Errors and Debugging of Blocks-Based Programming in App Inventor

Benjamin Xie, Supervised by Hal Abelson
MIT EECS – Google Research and Innovation Scholar

Objective and Motivations

Goal: Understand how users cause and recover from errors.
- Context leading up to errors?
- How users debug errors?
- Which blocks involved? What were intended blocks?
- Differences in user types?

Motivation: Need to Better Understand Visual Programming Languages
- Important teaching tool, especially for younger students
- Need improved teaching resources for blocks languages, transition from blocks to text languages

Example of Blocks-Level Analytics

Blocks-Level Analysis collects data on the step-by-step process of app creation. It logs “relevant” actions with timestamps so that the entire process could be recreated and analyzed.

In this example, a user has confused text and variable blocks. He removes the erroneous text block, deletes it, and adds the variable block. This could suggest a lack of understanding of data types.

Research Questions

Bump Away Errors
- User attempts to connect blocks that logically should never attach.
- System prevents blocks from attaching by “bumping them” away
- Error nonexistent in text languages

Runtime Errors
- Live Development Mode: Mobile device paired w/ App Inventor, users can test program as they develop
- Can detect runtime errors during live development

Current Experiments

In Situ
- Users use AI as they typically would
- Objective: Assess frequency of errors, identify commonalities in occurrence of types of errors
- Currently deployed in high school classroom and ~30 self-selected users

Formal
- Users have specific task or encounter same scenario (ex: same bug). Users interact w/ code they have not written
- Objective: assess different styles of debugging given same environment
- Currently deployed in high school classroom

System

Function
create_block(data)
{
    POST(db, data)
}

1. Calls to database are embedded in App Inventor code for Blockly. Calls are asynchronous, made with Google Closure.
2. Data is stored in CouchDB database (JSON format)
3. Data is extracted and processed with Pandas, a python analytics library.

Example of Blocks-Level Analytics

1. text_box disconnect logic_equals
2. text_box delete -
3. get_variable connect logic_equals

Research Questions

Bump Away Errors
- User attempts to connect blocks that logically should never attach.
- System prevents blocks from attaching by “bumping them” away
- Error nonexistent in text languages

Runtime Errors
- Live Development Mode: Mobile device paired w/ App Inventor, users can test program as they develop
- Can detect runtime errors during live development

Current Experiments

In Situ
- Users use AI as they typically would
- Objective: Assess frequency of errors, identify commonalities in occurrence of types of errors
- Currently deployed in high school classroom and ~30 self-selected users

Formal
- Users have specific task or encounter same scenario (ex: same bug). Users interact w/ code they have not written
- Objective: assess different styles of debugging given same environment
- Currently deployed in high school classroom

System

Function
create_block(data)
{
    POST(db, data)
}

1. Calls to database are embedded in App Inventor code for Blockly. Calls are asynchronous, made with Google Closure.
2. Data is stored in CouchDB database (JSON format)
3. Data is extracted and processed with Pandas, a python analytics library.