



# CHI 2011 Study

Added by [Todd Kulesza](#), last edited by [Todd Kulesza](#) on Aug 05, 2010

## Purpose

This study will explore the concept of WYSIWYT as it applies to end users debugging machine-learned programs.

## Schedule

Date	Milestone	Responsibility
April 5	Prototype mockups ready for CogTool analysis	Todd
April 8	CogTool analysis completed	Kevin
April 12	Prototype design finalized, incorporating additions from CogTool analysis	Todd
May 1	IRB submitted	Amber
May 3	Functional prototype ready	Todd/Amber/Moy
May 4-6	Sandbox prototype to ensure it's giving us useful data	Todd/Amber/Kevin
May 7-31	Bug fix and tune prototype using results of sandbox tests	Todd/Moy
May 24-28	Sandbox experiment	Todd
June 1-4	2nd run of sandbox pilots	Todd/Kevin
June 7-11	Run the pilot study	Todd/Kevin
June 14-18	Run the study	Todd/Simone
September 17	Everyone leaves for VL/HCC, effective CHI deadline	Paper-writing gnomes
September 24	CHI deadline	Paper-writing gnomes

## Research Questions

1. Which of the coverage metrics helps a user find the most faults in the shortest amount of time?
  - a. Is there a difference in types of faults found? (ambiguity faults vs. rarity faults vs. "surprise" faults in which the system is very sure of itself but still wrong). Our hyp: confidence will do better with ambiguity faults, the other two will do better with rarity faults. We have little hope right now of getting any of the "surprise" faults.



2. Which coverage metric gives users the most accurate perception of the accuracy of the learned program? (This measures users' ability to judge how close to being "done" they are. It matters because in the real world, when people decide they're done testing/debugging, that's when they start using the program for real and trusting its outputs. So if they guess judge about being done, they trust faulty programs.)
3. Which coverage metric results in the most accurate learned programs?
  - a. Difference in faults/bugs introduced by users in the process?
4. Do users of one coverage metric finish faster than others? (MMB doubts that we'll get this, but ok to try.)
  - a. Do users of one coverage metric work with more segments than others?
  - b. And if some finish faster, did their number of faults detected suffer or was it just as good?
5. What is user comprehension of the coverage feedback (asked in a post-session questionnaire with 5-ish border-color/progress-bar/density interpretation questions and 5-ish border-color/progress-bar/density prediction questions).
6. User-as-oracle: How often was the user right (i.e., when they say a value is wrong, was it actually wrong?)
7. Which of the 4-tuple marks do users use the most?
8. The usual questions based on the NASA TLX questionnaire.
9. Which coverage criteria gave the most useful (for finding faults) testing picks (collected from questionnaire asking users' opinion of this).
10. Which system did users like the best (questionnaire asking them to pick their favorite system).
11. Do users in Control find bugs that would have been highly prioritized by Similarity or Confidence prioritizers?
12. Strategy questions (qualitative), eg.
  - a. Which strategies did they use? (eg, follow bright green vs. follow topic vs. localized-topic, vs. class balance vs. ....) NOTE -- you cannot get this just by treatment group or just by sort order.
  - b. Which strategies were most effective at finding faults?
  - c. Which strategies were most effective at coverage?
  - d. Did they focus on low-priority or high-priority or a combination?
  - e. Strategies and "surprise faults"?
  - f. Did some treatments lead to particular strategies more than others?
  - g. Sometimes: x-dimension is time, other dimension is interesting thing (eg, priority, % testedness (progress bar), % faults found).
  - h. This is QUALITATIVE.

## The Coverage/Prioritization Criteria:

1. Do we want 4 treatments? Control vs. Confidence (pies) vs. Similarity (fishbowls/beakers) (don't know which flavor yet) vs. Least-Relevant ?
  - a. From 2010-04-01 meeting, it looks like yes, we've tentatively agreed to use these three metrics.

## Decisions about how the system will work:

1. We have 2 things to communicate to the user. We will separate this into 2 devices, not fuzz them together, **because we want the user to be able to predict exactly what the colors/widgets will look like if they take an action**.
  - a. Tested-Similarity Coverage: How tested is the collection of instances? How tested is each instance?
  - b. Prioritizing: How do we want the user to prioritize their testing time?

2. What is "Tested-Similarity" Coverage:
  - a. It's important that 1 test potentially gets MORE THAN 1 instance tested. Otherwise, this is not like "real" testing at all.
  - b. Potential solution that WKW and Peggy buy: Suppose instances A and B have the same label AND ~~same similarity score to the centroid of that label/class~~ are similar to each other (by some metric, eg the k% most similar). Then approving A (checkmark, turns border "tested") makes B tested (no /, but border turns "tested" color) too.
  - c. Note that what we have here is "tested-similarity coverage", ie we're dividing the world into "similarity classes" in which the TEST set instances that are similar to ONE THAT THE USER TESTS are all covered by a test of that one test.
3. Retraining/remetric'ing: It's important for the system to retrain after each test, and to recompute the priority metrics too. If we don't, we'll have to work very hard explaining to reviewers why we did not do this obvious thing. WKW/Moi to look into the speed with which this can be done. (Possible objection: it's a moving target for the user. Answer: yes, but it's totally artificial not to do this.)
  - a. From 2010-04-01 meeting, it looks like the speed of retraining will not be a problem.
  - b. From 2010-04-02 email from WKW, it looks like speed of re-metric'ing is also not a problem: "Moy has done the distance computation test and calculating 20 x 20 distances calculations is really fast (on the order of milliseconds), so no worries there"
  - c. When an instance has been explicitly tested (/ed), it moves from the test set to the training set.
  - d. The system predicts elements of ONLY the TEST set. It does not predict anything in the training set, and thus does not re-predict the thing we just /ed off, since it's now in the training set.
4. UI representations:
  - a. Tested-similarity coverage is represented by border colors:
    - i. Bright border (eg: bright blue?) means not tested. We want the user to want these to go away.
    - ii. Tested ordinary border (eg: black?) means tested with a fair amount of user's confidence: /d or X'd or similar to / or X.
    - iii. Between (eg: dim blue?) means tested with less user's confidence: v'd or x'd or similar to v or x.
  - b. Priority criteria (how much we want User to look at this) is represented by widget density (eg, of the pie or fishbowl):
    - i. Really dense: Similar to X'd or x's instance.
    - ii. Really dense:
      1. If Confidence metric: system is unconfident.
      2. If Similarity metric: really unsimilar to training centroid.
      3. If Relevance metric: has almost no relevant features.
    - iii. Normal density: It and its similar friends have never been tested, but they don't satisfy the other criteria in this list of priority criteria.
    - iv. Faded: Similar to something already tested (this trumps #i above).
    - v. Faded: Segment's characteristics are pretty opposite of #ii above.
5. The progress bar is the aggregation of the border colors. So, it's testedness coverage. There's a failed-tests fraction, an approved-tests fraction, and a not-tested fraction. So if there are 10 samples, 1 is failed, 2 are approved (and/or similar to approved), and the rest are not-tested, the progress bar shows 10% failed, 20% approved, and 70% not-tested.
6. Actions the user can take: check , fuzzy-check, X, fuzzy-X, or change label. And they can undo. And they can hover to get a tooltip.
  - a. Check  UI: If they /, that instance gets "tested" border (or whatever), and so do "tested-similar" other instances (as per above). The confidence-pie (or whatever) for THIS instance (only) is replaced by a / (fuzzy-check: the / is fuzzier or less saturated). Also progress bar gets updated (as per above). ML: this instance is now considered part of the training set, and system retrains.

- b. X UI: Border changes like /. This instance's pie is replaced by a X. ML: This instance does NOT go to the training set.
- c. Label change: That's the equivalent of: X, change-label, /. So at least in theory, everything that would have happened for X happens first (but where the system would have used "?" instead it uses the new label), then everything that would have happened with / on the new label happens.
- d. Undo: UI: We can support this most easily with the 4-tuple, simply allowing them to pick a different one or go back to not-tested as a 5th option. ML: We need to think carefully about which training sets it may have already been in, removing from those, and so on...
- e. Tooltips: we need to tooltip the pie (or whatever), the border (or whatever), the checkmark/X that has replaced the pie, each element of the 4-tuple, and maybe the progress bar. The tooltips should follow SER principles: Be short, give just enough info to explain what the user is looking at (eg, "The system thinks there's a 30% chance this should be Soccer, with smaller chances for other tags.", suggest the action we want them to take (eg, "/" if Soccer is right, change tag or X if it's wrong"), and say what the benefit would be ("Testing helps find errors."). Or worded more tightly if possible.

## Experiment design

Within-subject.

1. 32 subjects overall. Plus no-shows, plus 4 pilots. Specifically:
  - a. We need **32 (4 people x 8 sessions), plus calculating in 2 no-shows per session = 48 participants, + 4 pilots = 52.**
2. Each subject completes the following 4 tasks/treatments (So 32 subjects total):
  - a. control, distance, least-relevant, confidence
  - b. order of tasks is varied to avoid learning
  - c. We are making them do all
  - d. 5 subjects (approx) per session.
3. Subjects are RANDOMLY ASSIGNED a session # (eg, by throwing a many-sided die). If the session # is not good (eg, the subject cannot come then, or their participation would throw off gender balance for that session), reject that session# and throw the die again.
4. Counterbalancing treatments (different message sets for each treatment).
  - a. So, a total of 4 message sets across participants and treatments.
  - b. Message sets will be varied across participants- ex. P1 gets M1 first then M2, P2 gets M2 first then M3, etc.
  - c. Order of Treatments will be randomized among ~~participants~~ sessions.

Session Treatment/Message sets (assuming M0 is used for tutorial):

Treatment Sequence #1 - <ul style="list-style-type: none"> <li>• Control</li> <li>• Confidence</li> <li>• Least Relevant</li> <li>• Distance</li> </ul>	Treatment Sequence #5 - Like #1 -
Treatment Sequence #2 - <ul style="list-style-type: none"> <li>• Confidence</li> </ul>	Treatment Sequence #6 -Like #2

<ul style="list-style-type: none"> <li>• Least Relevant</li> <li>• Distance</li> <li>• Control</li> </ul>	
Treatment Sequence #3 - <ul style="list-style-type: none"> <li>• Least Relevant</li> <li>• Distance</li> <li>• Control</li> <li>• Confidence</li> </ul>	Treatment Sequence #7 -Like #3
Treatment Sequence #4 - <ul style="list-style-type: none"> <li>• Distance</li> <li>• Control</li> <li>• Confidence</li> <li>• Least Relevant</li> </ul>	Treatment Sequence #8 - Like #4

**Message sequence:** Each SUBJECT in a session gets a different Message Sequence. We throw a 24-sided dice ( $4*3*2*1=24$ ) for each SUBJECT and use Randomization to assign message sequences to subjects

1. 1234
2. 1243
3. 1324
4. 1342
5. 1423
6. 1432
7. 2134
8. 2143
9. 2314
10. 2341
11. 2413
12. 2431
13. 3124
14. 3142
15. 3214
16. 3241
17. 3412
18. 3421
19. 4123
20. 4132
21. 4213
22. 4231
23. 4312
24. 4321

Prioritization widgets per treatment:

1. Control: none

2. Confidence: the confidence indicator
3. Least Relevant: the least relevance indicator
4. Distance: the distance indicator

### Procedure

1. Informed Consent(We need to figure out if we need this?)
2. Administer Background questionnaire
3. Administer system tutorial (explain the all the possible widgets and what will happen in the course of the experiment. # This only occurs once).
4. First treatment (Task 1) (15 minutes??). Participants will test all of the same message set . Randomized treatments.
5. For all users
  - a. All users receive the same message set as the other users for each task.
6. For each of the tasks, users have the following options:
  - a. Users can instruct the machine by marking a check, fuzzy-check, X, fuzzyX
  - b. Users can undo and hover over certain features (their markings, border, pie) to get a tooltip
  - c. User can select the "I'm done" button if they think they've finished testing.
1. Post-task questionnaire administered
2. Second treatment (Task 2): Change to next treatment.
3. Post-task questionnaire administered
4. Third treatment (Task 3): Change to next treatment.
5. Post-task questionnaire administered.
6. Fourth treatment (Task 4): Change to next treatment.
7. Post-task questionnaire administered.
8. Administer post-session questionnaire
9. Get receipt for money.

### Roles:

- Kevin: makes sure all the necessary materials are present and organized the right way.
- Tutorializer, driver, and helper: set up the room.
- Tutorializer: gives the tutorial, manages timing of things, might help out a little in between tutorial moments.
- Driver: drives during the tutorial. That person could also do data entry in between driving.
- Helpers: help throughout. Keeps the subjects from straying from the tutorial instructions, gets them out of trouble if they have trouble. Very important role.
- Door person: is the greeter. Does these things:
  - Checks people off the list.
  - If anyone hasn't shown up, calls them. (bring your cell phone).
  - Once the session starts, either admits late people or, after a certain point, turns them away (after signing them up for a different session).
  - Then becomes a helper for a bit, then ok to leave.
- Todd: pays and gets receipts at the end. (Unless he delegates to someone else.)
- Everyone: remaining at the end put everything away.

### [To Do List for the Tutorial](#)

## Vocab

- Subject - The component/message header that holds the subject line and what the user must click on in order to interact with the slider and widget.
- Message - The newsgroup document that pertains to the subject being tested.
- Topic - The label/classification of that the message
- Label - The action of the user or machine giving a topic to a message

## Code

- The source code is hosted on Beaver Source at <http://beaversource.oregonstate.edu/projects/wysiwy4ml/wiki/WikiStart>.
- Instructions for setting up Eclipse are available [here](#)

## Other Useful Info:

- Computer accounts are [here](#)

**Labels:** None

### 4 Child Pages

[Analysis](#)

[Experiment to do List](#)

[Logging](#)

[Setting Up Eclipse \(CHI 2011\)](#)

[Add Comment](#)

*Printed by Atlassian Confluence 3.5, the Enterprise Wiki.*