Executable Acceptance Tests for Communicating Business Requirements: Customer Perspective

Grigori Melnik
Department of Computer Science, University of Calgary
Calgary, Alberta, Canada
melnik@cpsc.ucalgary.ca

Frank Maurer
Department of Computer Science, University of Calgary
Calgary, Alberta, Canada
melnik@cpsc.ucalgary.ca

Mike Chiasson
Department of Management Science, Lancaster University
Bailrigg, Lancaster, UK
m.chiasson@lancaster.ac.uk

Abstract

Using an experimental method, we found that customers, partnered with an IT professional, are able to use executable acceptance test (storytest)-based specifications to communicate and validate functional business requirements. However, learnability and ease of use analysis indicates that an average customer may experience difficulties learning the technique. Several additional propositions are evaluated and usage observations made.

1. Introduction

Testing has a prominent role in most agile projects regardless of which agile method is adopted. Testing is no longer just a phase at the end of the development lifecycle done when the coding is completed. In the agile world, testing is seen as an all-encompassing activity performed at different levels by all stakeholders – programmers (unit testing), customers (acceptance testing), and testers (acceptance testing, exploratory testing, para-functional testing).

Acceptance test (which is known under several other synonyms, see Table 1) is a (formal) test conducted to determine whether or not a system satisfied its acceptance criteria and it enables the customer to determine whether or not to accept the system (as defined in [1] and [26]). The objective is to provide confidence that the delivered system meets the business requirements of the customer (and users).

The iterative nature of agile processes dictates automation of the acceptance test (i.e. producing "executable acceptance tests"[1]) as manual regression testing at the customer level is too time consuming to be practical. At the same time, making tests too formal may push the customer away from the process.

Furthermore, XP and Industrial XP advocate writing acceptance tests in the test-first/TDD fashion. As a result, "*executable acceptance test-driven development*", or "*story-test driven development*", makes it possible to formalize the expectation of the customer into an executable and readable contract that programmers follow in order to produce and finalize a working system [29]. This also establishes "a clear context for customers and developers to have a conversation and weed out misunderstandings." [27] As a result the risk of building the wrong system is reduced.

Executable acceptance tests can be accessed and run by anyone on the team. This includes the customer, who may be interested in seeing the progress of the development, or exercising some additional "what-if" scenarios to ensure that the system is working properly.

In agile circles, there is uncertainty about who should author acceptance tests. Since early days of XP, customers were responsible for specifying acceptance criteria for each user story (which may include calculation rules, workflow scenarios etc.). Some

Table 1. Acceptance Test Synonyms.

Term	Introduced by
– *"functional tests"*	Beck, Extreme Programming Explained, 1/e [6]
– *"customer tests"*	Jeffries [17], Beck, Extreme Programming Explained, 2/e
– *"customer-inspired tests"*	Beck, Extreme Programming Explained, 1/e [6]
– *"story-tests"* and *"story-test-driven development"*	Kerievsky [20]
– *"specification by example"*	Fowler [14]
– *"coaching tests"*	Marick [22]
– *"examples"*, *"business-facing example"*, and *"example-driven-development"*	Marick [21]
– *"conditions of satisfaction"*	Cohn [10]
– *"scenario tests"*	Kaner [18]
– *"soap opera tests"*	Buwalda [8]
– *"formal qualification tests"*	e.g. DOD [11]
– *"system tests"*	e.g. IEEE [15]

[1] In our research, we follow this terminology.

believe that "customers typically can't write functional [acceptance] tests by themselves. They need the help of someone who can first translate their test data into tests, and over time can create tools that let the customer write, run, and maintain their own tests" [6]. There are those who mandate acceptance tests to be, at a minimum, understandable by the customer, and in ideal cases to be written by them [4]. Most agree though that the customer should not do it alone, but, instead, pair up with a tester, a business analyst, a user experience designer, or a programmer.

Unfortunately, little is known about the capability of customers to specify acceptance tests (and what is known is usually in the form of the anecdotal evidence). Modern frameworks and tools (like FIT [12], FitNesse [13], TextTest [4]) are supposed to make it easy for the customer to specify, organize and execute acceptance tests. But is it enough? Christian Sepulveda, an XP coach, questions this common expectation of the agile teams for "the customer to magically have the ability to write acceptance tests. At least developers are trained in logical and cognitive activities; the customer may not be."[30].

Thus, the main question remains whether customers on agile projects are capable of and comfortable authoring acceptance tests.

The rest of the paper is structured as follows. It first states the research goals and opening questions. It continues by discussing the research design, including sampling, methodology, formulation of hypotheses, and procedure. Then it presents findings based on quantitative data and their interpretation. The practice of executable acceptance testing for communicating business requirements is further elaborated in a discussion of qualitative data and user perceptions. This is followed by a discussion of threats to validity and limitations of the study. We conclude with the questions that remain for future work.

2. Study goals and study questions

We are interested in the capability of executable acceptance tests to communicate and validate functional requirements. In our previous work [23], we examined the suitability of acceptance tests for specifying functional requirements from the developers' perspective, and identified usage patterns in [28]. Suitability analysis was based on the extended list of twelve requirement specification "sins" inspired by [24] and included: noise, silence, over-specification, wishful thinking, ambiguity, forward referencing, oversized documents, reader subjectivity, customer uncertainty, multiple representations, use of tools, and user involvement. It was concluded that executable acceptance tests effectively addresses these issues, with the exception of silence and use of tools. Moreover, our evidence directly supported the understandability of this form of functional requirements specification by developers, even if they had little background in the FIT framework which was used for writing executable acceptance tests. Over 90% of the teams delivered an implementation that fully satisfied the customer's acceptance test suite, and the learning curve for reading and implementing executable acceptance tests was not prohibitively steep. However, in the second part of the study, in which subjects themselves were responsible for describing additional requirements in the form of acceptance tests to a different team, the results were less optimistic. The acceptance tests that were written by the subjects were of limited quality. Several factors may have played a role in this (including a short time frame, lack of motivation, and other priorities). In the present study, we set out to investigate specifically this aspect of executable acceptance tests specifications and to ask whether customers are able to effectively author their functional requirements and business rules in the form of executable acceptance tests, and communicate them to the development teams.

One of the threats of external validity of the earlier studies was the use of software engineering undergraduate students to specify acceptance tests. Though some of them may be involved with the requirements specification process in the future, they served as a poor sample of the customer population. A better representation was needed. To address this problem, in this study, we tried to approximate business customers by including both business school graduate students and computer science graduate students as our customer representatives.

Our research questions pertain to both the customer team's capability and the substance of the acceptance tests produced, specifically:

Q1: Can customers specify functional business requirements in the form of executable acceptance tests clearly when paired with an IT professional?

Q2: How do customers use FIT for authoring business requirements?

Q3: What are the trends in customer-authored executable acceptance test-based specifications?

Q4: Does a software engineering background have an effect on the quality of the executable acceptance test-based specification?

Q5: Is executable acceptance test-driven development a satisfactory method for customers, based on their satisfaction, their intention on using it in the future, and their intention to recommend it to other colleagues?

3. Research design and methodology

3.1. Participants

Three groups of University of Calgary students were involved in the study (see Table 2):

- Business school graduate students (further denoted as "*Business-grads*") enrolled in a Master of Business Administration program, taking a course in e-business as one of their elective courses.
- Computer Science graduate students plus one Computer Engineering graduate student ("*Computer-grads*"), typically enrolled in their first year of a Master's degree program, and enrolled in the same course with the Business-grads. Most of them had prior experience in the software industry.
- Senior Computer Science and Computer Engineering undergraduate students ("*Computer-undergrads*") enrolled in a separate course from the other two groups, on enterprise Web-based systems.

Both the graduate and undergraduate courses ran during the same term (Fall 2005).

The graduate students (*Business-grads* and *Computer-grads*) formed customer teams and specified requirements for a Web-based project management system; while the undergrad students formed development teams who were responsible for implementing requirements specified by the customer teams. The system was deliberately chosen such that its requirements would be more accessible to the business students and not as apparent to the undergraduates.

Graduate students self-organized into teams of two, with only one constraint: only one Business-grad was allowed per team. As a result, given differences in business and computer science graduates, nine customer teams were formed, three of which were purely comprised of Computer-grads, while six were a mix of one Business-grad and one Computer-grad. Undergraduate students also self-organized into an equal number of development teams. The total number of teams involved in the research included 9 customer teams and 9 development teams. As a result, there were a total of 18 customer-subjects, and the total number of all participants (including development teams) was 40.

With the exception of one person, all members of the customer teams were mature students and had related industrial experience, with mode being "more than 5 years" and median being "3-5 years". Female/male ratio of subjects was: 1/5 for Business-grads, 4/8 for Computer-grads, and 2/20 for Computer-undergrads.

3.2. Method

A quasi-experiment [31][5] was used as a basis for our research design. The choice was motivated by the use of nonrandom sample (convenience sample of graduate students) and a small size (18 students, 9 teams, out of which one team was disqualified due to

Table 2. Sample, Programs, and Courses.

Abbreviation	Major	Course	URL	Instructor (s)	Role	% female	# participants	Team size
Business-grad	Management Information Systems	Enabling E-Business	http://ebe.cpsc.ucalgary.ca/ebe/Wiki.jsp?page=CPSC_601_11_MGIS_797_03_F2005	Author 2, Author 3	Customer	17%	6	2
Computer-grad	Computer Science/ Computer Engineering	Enabling E-Business	http://ebe.cpsc.ucalgary.ca/ebe/Wiki.jsp?page=CPSC_601_11_MGIS_797_03_F2005	Author 2, Author 3	Customer	33%	12	
Computer-undergrad	Computer Science/ Computer Engineering	Web-Based Systems	http://mase.cpsc.ucalgary.ca/seng513/F2005	Author 1	Developer	9%	22	2–3

their poor participation). The primary source of evidence about the sample's representation to the population of customers is that the majority of subjects had more than 5 years of industrial experience and all were trained in either Management Information Systems or Computer Science. The assignment of development teams to customer teams was also random.

In this research, we primarily sought trends rather than cause-effect relationships.

3.3 Hypotheses

Our central hypothesis was that "*Customers in partnership with an IT professional would be able to effectively specify functional requirements of the system in the form of executable acceptance tests*". We took "effectively" to mean that we would see evidence of "good" tests and thorough coverage of the major system functionality (which would result in a grade of 75% or higher). In order to be "good", acceptance tests had to satisfy the following criteria (these are based on [19]):

- credible (contain realistic and reasonable set of operations to be likely performed by the customer);
- appropriate complexity (involve many features, attributes, workflows, etc.);
- coverage of the major functionality for a Human Resources Intranet system (management of projects, subprojects, time sheets, time-sheet allocation to projects, reporting of cumulative project time, expense claims management; administrative staff features);
- easy to read and informative;
- easy to manage (packaged in meaningfully structured suites, subsuites etc.).

The role of "an IT professional" was performed by the Computer Science graduate students, most of whom had prior work experience in the software industry.

Additionally, we hypothesized that:

A) *Customers with no previous experience with executable acceptance testing or FIT will find it easy to learn how to use FIT given the time provided.* Learnability is to be determined based on individual perceptions (scored on the Likert-scale). Time was provided included to learn the FIT system in a three-hour in-class tutorial by a FIT expert and four-weeks of practice.

B) *Customers will specify predominantly positive test cases*. By "predominantly", we mean 90% or more of all test cases being positive, and 10% or less being negative (i.e. testing various error conditions).

C) *There exists a significant difference in the quality of the specifications produced by customer teams comprised of two members with Computer Science background, vs. customer teams comprised of one Business-graduate and one Computer-graduate.* In this case, the response variable was a subjective grade based on consensus evaluation by two instructors (blinded to the student composition in each group).

D) *The quality of the executable acceptance test specification is strongly and positively correlated with the quality of the implementation produced by the development team.* Correlation is to be determined by calculating Spearman's correlation coefficient. We use the following interpretation ranges: <.01 – none, .01-.03 – weak, .03-.05 – moderate, .05-.08 – strong, >.08 – very strong.

3.4 Procedure

A project was conceived by the instructors to develop a Human Resources Intranet system to manage projects, consultants' time sheets, and expense claims (see Figure 1). A one-page narrative was given to the customer teams to provide initial ideas about the type of the system to be developed. This was a high-level, generic outline of the vision of what the system was supposed to achieve. No particulars were given and customer teams were free to decide on the business constraints and rules of the system. It was critical to the experiment that the system chosen was more accessible to the business students and not as apparent to the undergraduates to eliminate a potential experience bias.

A three-hour tutorial on executable acceptance testing, the FIT framework and the FitNesse tool was offered to the customer teams. It was attended by all subjects. The tutorial demonstrated the use of FIT for specifying two types of business rules: a) transactions, workflows and processes (with DoFixture type tables); and b) decision tables and business calculations (with ColumnFixtures type tables). The use of RowFixture for business queries and reporting features was left for self-study using the FIT/FitNesse documentation

available online [12][13]. A case study was used to illustrate the framework during the tutorial. An expert on FIT (Author 1) was available for consultation both in person and via email.

Three of the subjects had prior experience with the framework. The rest of the customers had no experience with the authoring of executable acceptance tests in FIT or any other format. Table 3 depicts level of experience of subjects with other requirement specification techniques.

Table 3. Summary of Knowledge Levels of Customers' Experiences with Various Requirement Specification Techniques.

	Median	Mode	Min	Max
Narrative/prose	2	2	0	4
Use cases	2	3	0	4
User stories	1	1	0	4
Domain-specific languages	1	0	0	3
Scenarios	2	2	0	4
Stroyboards	1	1	0	4
Prototypes	2	3	0	4
Mind maps	0	0	0	4
Personas	0	0	0	4
System archeology	0	0	0	3
Executable acceptance tests/FIT	0	0	0	4
Other	0	0	0	0

Note: N=17, legend: 0 = unfamiliar, 1 - some knowledge, 2 - average knowledge, 3 - good knowledge, 4 - extensive knowledge

Customer teams in the graduate class were required to specify suites of executable acceptance tests so that the development teams in the undergraduate course could implement a system. To motivate exploration and to eliminate a fear of potentially damaging the test artifacts, customers were informed that all test suites were version-controlled and that it was possible to revert any changes they made at any time.

Development teams were given a one-paragraph mission statement (no more detailed than outlined in Figure 1). Detailed requirements were to be given in the form of executable acceptance tests by the customer teams. Development teams were instructed that their designs and implementations should be driven by those tests. If they were unclear about a requirement, they were encouraged to communicate directly with their customer by any means they deemed useful. Development teams were also warned about the potential change of the requirements and their responsibility was to adapt their code to that change if it occurred.

To allow communication between the customer and development teams, each customer-development team combo had their own dedicated password-protected Wiki-based virtual space that also contained all acceptance tests (9 instances of FitNesse servers were running on different ports of a centralized server machine).

Considering the time limitation of the customer team members (they could only be involved with the project – on a part time basis – for 4 weeks), this study only covers the first iteration of the project. During this iteration, development teams were required to design a content model for the artifacts to be used by the system (in the form of XML Schemas), to build the necessary XSLT sheets for performing queries and renderings on XML raw data as per customer requirements, and to implement the XML processing logic necessary to satisfy customer requirements.

Figure 1. Project Mission Statement.

> Deltoid Consulting is a consulting company, headquartered in Calgary, with over 120 IT and business consultants. They are hired by their major corporations in the development of large IT-enabled systems for streamlining business processes. Their engagements with clients take them all over North America. Ironically, Deltoid has trouble managing **time sheets** and **expense claims** for its large consulting staff, and requires an information system to support this task.

All test and coding artifacts were archived and analyzed after both courses were completed and final grades were assigned.

In addition, two questionnaires were administered (pre- and post-iteration). The objective of the pre-questionnaire was to collect data on the prior experience of the customer teams and their familiarity with various requirement specification techniques. The objective of the post-questionnaire was to gain a better understanding on how customer teams accomplished their task of specifying business requirements with executable acceptance tests and to gather qualitative feedback on various aspects of using the framework, various requirement "sins", communication with the development team, and their perceptions of the executable acceptance test-based specification technique.

Participation in this research was voluntary. Subjects were permitted to withdraw their data from the study (though nobody did). Knowledge of their participation remained anonymous until after the course grades were submitted.

4. Findings

4.1. Central hypothesis: Customers in partnership with an IT professional can effectively specify acceptance tests

Our central hypothesis was that customers would be able to effectively describe functional requirements of the system in the form of executable acceptance tests so that a development team could implement the features for those requirements. The evaluation of the quality of executable acceptance test specifications ("spec scores") produced by the customer teams are shown in Table 4. Notice, one team was excluded from the data analysis because both customers acknowledged their lack of participation in this project due to other heavy commitments and time constraints. Using a one-sample t-test, we wish to test whether the mean of spec score differs significantly from 75% (in accordance with our central hypothesis formulation). Specifically, we test: H_0: $\mu = 75.00$, H_A: $\mu > 75.00$. The results of this t-test are as follows: t=2.873, df=7, α=0.024, mean=91.56, mean difference=16.56, 95% CI of the difference = [2.93, 30.20]. The mean of the spec score variable for our sample of teams is 91.56, which is statistically significantly different from the test value of 75.00. We conclude that this supports our central hypothesis at the 5% significance level, i.e. the sampled group of teams has a significantly higher mean on the quality of executable acceptance test specifications than 75%.

Table 4. Evaluation of the Quality of Specification and the Quality of Implementation

Customer Team	Team Type	Spec score, /100	Code score, /100
1	Business+CompSci	100.0	94.0
2	Business+CompSci	75.0	67.0
3	Business+CompSci	57.5	71.0
4	Business+CompSci	100.0	79.0
5*	Business+CompSci	100.0	80.0
6	CompSci+CompSci	100.0	89.0
7	CompSci+CompSc	100.0	70.0
8	CompSci+CompSc	100.0	70.0
9**	*MGMT+CPSC*	*42.5*	*89.0**
	Mean	91.56	77.50
	SD	16.31	9.84
	Median	100.00	75.00

Notes: * Team 5 produced an exceptional suite of acceptance tests and was awarded 100.0+.
** Team 9 was excluded from the analysis due to lack of participation.

4.2. Learnability and ease-of-use of FIT and FitNesse

To remove prior knowledge bias, we have excluded from analysis three students who have characterized their level of knowledge of executable acceptance testing and FIT as "good" or "extensive". The total number of responses evaluated for this hypothesis was 14.

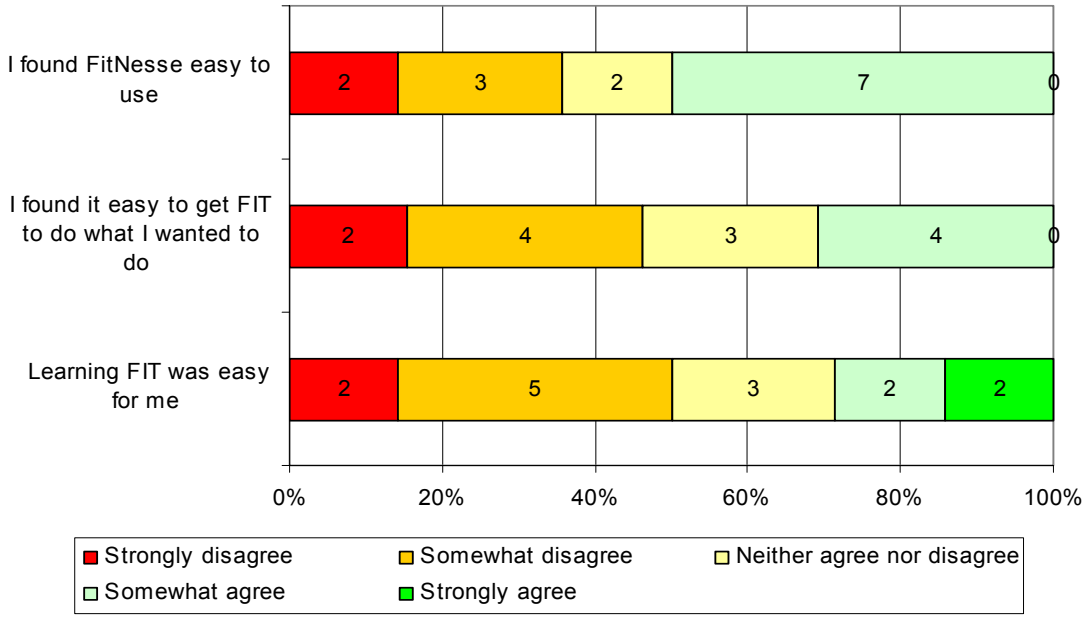
As can be seen from Figure 2, half of the customer team members found it hard to learn FIT, thus rejecting our Hypothesis A that customers with no previous experience with executable acceptance testing or FIT will find it easy to learn how to use FIT given the time provided. Furthermore, four subjects (29%) found FIT to be easy to use and seven subjects (50%) found FitNesse easy to use. This speaks to the usability aspect of FitNesse, which is based on a Wiki and follows very simple syntax rules. FitNesse also allows test specification, management, and execution though a consistent centralized Web interface. Notice, these evaluations are made by the customer teams only, and they do not address the issue of usability of FIT/FitNesse from the developer's perspective (i.e. those responsible for writing the "glue" in the form of fixtures).

Spearman's rho calculation shows a statistically significant correlation (rho_s=0.549, p=0.042, N=14) between the FIT learnability and the program subjects were enrolled in. As expected, Computer-grads found it easier to learn FIT than Business-grads. Furthermore, a statistically significant strong positive correlation between learnability and FIT ease-of-use was found at p=0.001. In other words, those customers who thought it was easy to learn FIT, found it also easy to use. We arrived to similar conclusions with regard to the ease of use of FitNesse.

Further analysis of correlations between FIT learnability, FIT ease-of-use, and FitNesse ease-of-use with prior work experience, presents no significant evidence of such relationships.

To sum up, based on the results with highly trained Business-grads and Computer-grads, it seems that an average customer representative may experience similar difficulties in learning the FIT framework. However, once the learning curve has been surpassed, subjects find both FIT and FitNesse easy to use and they produce good-quality specifications (as Table 4 shows).

Figure 2. Learnability and Ease-of-Use (N=14)



4.3. Positive vs negative testing

Negative test cases identify how a system responds to incorrect or inappropriate information or action. Negative testing is performed to ensure that the system is able to handle inconsistent information. Negative acceptance tests (often expressed in the form of negative scenarios) are increasingly recognized as a powerful way of thinking about requirements, possible conflicts, and identifying threats [3] [2].

In our experiment, we hypothesized that positive tests (the normal flows of logic) would dominate. Indeed, negative tests (that would deal with deviance from the course of action or misuse of the system) accounted only for 6% of all tests (as can be evidenced from Table 5). Additionally, Table 5 contains descriptive statistics of the total number of test pages (those would be the test pages in FitNesse) and total number of test/test cases (of which there could be few on a single test page). A one-sample t-test for test value=80% renders the following results: t=3.366, df=7, α=0.012, mean=90.86, 95% CI of the difference = [3.23, 18.49]. This statistically supports our Hypothesis B that positive tests are prevalent in the executable acceptance test specification written by the customers.

Having examined the patterns of negative vs positive test cases among the pure Computer-grad customer teams and mixed teams (Computer-grad + Business-grad), no evidence was found that pure Computer-grad teams would produce larger number of negative tests; though one would expect that to be the case since Computer-grad students should be familiar with testing techniques and aware of the need for negative test cases.

Table 5. Test Page and Test Case Type Distributions.

Customer team	Total, Test Pages	# Negative tests	% Negative of Total	# Positive tests	% Positive of Total	Total tests
1	31	18	26%	52	74%	70
2	48	14	6%	220	94%	234
3	44	0	0%	199	100%	199
4	11	19	14%	121	86%	140
5	118	1	0%	490	100%	491
6	6	39	18%	181	82%	220
7	21	16	7%	220	93%	236
8	27	9	3%	279	97%	288
Mean	38		6%		94%	

4.4. All Computer-grad customers teams vs. mixed customer teams

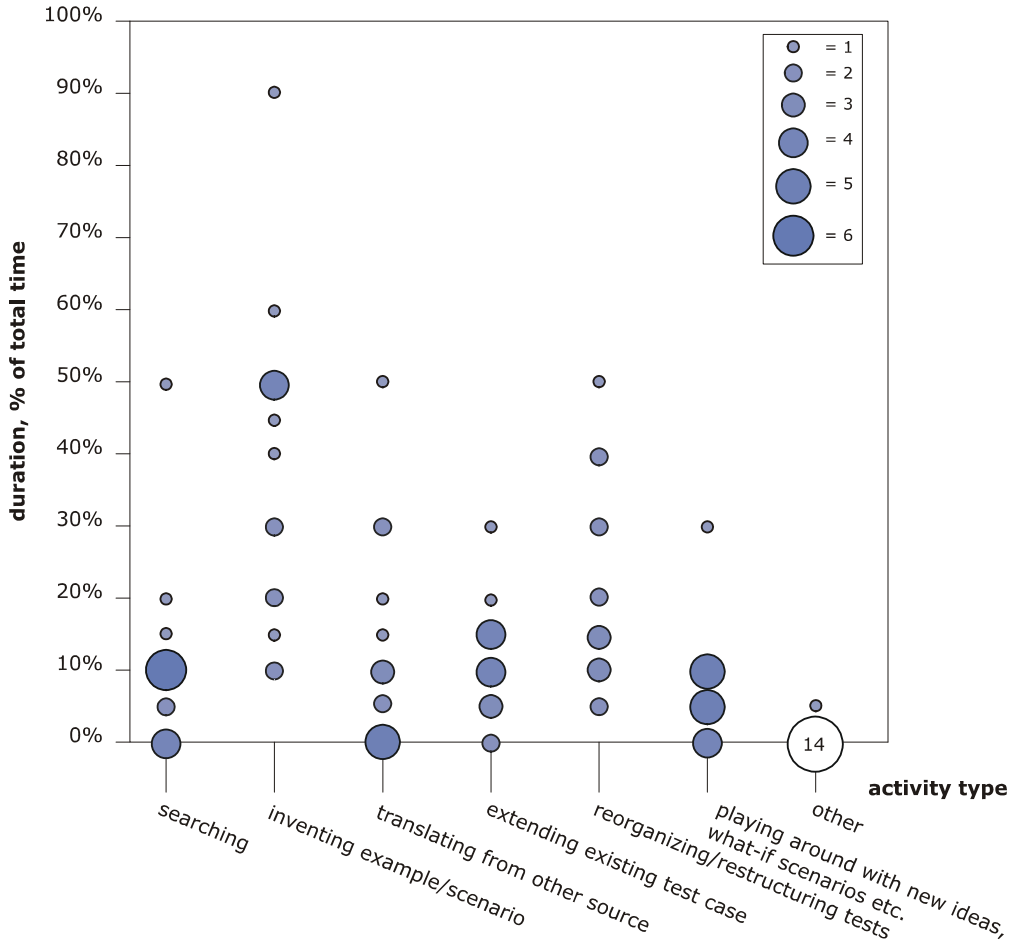
Recall, that according to the research design (Section 3.1), two types of customer teams were formed (see Table 3). In one type, only Computer-graduates were put together (there were 3 teams of this type). The other type mixed one Business-graduate with one Computer-graduate (6 teams). In order to

investigate whether there is a significant difference in quality of specifications produced by these treatment groups and bearing in mind that the normality of the data cannot be assumed, we resort to a non-parametric Mann-Whitney U/Wilcoxon Rank Sum right-tailed test. Specifically, the following hypotheses were tested: H_0: *specification quality of mixed team = specification quality of pure computer science team;* H_A: *specification quality of pure computer science team > specification quality of mixed team*. The result of the test (U=4.5) indicates no significant difference in the quality of the specification produced between two team types at the 5% significance level, and therefore does not support Hypothesis C. This is an interesting finding because it suggests that the customer teams we equal (perhaps because the technique bootstraps the Business group to the same level as the pure Computer Science students}. At a minimum, this deserves further investigation.

4.5. Correlation between the quality of acceptance test-based specification and the quality of implementation

In addition to the evaluation of the executable acceptance specifications created by the customer teams, the code produced by the development teams was subjectively evaluated by Author 1 and his teaching assistant (see Table 3). Since in our study, normality of data distribution cannot be assumed (the distribution of the specification quality has a kurtosis of 2.137 and skewness of -1.777; and the distribution of code quality has a kurtosis of -.810 and skewness of .751), a Spearman's correlation coefficient computation was used.

Figure 3. Activity Categories Duration Data.



Although there is a moderate and positive correlation between the quality of executable acceptance test specifications created by the customer team and the quality of implementation produced by the development team (rho_s = 0.455), the probability of this correlation occurring by chance is too high (p=.229). Therefore, no significant correlation between these two variables at the 5% significance level can be reported. As a result, Hypothesis D, that the quality of the executable acceptance test specification produced by the customer team will be strongly and positively correlated with the quality of the implementation produced by the development team, is not supported. Considering the importance of the quality of requirement specifications on the success of software development project (see [9]), this issue deserves further attention and experimentation by other researchers.

5. Additional observations.

Here we discuss additional observations for which no prior hypotheses were made, but which help in answering the research questions (outlined in Section 2).

5.1 Types of activities

Subjects were asked to self-assess the amount of time they spent on the following activities[2] as a portion of the total time devoted to the project:

- searching for information;
- inventing an example/scenario;
- translating substantial amounts of information from some other source into the system;
- adding small bits of information to a test that you have previously created;
- reorganizing and restructuring tests that you have previously created;
- playing around with the new ideas, what-if scenarios, without being sure what will result;
- other.

Figure 3 depicts distribution of activities by duration (position on the chart) and popularity (diameter of the bubble). It appears that most of the time is spent on inventing examples/scenarios/test cases (from 10% to 90% of the total project time, with mode = 50%) and reorganizing/restructuring existing tests (from 5% to 50%, with mode = 15%). The majority of subjects (87%) were involved with extending existing test cases, but this activity, on average, did not consume more than 15% of the total project time. Two-thirds of subjects played around with the new ideas, trying what-if scenarios etc. without being sure what will result – but no more than 10% of the time was used up for this activity. Only one person identified an "other" activity, which was navigating between tests, and it occupied 5% of that person's time.

Table 6. Effort Spent (N=17).

Team	Effort own, hrs	Effort pairing, hrs	Effort, partner-ship with develop-ment team, hrs	Total indivi-dual effort, hrs	Total effort per team, hrs
1	8	3	3	14	**35**
	18	1	2	21	
2	15	3	1	19	**19**
	N/A	N/A	N/A	N/A	
3	5	2	1	8	**13**
	3	1	1	5	
4	10	2	5	17	**26**
	5	2	2	9	
5	0	0	3	3	**16**
	10	2	1	13	
6	3	3	1	7	**23**
	10	3	3	16	
7	8	1.5	1	10.5	**26.5**
	12	4	0	16	
8	15	4	1	20	**29**
	8	1	0	9	
9	4	2	3	9	**15**
	1	5	0	6	

5.2 Efficiency

Table 6 contains information on the effort of the customer team members spent on writing the executable acceptance-test specification and communicating with the development team. We expected each person to contribute about 4 hours a week on this task (16 hours in total per person; or 32 hours per team). The total time spent per

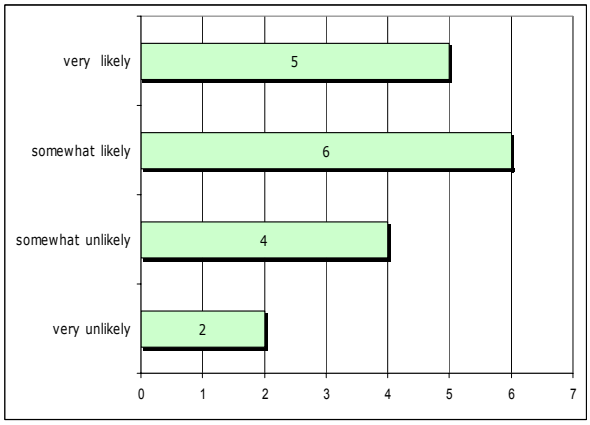
[2] The list of activities was inspired by the Cognitive Dimensions of Notations framework ([15], [7])

team (between 13 and 35 hours, with the median of 23 hours) suggests that customer teams were able to achieve their goals in reasonably-expected time frame or even sooner, with little overtime occurring.

5.3 Usefulness perceptions.

At the end of the project, subjects were asked about their perceptions of the technique. Figure 4 shows the distribution of the answers to how likely they would recommend using executable acceptance tests (in FIT, for example) for specifying business requirements, to a colleague.

Figure 4. Would Recommend to a Colleague?



Noticeably, no correlation between these perceptions and the quality of the produced specification during the course of the experiment was found. This is remarkable as it seems that the subjects are almost unaware of the quality the approach produced for them.

When asked whether they would use FIT/FitNesse on a regular basis at work (assuming it would be available), the opinions were split half and half. Some people expressed their strong desire to adopt this practice, as can be seen from the following testimonial:

> *"Yes, I would like to use FIT/FitNesse for specifying requirements provided my organization supports it. It is easy to use but the whole team (i.e. analysts and developers) must be committed to using it and communicating though acceptance tests".*

Several people indicated their preference to use FIT on a regular basis but with more practice. One respondent pointed out:

> "*Yes, [I would use it] but as a clarification and elaboration of narratives/event flows. The leap from story to executable acceptance test is too large to easily overcome*".

A member of the customer team that did an extremely good job with their specification (see footnote to Table 3), describes his experience:

> "*I had a difficult time with describing simple functionality. In some cases, I would find it easier to tell (verbal) a developer what I want. Also, I found the "setup" and pre-conditions to be tedious and painful. However, in my line of work, I have noticed a gap between business requirements and technical spec. I can see FIT providing the missing functional layer".*

This goes along with the notion that acceptance tests are meant to support communication, not to replace it.

One subject indicated that he "*liked to use FIT/FitNesse for simple scenarios*". However, for more complex ones, he indicated the preference for use cases. Another person considered executable acceptance testing to be complementary to "more formal requirements elicitation techniques". Several people, who did not express a clear 'yes' or 'no' preference, said that their decision would "*depend on the project and the maturity of the customer*".

Among those who rejected the practice was one person from a team that, nevertheless, created a high-quality specification:

> "*Writing FIT / FitNesse almost felt like writing code. I felt like I was literally specifying [to the] development team the details down to function names. On top of that I had to explain what I wanted in paragraphs before the test cases. I felt like I was doing double the amount of work*".

This speaks of an expectation of the role of the customer and the work involved.

To sum up, half of the sample found this technique promising and would consider using it at work and also recommend it to their colleagues; while the other half were more skeptical and did not plan on adopting the practice. Nevertheless, the fact that the practice was wholeheartedly accepted by some, and moderately by others, is indicative of its potential to be adopted by various industries.

6. Validity

Several threats to the internal and external validity have been identified. The biggest concern is the small size of the sample (18 individuals, 9 teams). We addressed this risk by using statistical procedures that are recommended for the small samples, for which the normality of data distribution is not assumed.

Another concern is the type of the sample – it was a convenience sample based on self-organized/self-selected teams. Because we did not have a random sample, we caution the reader to be conservative in the interpretation of our study results. In fact, we cannot claim direct cause-effect relationships that will hold in general, in many settings. We only report the trends that manifested themselves in the course of this experiment.

The fact that the teams were composed of the graduate students suggests two potential biases. The first one is that volunteers (students enroll in the course based on their interests of the topic) may bias the results because they are often more motivated and skilled than average industry customer. Graduate students are usually pre-selected and represent the best of the student population. Even though, many of them will end up in the industry, this sample may not be indicative of the population of the customer representatives from the industry. Despite, this, we used customer teams external to the development teams, and this constitutes, in our opinion, an improvement to the sampling approaches in a previous study (see discussion in Section 2 and [23]).

Business-grad students may also have had a strong expertise in computer science as they were taking an elective course in software engineering, which may point to some previous interest in computer science and technical topics.

We are also aware that cross-team discussion may have taken place and those may have skewed our results. In addition, there might be a possibility of an unequal workload when one pair partner may have contributed more than the other one but did not report it in the post-experiment questionnaire – this would have obscured the results and would not have created the main effect.

The project provided to the system also poses a potential bias. Though it was not trivial, it may not have been complex enough to approximate the real world. This would have produced weaker results-effects. In addition, the short-term involvement of the customer team (only 4 weeks on the part-time basis) may have lessened their level of engagement with the project implementation towards the end of the iteration.

Ultimately, the authors recognize the limitations of the study and therefore intend to conduct further case studies in the industry to reinforce and extend the findings of this research.

7. Conclusions

Executable acceptance test-driven development is a promising approach that helps to clarify customer expectations, express them with concrete examples, and establish contexts for future conversations about the requirements. Executable acceptance tests can "help focus the development process, with a major aim of providing the most useful feedback as early as possible to enable businesspeople to guide the development in the best direction, according to their latest understanding of the problem and the solution" [25].

Our study addresses a need to understand if and how executable acceptance testing technique can be used directly by customers, to support the elicitation and development of system requirements. To test this, we conducted an experiment with teams of customers, composed of both business and computer science graduates, who used FIT and Fitnesse to develop test requirements for a human resources information system, that were communicated and used by a development team composed of undergraduate students. Our results show that customers can specify functional requirements clearly, despite some initial difficulty in doing so, and that a number of both business and computer science graduates will recommend and use the executable acceptance testing technique in specific future practice scenarios. Future work needs to be conducted to confirm our initial results, and to address an intriguing finding that some who were dissatisfied with the approach produced some of the best specifications.

Acknowledgements

The authors would like to thank all University of Calgary students who participated in this research. This study is partially sponsored by National Sciences and Engineering Research Council of Canada (NSERC), Killam Trusts and iCore Alberta Informatics Circle of Excellence.

References

[1] Acceptance Test. Online: http://c2.com/cgi/wiki?AcceptanceTest

[2] Alexander, I. "Initial industrial experience of misuse cases in trade-off analysis", *Proc. IEEE Int. Conf. on Requirements Engineering (RE'02)*, IEEE Press: 61–68, 2002.

[3] Alexander, I. "Positive Results from Negative Scenarios". *Pres. for IDEX Project Challenge*, May 2002. Online: http://easyweb.easynet.co.uk/~iany/consultancy/negative_scenarios.ppt

[4] Bache, G. "What is Acceptance Testing?" Online: http://texttest.carmen.se/AcceptanceTesting/

[5] Basili, V.R. "The Role of Experimentation in Software Engineering: Past, Current, and Future"; *Proc. 18th Int. Conf. on Software Engineering*, IEEE Press: 442–449, 1996.

[6] Beck, K. *Extreme Programming Explained: Embrace Change, 1/e*. Addison-Wesley, Boston, MA, 1999.

[7] Blackwell, A.F., Green, T.R.G. Investment of attention as an analytic approach to Cognitive Dimensions. *Collected Papers of the 11th Annual Workshop of the Psychology of Programming Interest Group (PPIG-11)*: 24-35, 1999.

[8] Buwalda, H. "Soap Opera Testing". *Better Software, 6(2)*: 30–37, 2004.

[9] *Chaos Report*. The Standish Group, West Yarmouth, MA, 1995, 1997, 1999, 2001, 2003.

[10] Cohn, M. "Do-It-Yourself", *Better Software, 7(9)*: 18–22, 2005.

[11] Department of Defense. *Military Standard Defense System Software Development DOD-STD-2167,* section.5.3.3. Online: http://www2.umassd.edu/SWPI/DOD/MIL-STD-2167A/DOD2167A.html.

[12] FIT: The Framework for Integrated Testing Documentation. Online: http://fit.c2.com/wiki.cgi?FitDocumentation

[13] FitNesse Documentation. Online: http://www.fitnesse.org/FitNesse.UserGuide

[14] Fowler, M. "Specification by Example". Online: http://www.martinfowler.com/bliki/SpecificationByExample.html

[15] Green, T.R.G. Cognitive dimensions of notations. In A. Sutcliffe & L. Macaulay (Eds.), *People and Computers V*. Cambridge University Press: 1989.

[16] Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* New York, NY: 1990.

[17] Jeffries, R. "What is XP?" Online: http://www.XProgramming.com/xpmag/whatisXP.htm

[18] Kaner, C. "Cem Kaner on Scenario Testing: The Power of 'What-If…' and Nine Ways to Fuel Your Imagination", *Better Software*, *5(5)*:16–22, 2003.

[19] Kaner, C. "What is a Good Test Case?" *STAR East Conf. 2003*, May 2003. Online: http://www.testingeducation.org/a/testcase.pdf

[20] Kerievsky, J. "Storytesting". Online: http://industrialxp.org/storytesting.html

[21] Marick, B. "Example-Driven Development". Online: http://www.exampler.com, and http://www.testing.com/cgi-bin/blog/2003/09/05#agile-testing-project-4

[22] Marick, B. Agile Acceptance Testing Workshop Report, *XP/Agile Universe 2002 Conf*. Online: http://www.pettichord.com/XP_Agile_Universe_trip_report.txt

[23] Melnik, G., Read, K., and Maurer, F. "Suitability of FIT User Acceptance Tests for Specifying Functional Requirements: Developer Perspective". *Proc. XP/Agile Universe 2004*, *LNCS, Vol. 3134*, Springer Verlag: 60–72, 2004.

[24] Meyer, B. "On Formalism in Specifications". *IEEE Software*, *2(1)*: 6–26, 1985.

[25] Mugridge, R., and Cunningham, W. *FIT for Developing Software: Framework for Integrated Tests*, Prentice Hall, Upper Saddle River, NJ: 2005.

[26] Perry, W. *Effective Methods for Software Testing, 2/e*, John Wiley & Sons: New York, NY, 2000.

[27] Raha, S. Comment in [29], p.19.

[28] Read, K., Melnik, G., and Maurer, F. "Examining Usage Patters of the FIT Acceptance Testing Framework". *Proc. XP2005, LNCS, Vol. 3556*, Springer Verlag: 127–136, 2005.

[29] Reppert, T. "Don't Just Break Software, Make Software: How Story-Test-Driven-Development is Changing the Way QA, Customers, and Developers Work". *Better Software*, *6(6)*: 18–23, 2004.

[30] Sepulveda, C. "XP and Customer Tests: Is It Fair?" Online: http://christiansepulveda.com/blog/archives/cat_software_development.html

[31] Shadish, W.R., Cook, T.D., and Campbell, D.T. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin, 2002.