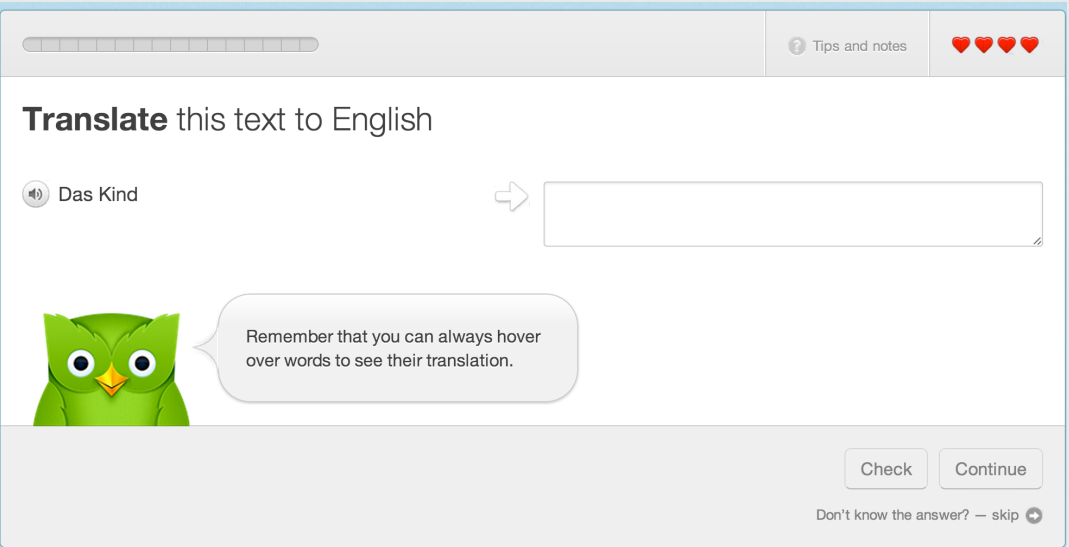


Crowd Development

Thomas D. LaToza¹, W. Ben Towne², André van der Hoek¹, and James D. Herbsleb²
¹ University of California, Irvine ² Carnegie Mellon University

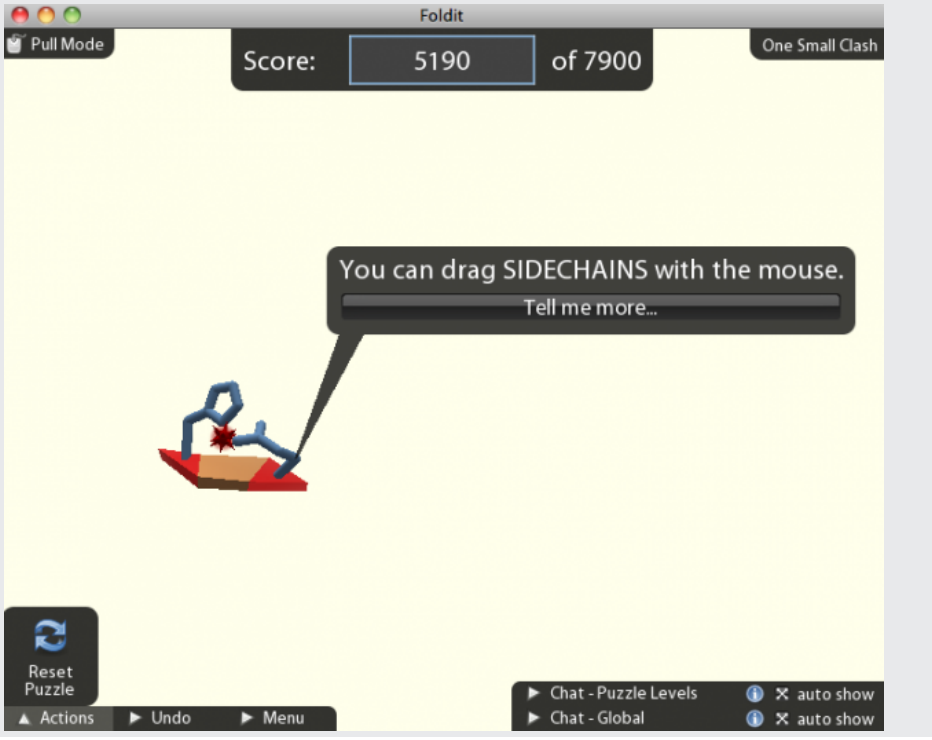


Translate the web... by learning a language





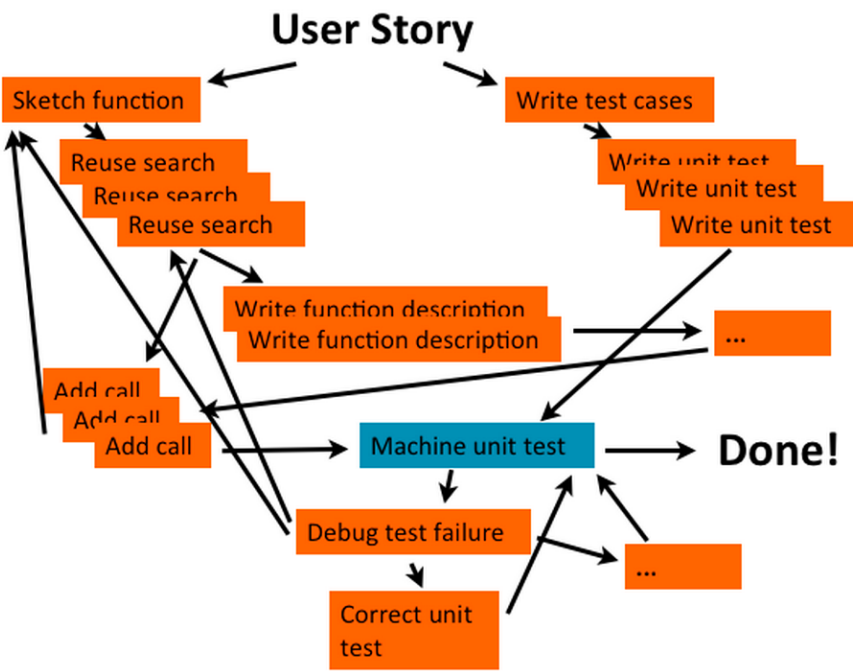
Non-expert players solved a decade long hard research problem in 10 days.



CrowdCode

Build software with a crowd!

CrowdCode organizes work into **microtasks**, small, self-describing bits like writing pseudocode or brainstorming test cases. After you finish a microtask, CrowdCode figures out what to do next, generating and distributing microtasks to the **crowd**. So you might write a description for function one, debug a test failure for another, and then edit the pseudocode the crowd wrote for function one to add a call. As you complete microtasks, you earn **points**, and can see how you're doing on the leaderboard.



Let's get started!

CrowdCode

8 lines of code 2 functions written 14 microtasks completed test@example.com

Your score ★

60 points

Leaders

80 Bob
60 test@example.com

Ask the Crowd

test@example.com are numbers supposed to be ints or can they be other values like doubles?
Bob all numbers should just be ints

Edit a function 10 pts

Can you figure out how this user story should be implemented?

Add two numbers together, returning the sum.

The main function - the entrypoint into the application - is below. Sketch a design of this user story by editing the function's description (the comments above the function header) and sketching an implementation. Note that you should NOT implement everything in main, but instead use pseudocalls (see below) to ask the crowd to create new functions or reuse existing functionality. Try not to break other user stories that may already be implemented. But don't worry too much - it'll all be tested.

If you're not yet exactly sure how to do something, indicate a line or portion of a line as **pseudocode** by beginning it with `///. If you'd like to call a function, describe what you'd like it to do with a pseudocall - a line or portion of a line beginning with ///. Update the description and header to reflect the function's actual behavior - the crowd will refactor callers and tests to match the new behavior. (Except if you are editing the function "main" - you can't change this function's name or number of parameters, but you can still change its description).`

```
1 /**  
2  [INSERT A DESCRIPTION OF THE FUNCTION HERE!]  
3  
4  Describe the purpose and intent of the function.  
5  List each parameter, describing its structure (what fields it has)  
6  and it's intent. For example, if you had a function that took  
7  a param named parsedSentence that looked like  
8  { sentence: [ 'Hello', 'world!' ] } and returns a  
9  boolean, you might describe its params and return as follows:  
10  
11  @param { sentence: [ strings ] } parsedSentence - sentence parsed  
12  into an array of words  
13  @return boolean - whether something is true about the sentence  
14  */  
15 function main(input)  
16 {  
17   return input;  
18 }
```

Recent Activity

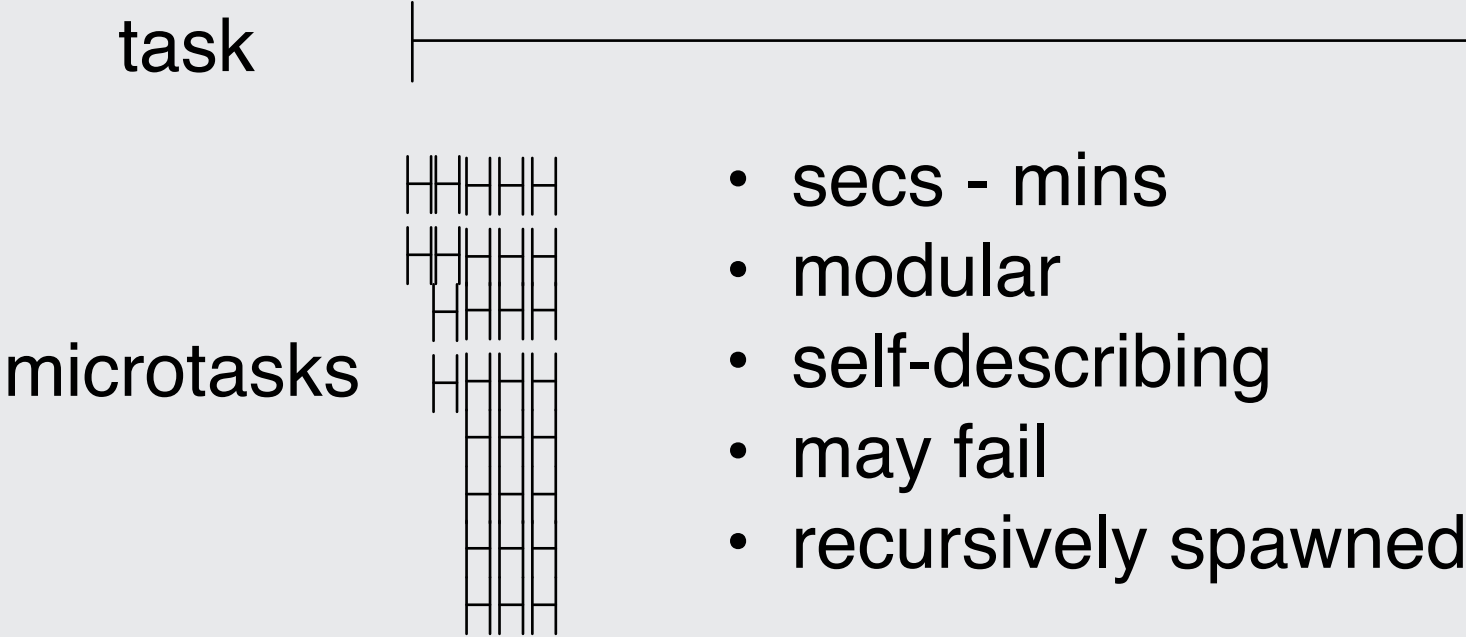
- ★ You earned 10 points for adding a call!
- ★ You earned 10 points for describing a function!
- ★ You earned 10 points for writing a test!
- ★ You earned 10 points for conducting a reuse search!
- ★ You earned 10 points for writing test cases!
- ★ You earned 10 points for editing a function!

Give us feedback on CrowdCode! What do you like? What don't you like?

Send feedback

What if a large application could be built in a day?

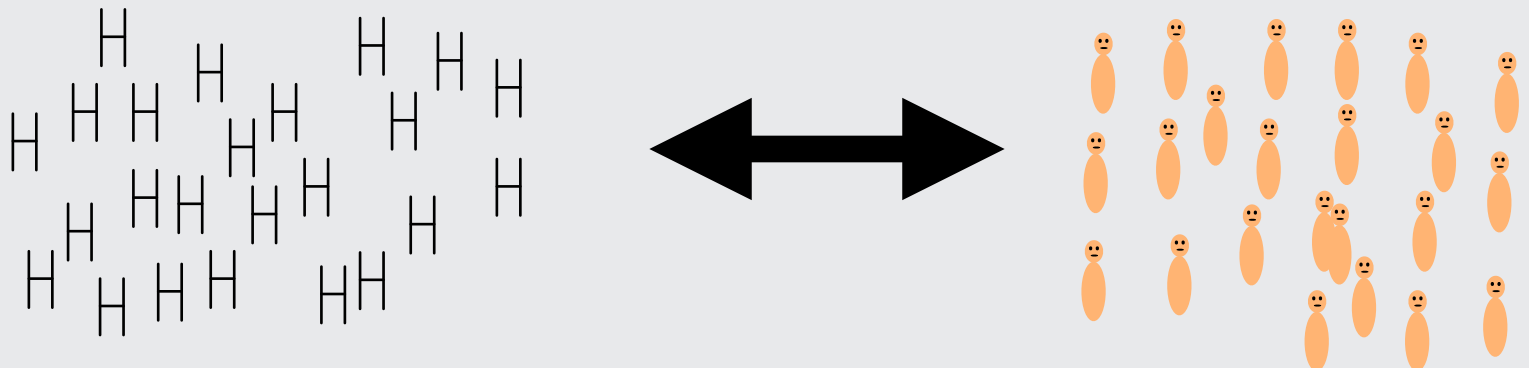
Increasing **parallelism** reduces time to market



What if grandma or grandpa's hobby was writing software?

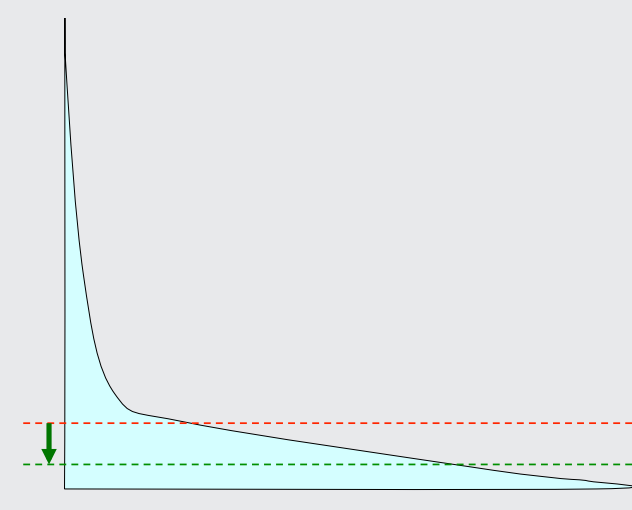
- reducing range of knowledge required enables **specialization** through editors or in expertise
- some artifacts edited by EUP w/ EUP tools
 - expert architect contributes for a few microtasks
 - workers become expert in sort routines

How can work be effectively allocated to a crowd?



- worker** characteristics, **task** characteristics
- experience w/ relevant artifacts
 - skill in type of programming
 - reputation / trust
 - overall knowledge of system
- best available match vs. wait for better match

What if you could navigate to a site and start contributing?



- capture the **"long tail"** of contribution by lowering joining costs
- just **start** working
 - **stop** when you're bored
 - **skip** what you don't want to do

What if software development felt like a game?

- optimal** challenge + **social** might make building software more fun & educational
- earn as many **points** as you can
 - **level** up to harder work
 - cash in points for **prizes**
 - watch how your **friends** are doing
 - compete in pair programming

Code
with
Friends

How can a crowd create a design with conceptual integrity?

- push & pull information over the **dependency** graph
- iterative critique - many solutions, **critique**, recombine, iterate
- collective decision making - **StackOverflow** for a project

How do you create high-quality software with transient, potentially malicious workers of varying expertise?

- redundancy, reviews, reputation
- apportion **value created** back to contributors
- function reused
 - function passes all its tests
 - user story completed quickly