MICROTASK

- Decontextualization of work
- Offering a short, self-contained unit of work with a clear objective
- Enable a contribution to be made in isolation of other ongoing work and with no requirements for prior knowledge
**APPROACH**

**Decontextualize Contributions:**
- *Local edits to a single artifact*
- Offering a preconfigured environment

**Automatically Generate Microtasks**
- *Track artifact state*
- *Signal interface changes across dependencies*

**Nurture Quality through Iteration**
- Allowing long contributions enables weak contributors to lock artifacts
- *Support reviews and tests*
CrowdCode Tutorial of primary interface elements

After selecting the first Microtask, a tutorial explain how to perform the task by examples

Complete/skip/submit

CrowdCode Microtasks

After the user request a new project creation:
  - Edit a Function
  - Write Test Cases

After Submission of the Functions, a new Edit a function is generated to continue work

Writing Test
  - List of tests
  - Separate WriteTest microtask for each test (work parallel)

Fig. 2. A step in the microtask tutorial offered when workers begin their first Edit a Function microtask.
<table>
<thead>
<tr>
<th>Microtask</th>
<th>Editor</th>
<th>Context views</th>
<th>Possible contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Function Description</td>
<td>Function description editor</td>
<td>Stub Viewer with requested function, implementation of requesting function, ADTs</td>
<td>(1) Function description, (2) Report function as not implementable</td>
</tr>
<tr>
<td>Edit a Function</td>
<td>Code editor</td>
<td>ADTs, diff of change to function signature (if any)</td>
<td>(1) Code, pseudocode, and stubs, (2) Report function as not implementable</td>
</tr>
<tr>
<td>Debug Test Failure</td>
<td>Code editor</td>
<td>ADTs, Test Runner</td>
<td>(1) Code, pseudocode, and stubs, (2) Report issue in test</td>
</tr>
<tr>
<td>Reuse Search</td>
<td>Function search</td>
<td>Stub View of requested function, implementation of requesting function</td>
<td>(1) Identify existing function providing requested behavior, (2) No function found</td>
</tr>
<tr>
<td>Write Call</td>
<td>Code editor</td>
<td>Description and signature of identified function, ADTs</td>
<td>(1) Code, pseudocode, and stubs</td>
</tr>
<tr>
<td>Write Test Cases</td>
<td>Test case editor</td>
<td>Description and signature of function, ADTs</td>
<td>(1) List of test cases, (2) Report issue with function</td>
</tr>
<tr>
<td>Write Test</td>
<td>Test editor</td>
<td>Test case, description and signature of function (and diff, if any), ADTs</td>
<td>(1) Test, (2) Report issue with function, (3) Report issue with test case</td>
</tr>
<tr>
<td>Review</td>
<td>Review</td>
<td>Contribution and original task context</td>
<td>(1) Review and rating</td>
</tr>
</tbody>
</table>
**SYSTEM**

- CrowdCode Tutorial of primary interface elements
- After selecting the first Microtask, a tutorial explain how to perform the task by examples
- complete/skip/submit
- CrowdCode Microtasks(insert image)

**After the user request a new project creation:**
  - Edit a Function
  - Write Test Cases

*After Submission of the Functions, a new Edit a function is generated to continue work*

**Writing Test**
  - Identifying all list of tests
  - Separate WriteTest microtask for each test
A worker was asked to edit the code of the function `doSpreadsheetCommand`. Can you review this work?

```c
/**
 * Executes a command, updating the Spreadsheet contents. Commands may include:
 * InsertColumnBefore, InsertRowBefore, DeleteColumn, DeleteRow, Paste.
 *
 * @param Spreadsheet spreadsheet, the spreadsheet document
 * @param Spreadsheet document
 * @param view View view, the current user's
 * @param String command, the command to execute
 *
 * @return Spreadsheet
 */

function doSpreadsheetCommand(spreadsheet, view, command){
    // Mark this function as implemented by removing this line.
    return null;
}

function doInsertColumnBefore(spreadsheet, view, command){
    switch(command){
        case 'InsertColumnBefore':
            spreadsheet = doInsertColumnBefore(spreadsheet, view);
            break;
        default:
            throw new Error('Unknown command!');
    }
    return spreadsheet;
}

/* insert a column before the selected one
 */

# RATING SYSTEM
1 Star: Incoherent or unfocused
2 Stars: Unconvincing or weak
3 Stars: There are some weaknesses
4 Stars: Good quality, without weakness
5 Stars: Excellent without weakness

# AVAILABLE DATA TYPES
> String
> Number
> Boolean
> Spreadsheet

---

nice decomposition
A worker editing the function `doSpreadsheetCommand` requested a call to a function providing the behavior of `doInsertColumnBefore`. Can you find a function providing such behavior (which might be named differently), or indicate that no such function exists?

```
// insert a column before the selected one
function doInsertColumnBefore(spreadsheet, view)
```

**REQUESTED BEHAVIOR**

```
/**
 * Executes a command, updating the Spreadsheet contents. Commands may include:
 * InsertColumnBefore, InsertRowBefore, DeleteColumn, DeleteRow, Paste.
 */

@param Spreadsheet spreadsheet, the spreadsheet document
@param View view, the current users view
@param String command, the command to execute

@return Spreadsheet
```

```
function doInsertColumnBefore(spreadsheet, view, command){
    switch(command){
        case 'insertColumnBefore':
            spreadsheet = doInsertColumnBefore()
            break
        default:
            //TODO: manage the other commands
            break
    }
    return spreadsheet;
}
```

**REQUESTING FUNCTION**

```
/**
 * Executes a command which modifies the View (but not the Spreadsheet itself), creating an updated user View of the Spreadsheet. Commands may include user actions such as Copy.
 */

@param Spreadsheet spreadsheet, the spreadsheet document
@param View view, the current users view
@param String command, the command to execute

@return View
```

```
function doViewCommand(spreadsheet, view, command)
```

**HINT**

Choose a function that provides the requested behavior (you can filter the list of functions by entering text in the input box). If there isn’t the right function, click check “no function found”.

```
function doViewCommand(spreadsheet, view, command)
```
WRITE A FUNCTION DESCRIPTION

A worker editing the function `doSpreadsheetCommand` requested that a function `doInsertColumnBefore` be created. Can you write a detailed description for the function `doSub`?

REQUESTED FUNCTION

```javascript
// insert a column before the selected one
function doInsertColumnBefore(spreadsheet, view)
```

REQUESTING FUNCTION

```javascript
/**
 * Executes a command, updating the Spreadsheet contents. Commands may include:
 * InsertColumnBefore, InsertRowBefore, DeleteColumn, DeleteRow, Paste.
 * @param Spreadsheet spreadsheet, the spreadsheet document
 * @param View view, the current user's view
 * @param String command, the command to execute
 */
return Spreadsheet
```

function doSpreadsheetCommand(spreadsheet, view, command){
    //TODO: validate parameters
    switch(command){
        case 'insertColumnBefore':
            //TODO: manage the other commands
            break;
        default:
            //TODO: manage the other commands
            return spreadsheet;
    }
The crowd has created a description for the function `doInsertColumnBefore`, called by the function below. Based on the description, can you check if the call(s) are correct, and revise them if necessary?

**Tip:** If you know a better way to implement the function, you may revise the function as you see fit.

```javascript
// Insert the view data inside the spreadsheet
@param Spreadsheet spreadsheet, the spreadsheet
@param View view, The view

return Spreadsheet
```

```javascript
// Executes a command, updating the Spreadsheet contents. Commands may include:
// InsertColumnBefore, InsertRowBefore, DeleteColumn, DeleteRow, Paste.
@param Spreadsheet spreadsheet, the spreadsheet document
@param View view, the current users view
@param String command, the command to execute

return Spreadsheet
```

```javascript
// Validate parameters

switch(command){
    case 'InsertColumnBefore':
        spreadsheet = doInsertColumnBefore(spreadsheet, view);
        break;
    default:
        //TODO: manage the other commands

return spreadsheet;
```
DEBUG A TEST FAILURE
15 pts

One of the tests for the function `doSpreadsheetCommand` has failed. Can you find and fix the bug (or report an issue with the test)?

**FAILING TEST**

- Given a spreadsheet and a given view, when an `InsertColumnBefore` command is received, a new column should be placed on the spreadsheet
  - executed in 2 ms - failed

**OTHER TESTS**

- Given a spreadsheet and a given view, when a null command is received, the spreadsheet does not change
  - executed in 2 ms - passed
- Given a spreadsheet and a given view, when a `DeleteRow` command is received, the indicated row should be deleted
  - executed in 1 ms - passed

---

Run the tests

**JAVASCRIPT TUTORIAL**

- **DEBUGGER TIPS**
  - `console.log()` to monitor statements
  - Click on the highlighted function calls for opening the stubs popup

**AVAILABLE DATA TYPES**

- String
- Number
- Boolean
- Spreadsheet
- Row
- Cell
- Style
- Position
- Range
- View

---

```
2  executes a command, updating the Spreadsheet content. Commands may include:
3  InsertColumnBefore, InsertRowBefore, DeleteColumn, DeleteRow, Paste.
4  
5  @param Spreadsheet spreadsheet, the Spreadsheet document.
6  @param View view, the current users view.
7  @param String command, the command to execute.
8  
9  return Spreadsheet;  
10  
11  function doSpreadsheetCommand(spreadsheet, view, command)
12   {  
13  if (command === null)  
14    return spreadsheet;  
15  
16  switch (command) {  
17  case 'InsertColumnBefore':  
18    return doInsertColumnBefore(spreadsheet, view);  
19  case 'InsertRowBefore':  
20    return doInsertRowBefore(spreadsheet, view);  
21  case 'DeleteColumn':  
22    return doDeleteColumn(spreadsheet, view);  
23  case 'DeleteRow':  
24    return doDeleteRow(spreadsheet, view);  
25  case 'Paste':  
26    return doPaste(spreadsheet, view);  
27  }  
28  //return window['do' + command](spreadsheet, view);  
29  ```
EVALUATION

▪ **RQ1:** Can developers complete automatically generated programming microtasks?
▪ **RQ2:** How quickly can developers onboard and begin making programming contributions?
▪ **RQ3:** Can programming contributions be made in a few minutes?
▪ **RQ4:** What are the effects of quality control mechanisms on the work produced?
CAN DEVELOPERS COMPLETE AUTOMATICALLY GENERATED PROGRAMMING MICROTASKS?

2 sessions:
- 1008 Microtasks
- 70 hours (44 hours Microtask working is submitted)
  - Review 37%
  - Write a test 22%
  - Edit a function 17%

<table>
<thead>
<tr>
<th>Microtask type</th>
<th>Completed</th>
<th>Skipped</th>
<th>Reissued</th>
<th>Median time (mm:ss)</th>
<th>Total time (hh:mm:ss)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>Review</td>
<td>260</td>
<td>227</td>
<td>22</td>
<td>22</td>
<td>1:27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1:14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9:29:32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6:43:43</td>
</tr>
<tr>
<td>Write Test</td>
<td>158</td>
<td>102</td>
<td>22</td>
<td>7</td>
<td>1:29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1:21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6:35:41</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3:15:12</td>
</tr>
<tr>
<td>Edit a Function</td>
<td>44</td>
<td>56</td>
<td>25</td>
<td>21</td>
<td>4:57</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2:31</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3:59:28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3:40:03</td>
</tr>
<tr>
<td>Write Test Cases</td>
<td>40</td>
<td>30</td>
<td>9</td>
<td>4</td>
<td>3:50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2:28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2:57:02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2:05:53</td>
</tr>
<tr>
<td>Debug Test Failure</td>
<td>14</td>
<td>18</td>
<td>5</td>
<td>6</td>
<td>2:32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4:00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0:57:22</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1:21:21</td>
</tr>
<tr>
<td>Write Function Description</td>
<td>8</td>
<td>16</td>
<td>3</td>
<td>0</td>
<td>3:12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2:44</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0:30:21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1:03:58</td>
</tr>
<tr>
<td>Write Call</td>
<td>7</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>1:37</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2:28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0:15:02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0:36:49</td>
</tr>
<tr>
<td>Reuse Search</td>
<td>9</td>
<td>10</td>
<td>3</td>
<td>0</td>
<td>0:42</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1:35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0:06:37</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0:22:33</td>
</tr>
<tr>
<td>Total</td>
<td>540</td>
<td>468</td>
<td>91</td>
<td>62</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>96</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24:51:05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19:09:32</td>
</tr>
<tr>
<td>Overall total</td>
<td>1008</td>
<td>153</td>
<td>164</td>
<td></td>
<td>44:00:37</td>
</tr>
</tbody>
</table>
While spending time to be familiarized with CrowdCode:

- Skipped the task a lot
- First task submitted after 14 minutes

2nd task: Between 1:31 – 5:29

After gaining experience:
CAN PROGRAMMING CONTRIBUTIONS BE MADE IN A FEW MINUTES?

Larger Completion time for larger Microtasks (4:57)
  Edit a function
  Write test cases
  DebugTestFailure

Shorter Microtasks below 2 minutes:
  Reuse search
  Write test
  Review

Skipped 13% -> 17% of which is automatic system skip
WHAT ARE THE EFFECTS OF QUALITY CONTROL MECHANISMS ON THE WORK PRODUCED?
WHAT ARE THE EFFECTS OF QUALITY CONTROL MECHANISMS ON THE WORK PRODUCED?

```javascript
function createAction(mouseDownPos, command, elements) {
  switch (command) {
    case 'Move':
      var el = null;
      for (el in elements) {
        if (isOutline(mouseDownPos, el)) {
          return;
        }
        *type* : 'Move',
        *elementId* : el,
        *mouseDownPos* : mouseDownPos
      }
    break;
    case 'Line':
      var nextEl = getNextElement(elements);
      if (command == 'Line') {
        // handle Line
        // handle Frehand
        // handle Rectangle
      } else if (command == 'Frehand') {
        return null;
      }
      return null;
    break;
    // handle Frehand and return it
    // handle Rectangle and return it
  }
  return null;
}
```
WHAT ARE THE EFFECTS OF QUALITY CONTROL MECHANISMS ON THE WORK PRODUCED?

- To identify and correct the defects, participants used test and review systems
  
  On average 6.8 tests per function
  
  If tests fail -> Debug Test Failure Microtask generated

- 37% of time is spent on Review Microtasks:

  "You can learn from your mistakes that others point out, or better ways to accomplish the same goal of a certain function".
OPEN QUESTIONS

- Overall reaction to the paper
- Are you willing to work on CrowdCode?
- Are you able to work on a 10 minutes time task? Don’t you feel rush?
- How to control the code quality and multiple code hand written?
- How to scale between Senior developers and Novices?