requirements. Moreover, in our experiment, the proposed regressive method based on the TDD assert-first technique resulted in the reduction of effort, and improvement in the assessed quality factors of the requirements.

References

1. L. F. S. Hoffmann, L. E. G. Vasconcelos, E. Lamas, A. M. da Cunha and L. A. Vieira Dias, Applying acceptance test driven development to a problem based learning academic real-time system, in *11th International Conference on Information Technology: New Generations*, 2014, pp. 3–8.

2. B. Haugset and G. K. Hanssen, Automated acceptance testing: A literature review and an industrial case study, in *AGILE'08 Conference*, 2008, pp. 27–38.
3. M. Kamalrudin, S. Sidek, M. Nor Aiza and M. Robinson, Automated acceptance testing tools evaluation in Agile software development, in *Sci. Int.* (Lahore), 2013, pp. 1053–1058.
4. K. Alvestad, Domain Specific Languages for Executables Specifications, in Institutt for datateknikk og informasjonsvitenskap. 2007, p. 63.
5. P. Vilain, D. Schwabe and C. Sieckenius, A diagrammatic tool for representing user interaction, in *UML 2000 — The Unified Modeling Language*, 2000, p. 133–147.
6. D. H. Longo and P. Vilain, Creating user scenarios through user interaction diagrams by non-technical customers, in *27th International Conference on Software Engineering and Knowledge Engineering*, 2015, pp. 330–335.
7. K. Beck, *Test Driven Development: By Example* (Addison-Wesley Professional, 2003).
8. R. K. Yin, *Case Study Research: Design and Methods*, 5th ed. (Sage, 2013).
9. Recommended Practice for Software Requirements Specifications, in IEEE Std 830-1998, 1998, pp. 1–40.