Teaching and Training for Citizen Cyberscience

David P. Anderson
Space Sciences Lab
U.C. Berkeley

Sept 10, 2008
Citizen Cyberscience

Using the Internet, involve the global public in current scientific research

- Volunteer computing
  - 50 projects
  - 500,000 participants
- Volunteer thinking
  - 10 projects
  - 50,000 participants
Why teach volunteers?

• Volunteer thinking
  - Training
  - Initial skill assessment

• Volunteer computing
  - Get people interested
  - Get people invested
    • create deep knowledge project’s science
    • knowledge as status
Facts about learning

- People learn by doing
- People have different preferred learning modes
- Long-term retention requires repetition
To test or not to test

• Unmeasured teaching
  − needed for initial teaching
  − how effective? Hard to know

• Measured teaching
  − include exercises/tests/quizzes

• Advantages of exercises:
  − It’s how people actually learn
  − tells you how much individuals have learned
    • status level
    • skill classification
  − tells you how good your lessons are
The volunteer “student body”

- Diversity
  - Age
  - Sex
  - Nationality
  - Education level
- Churn
- Motivation

Compared to formal education:
- Harder because more diverse
- Easier to experiment
Experiments

- Collect data for a few days
- Compute 90% confidence intervals of score means
- May find:
  - No significant difference
  - Lesson 1 is better for all students
  - Lesson 1 is better for an identifiable group
Adaptive courses

- Possible adaptation criteria
  - demographics or other user-supplied info
  - simple derived data
    - e.g. average exercise scores
  - complex derived data
    - e.g. cluster analysis
Bolt: middleware for Citizen Cyberscience teaching

• What does “middleware” mean?
  − sequencing
  − navigation
  − maybe other features

• Bolt design goals: make it easy to
  − create exercises
  − conduct experiments
  − change courses on the fly
  − create adaptive courses

• Bolt provides analytics to:
  − find “trouble spots” in course
  − understand experiment results
Integration with BOINC

- BOINC: volunteer computing
- Bossa: volunteer thinking
- Bolt: teaching, training

BOINC Basics: accounts, groups, credit, communication
Implementation

Student

browser

- Selects next item
- Adds navigation
- Records results in DB

Bolt scheduler

Bolt database

Course document

Lessons

Exercises

Interactions, student state warehoused here
Lessons

• Any HTML or PHP file
  - may include graphics/audio/video/Flash/applets

• Bolt adds:
  - navigation (Next/Back buttons, Question link)
  - your header/footer
Exercises

- PHP script using library of question types
  - multiple choice
    - Bolt randomizes order
  - fill in the blank
  - click on image
- Can have >1 question
- Same script is used to
  - show exercise
  - grade results
  - show answer page
Course documents

• PHP script returning a hierarchy of “units”
• Types of units:
  - items
    • lesson
    • exercise
  - control structures (nestable)
    • sequence
    • exercise set
    • select
    • random
Example course document

return sequence(
    name('course'),
    lesson(
        filename('conifer_intro.php')
    ),
    lesson(
        filename('conifer_decid.php')
    )
);
Using the power of PHP

function sample_select() {
    return select(
        name('sample select'),
        valuator('my_rand'),
        lesson(
            filename('bolt_sample_lesson.php?n=1')
        )
    );
}

function sample_xset() {
    ...
}

return sequence(
    name('course'),
    sample_select(),
    sample_xset(),
);
Changing course structure on the fly

- What if course changes under a student?

State (stored in DB):

- “course” => (index:0, name:“section 1”)
- “section 1” => (index: 1, name:”lesson 2”)

Like a call stack, but associative
Course maps

• For each lesson and exercise, show stats of:
  − “outcome”: Next, Prev, none
  − viewing time
  − exercise score

• Can filter or break down by
  − sex
  − age
  − other criteria easy to add
function my_rand($student, $unit) {
    return rand();
}

select(
    name('sample select'),
    valuator('my_rand'),
    lesson1(),
    lesson2(),
);

exercise_set(
    name('sample exercise set'),
    exercise1();
);
Analyzing an experiment

- Choose select and exercise set units
- Make a data snapshot
- View 90% confidence intervals for score means
- Filter, breakdown by student attributes
Possible experiments

• Text
• Learning modes
  - verbal, audio, graphical, experiential
• Presentation
  - color/font
  - typography
  - voice
• Language level
• Lesson granularity
• Density of exercises
Course evolution

• Run course for a while
• Study course map
  • Where are students quitting?
  • Where are they scoring too high or too low?
    • Is course too easy or hard?
    • For specific student groups?
• Create new lessons/exercises
• Add experiments
• Study results of experiments
• Repeat
Reviewing and repeating exercises

```java
exercise_set(
    name('exercise set 1'),
    number(2),
    exercise1(),
    exercise2(),
    repeat(.3, basic_review(), REVIEW),
    repeat(.7, int_review(), REVIEW|REPEAT),
    repeat(1, null, REPEAT|NEXT)
}
```

- If score < 30%, must review and repeat
- If 30% < score < 70%, must review or repeat
- If 70% < score < 100%, can repeat or move on
Memory refresh

• Students repeat exercise set at specified intervals
• Notifications via web or email

exercise_set(
    name('exercise set 1'),
    exercise1(),
    exercise2(),
    refresh(array(7, 14, 28))
);
Conclusion

• Citizen cyber-science needs teaching
• It presents both challenge and opportunity
  - diversity
  - continuous arrival
• Bolt: a framework for CCS teaching
• Use Bolt if:
  - you care what students are actually learning
  - you have resources to evolve course