Advanced Software Engineering Lab 3

Allen P. Hild
allen.hild@lmco.com
ahild@du.edu
303.977.7147

May 23, 2006

Contents

1 Lab Report

1.1 Assignments 2-8

The data tables at the end of this section detail the faults found per team, a brief agreed upon description, as well as the results of running the capture-recapture code in Matlab.

Our team replaced the norminv function, in the provided Matlab .m file, with erfcinv function according to the Matlab help file using the relationship:

$$\text{norminv(p)} = -\sqrt{2}\text{*erfcinv(2*p)}$$

So in the Matlab mhjke2.m file we replaced:
confProbVal = norminv(1-(1-confProb)/2);
with
confProbVal=-sqrt(2)*erfcinv(2*(1-(1-confProb)/2))

The mhjke2 method was run using first order and 0.95 confidence probability for the additional two inputs.

It is interesting to note that m0mle and mtmle methods reported the same output, even though mtmle is assuming variability between viewers. I speculate that this might have been because of too few data points to the input set for finding faults was too similar. The same behavior was also exhibited using the cumulative data between the two teams.

For the capjke and mhjke2 methods that reported the standard deviation, the values reported were almost 10% of the number of remaining faults - with the authors minimal insight into estimating techniques, 10% seems high for a standard deviation especially when considered with the size of the confidence intervals.

Other ways to estimate fault content could include:

- Guessing
- Asking for subject matter expert evaluation
- Using lessons learned or prior history of faults

Metrics that would be useful to capture and archive when trying to estimate faults are:

- length of document
- new or old requirement tag (meta-data)
- time spent reviewing (or other measure of complexity)
- capability of the creator (labor grade - year experience)
- frequency of faults from historical information

Reviewer 1	Allen P. Hild
Reviewer 2	Jon Fisher
Reviewer 3	Justin Pfeffer
Reviewer 4	Christine Phung

Enter 0 for No Fault Caught
Enter 1 for Fault

Initial Time spent Reading Document was approximately 1 hour.
Time for Usage based reading for all of the use cases was approximately 1 hour.

Data Table

Bug ID	Use Case	Team A/J	Team C/J	Description of Bug	Reviewer 1	Reviewer 2	Reviewer 3	Reviewer 4
1	2.1	1	1	no mention of terminal in req's	1	0	1	0
2	2.1	0	1	no specification in time out for UC/Req: ref 3.1.28	0	0	1	1
3	2.1	1	1	3.1.23: ambiguity - submittal of order cf	1	0	1	1
4	2.1	0	1	3.1.26 - if driver is with customer - will not submit additional order when you already have a customer	0	0	1	0
5	2.1	1	0	central does not have a method to receive orders: ambiguous	1	1	0	0
6	2.1	1	0	confirmation of orders within central: ambiguous	1	1	0	0
7	2.2	0	1	automatically dispatched - UC does not mention manual dispatched	0	0	1	0
8	2.2	1	1	no algorithm mentions for Auto dispatch:3.2.13 not clear	1	0	1	0
9	2.2	1	1	no order attributes mentioned in the functional requirements; pick up time priority -	1	1	1	1
10	2.2	0	1	no details on availability information - missing details	0	0	1	0

Data Table

Bug ID	Use Case	Team A/J	Team C/J	Description of Bug	Reviewer 1	Reviewer 2	Reviewer 3	Reviewer 4
11	2.2	1	1	no req's on cancelled or denied order - how they are handled - automatic	0	1	1	0
12	2.2	1	1	3.2.17: does not mention is this is manual or auto - ambiguous what mode of operation	1	0	1	1
13	2.2	1	0	operator does not have mechanism to receive orders	1	0	0	0
14	2.2	1	0	specific request - not well defined within properties of taxi	1	1	0	0
15	2.2	1	1	allergy - not defined within requirements	1	0	0	1
16	2.2	1	1	very close in time is very ambiguous:3.2.16	1	1	0	1
17	2.2	1	0	UC - order of operations in UC could be different	1	1	0	0
18	2.3	1	1	traffic overview not defined	1	1	0	1
19	2.3	0	1	no definition of algorithm for estimation of time, multiple algorithms??	0	0	1	0
20	2.3	1	1	Display - information display - not well defined	1	1	0	1
21	2.4	1	1	no req. to specify the authentication method: id card doesn't exist: how the driver logs in: verification of login	1	1	1	1
22	2.4	1	1	3 and 5 could be zone information being sent multiple times: duplication of info being sent	1	1	1	0
23	2.4	1	1	Duplication of information being sent:	0	1	1	0
24	2.4	1	0	3.1.42: Each zone	1	0	0	0

Data Table

Bug ID	Use Case	Team A/J	Team C/J	Description of Bug	Reviewer 1	Reviewer 2	Reviewer 3	Reviewer 4
25	2.4	1	0	Update: definition of all triggers not specified	1	1	0	0
26	2.5	1	1	2 minutes for time out not mentioned w/in the spec: ref3.1.28	1	1	1	1
27	2.5	1	1	no variant on the taxi being in the soon available state: How does the state transition	0	1	1	1
28	2.5	1	0	Central does not have a way to confirm accepted orders	1	1	0	0
29	2.5/2.6	1	1	Waiting for customers - as driving or stopped - ambiguous states: Can't drive then wait then be waiting to drive.	0	1	1	0
30	2.6	0	1	No details for the "waiting charge" if customer is not at pickup site	0	1	1	1
31	2.6	1	1	Soon available state is poorly defined: is it automatically set or set by the driver	1	1	1	1
32	2.6	0	1	Meter is not turned off in UC:	0	0	0	1
33	2.6	1	1	Driver picks up customer w/o order. This does not seemed to be allowed in the Req	0	1	1	0
34	2.6	1	1	Confident vs. Knows terminology: 3.1.11	1	0	0	1
35	2.7	1	1	Operator determines course of action, reset	1	0	1	1
36	2.7	0	1	UC doesn’t mention transmission of position data	0	0	1	1
37	2.8	1	1	deficiency in 3.1.33: terminating the link	1	0	0	1
38	2.9	1	1	Updates: are these sent whenever any taxi in the systems changes; is the info then sent to all taxis	0	1	1	0

Data Table

Bug ID	Use Case	Team A/J	Team C/J	Description of Bug	Reviewer 1	Reviewer 2	Reviewer 3	Reviewer 4
39	2.10	1	1	3.2.22: Terminology: rejects vs. denied, cancelled, ignored	1	0	0	1
40	2.10	0	1	3.1.25: no converse (analagous req for central) to say which states can be dispatched to the taxi.	0	0	1	0
41	Req	0	1	3.1.7: Drive has no state:	0	0	0	1
42	Req	0	1	3.1.5, 3.1.10 Inconsistent use of "System" and sub compenents	0	0	0	1
43	Req	0	1	3.1.6 etc who/what is sending the informations	0	0	0	1
44	Req	0	1	3.1.17-19; 3.2.7-9 Redundancy - poorly worded	0	0	0	1

			capjke			mhjke2			Allen's
	m0mle	mtmle	faults	std deviation	confidence interfal	faults	std deviation	confidence interval	Subjective Estimate (min,max)
Team AJ Errors	35	35	38	3.5707142	34,49	39.5	3.5707142	35,49	30,50
Team CJ Errors	49	49	47	4.330127	42,59	48.5	4.330127	43,60	n/a
Cumulative Errors	46	46	51	4.0594545	47,64	53	3.968627	48,64	40,50

2 Book Exercises

2.1 Exercise 2: A software engineering group is developing a mission-critical software system that guides a commercial rocket to its proper destination. This is a new product; the group and its parent organization have never built such a product before. There is a debate among the group as to whether an inspection or walkthrough is the best way to evaluate the quality of the code. The company standards are ambiguous as to which review type should be used here. Which would you recommend and why?

The best overall solution would be to use both inspections and walkthroughs to evaluate the code. A process of evolution is the best way to handle the code reviews. First having informal walkthroughs with members of the immediate team. Then when the code/design has been matured hold more formal inspections with member of the overseeing Systems Engineering team. This would allow for member of the development/design team to be very familiar with the artifacts and potentially reduce errors early on by having "everyone on the same page". The inspections could then serve a similar step but at a higher level in the overall system design.

Lastly, the company should seriously consider formalizing it reviewing processes. The system they are developing is new, and team is not very experienced - the implication being this is a high-risk development effort. The affect of a failure could be catastrophic and possibly cause the business to fail.

2.2 Exercise 3: What size of a review team would you recommend for the project in Problem 2, and why? What are the different roles for members of the review team? Which groups should send representatives to participate in the review?

The minimum size of the review team would be three members, and a suggestion of less than ten members. The thought being that you need to find a balance where you don't have too few or too many "cooks in the kitchen". Too few, and you may not have enough objectivity, could miss errors, or might overtax the review team requiring too many hours to review the code. Too many, and it might restrict the usefulness of the review, and cost a significant amount of time and money.

For walkthroughs a smaller more intimate team of reviewers who work together on a regular basis can provide feedback without restraint - i.e. they may feel more comfortable providing feedback to the developer, sentiment that could in a larger setting cause hard feelings.

The following roles should be considered for a review:

- Moderator (paramount to the success of any review)
- Developer
- Reviewer (1+)
- Scribe
- Reader

Groups that should be represented at a review might be:

- Systems Engineering
- Development Team
- Quality Assurance
- Test Group
- Maintenance
- Other non-related Development team (for independent review)

If possible avoid the inclusion of members of the management team at all costs. Their objectivity could easily be comprised by outside pressures, and it could be very easy for the code review to become the developers review.

2.3 Exercise 10: There is some debate as to whether code should be compiled [clean compile] and then reviewed, or vice versa. Based on your own experiences give an opinion on this matter.

My own experience would be to cleanly compile the code first. Most of today's Integrated Development Environments (IDEs) handle a majority of syntax as well the problems of not initializing variables, mixing types, etc. In addition, the benefit of cleanly compiling the code will serve to maximize the effectiveness of the review teams time. Spending the time and money for engineers (high hourly rate) is not the best use of an experienced engineers time. Their time could be better spent focusing on the higher level architecture problems or problems of greater complexity.

References

[1] Burnstein, I., *Practical Software Testing - A Process-Oriented Approach*, Springer-Verlag, 2003

[2] Sommerville, Ian:*Software Engineering Seventh Edition*, Addison Wesley, 2004