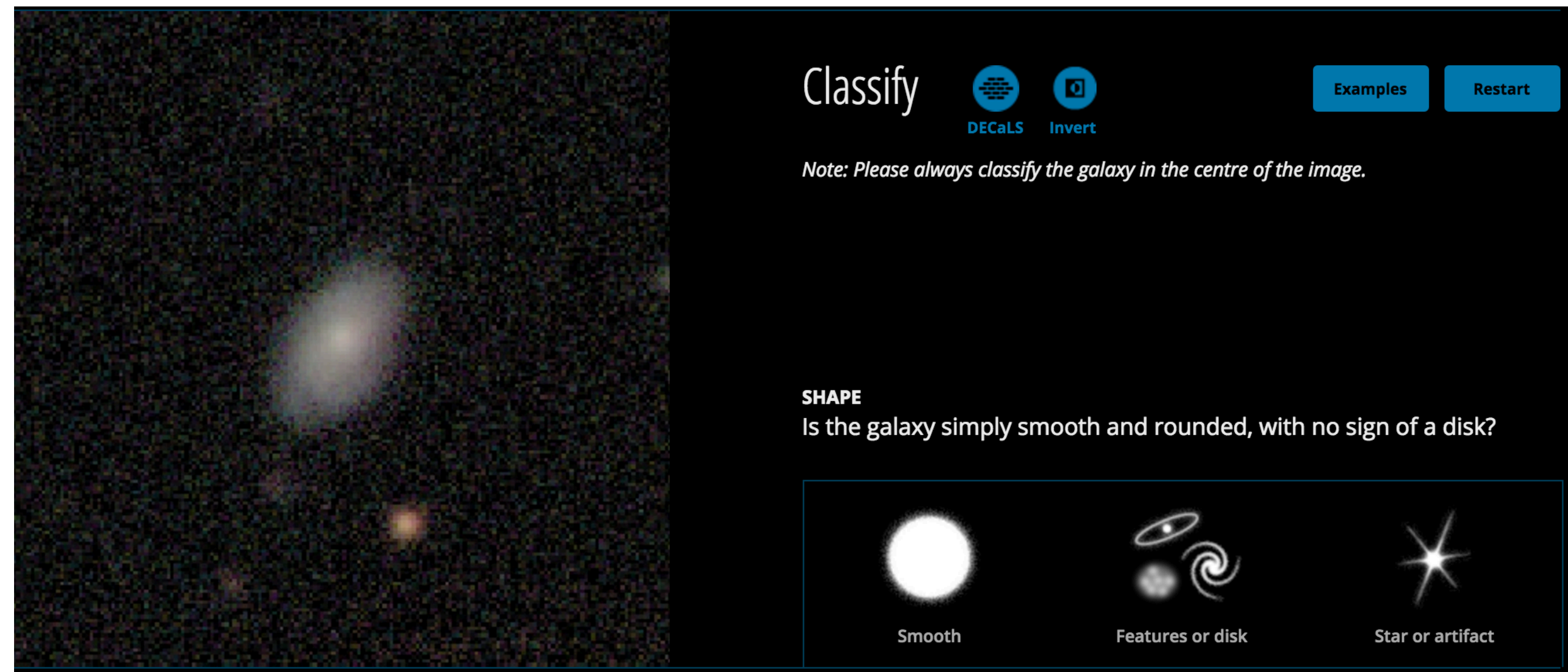


CrowdForge: Crowdsourcing Complex Work

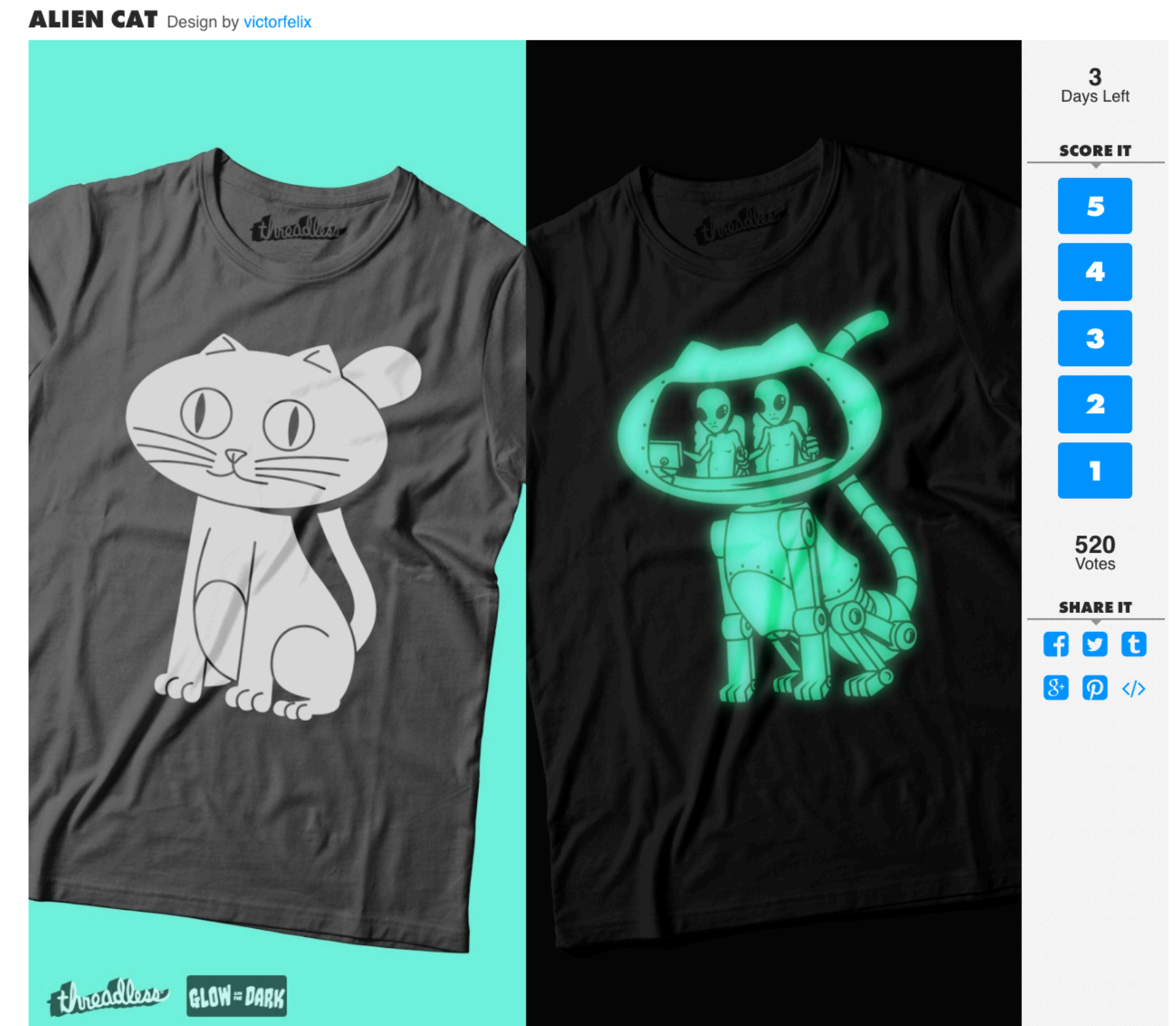
Han Bao, Master of Big Data



BACKGROUND



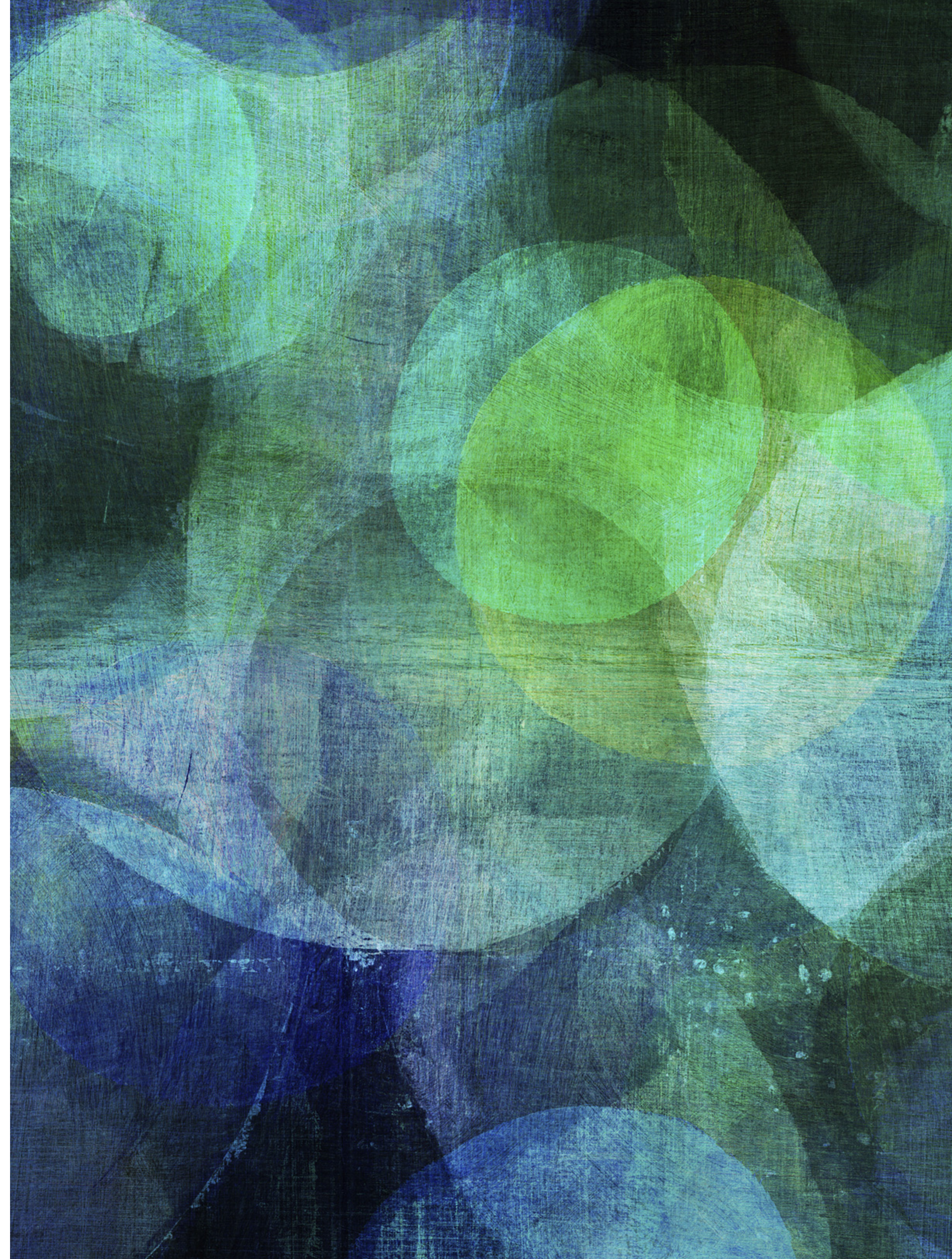
Discovering new galaxies: galaxyzoo.org



Crowdsourcing t-shirt designs: Threadless

Micro-task markets such as Amazon's Mechanical Turk have been primarily used for simple, independent tasks.

Here we present a general purpose framework for **accomplishing complex and inter-dependent tasks** using micro-task markets. We describe our framework, **a web-based prototype**, and **case studies** on article writing, decision making, and science journalism that demonstrate the benefits and limitations of the approach.



INTRODUCTION

- The types of tasks accomplished through MTurk have typically been limited to those that are low in complexity, independent, and require little time and cognitive effort to complete.
- In contrast to the typical tasks posted on Mechanical Turk, much of the works in many real-world work require substantially more coordination among co-workers than do the simple, independent tasks seen on micro-task markets.

EXAMPLE:

- Writing a short article about a locale, a newspaper, a travel guide, or a corporate retreat or an encyclopedia.

Deciding on the scope and structure of the article

Finding and collecting relevant information

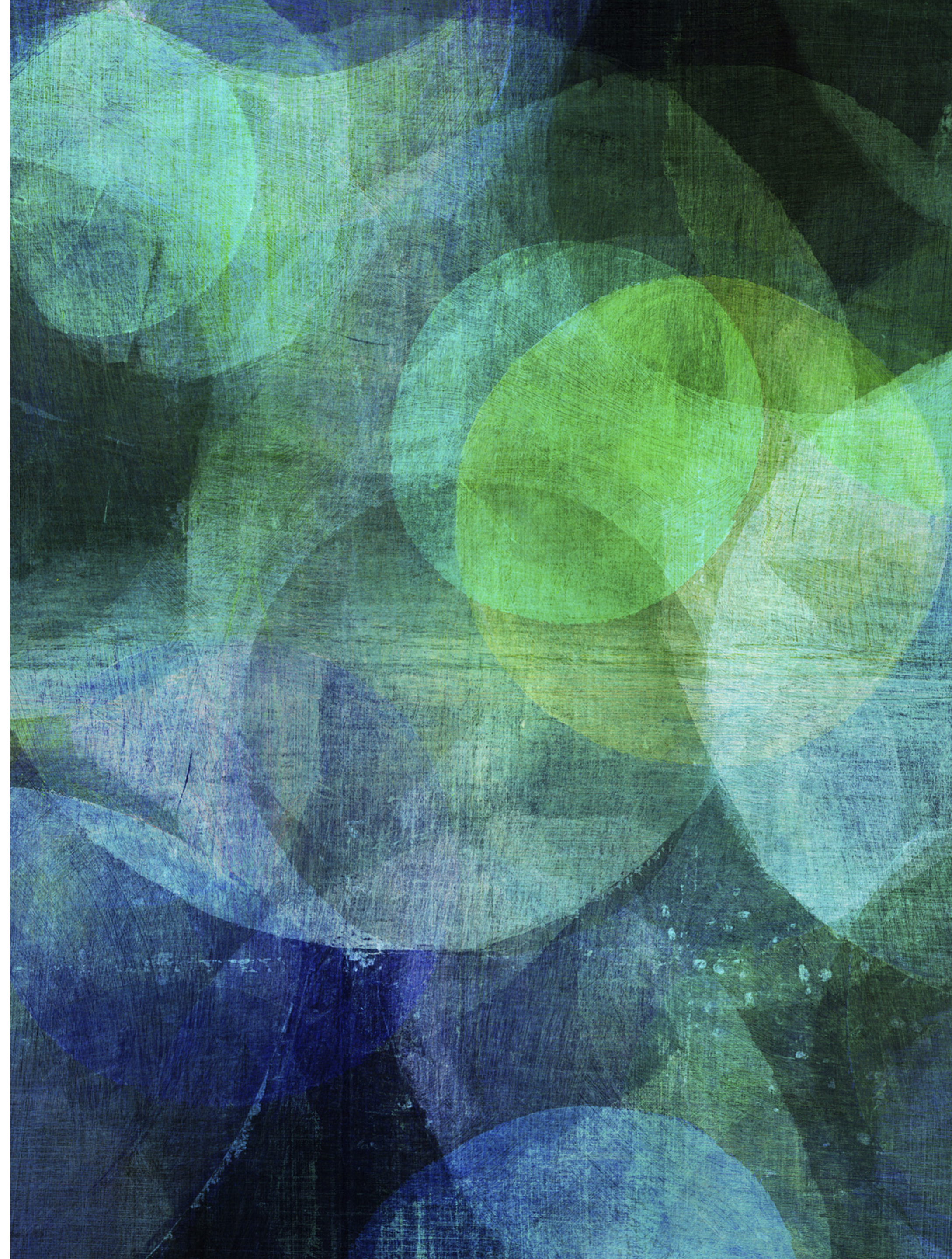
Writing the narrative

Taking pictures

Laying out the document and editing the final copy

Coordinate in order to avoid redundant work and to make the final product coherent.

*Present the CrowdForge framework and toolkit
for crowdsourcing complex work*



MECHANICAL TURK

- MTurk encompasses a large and heterogeneous pool of tasks and workers; Amazon claims over 100,000 workers in 100 different countries, and as of the time of writing there were approximately 80,000 tasks available.
- **APPROACH**

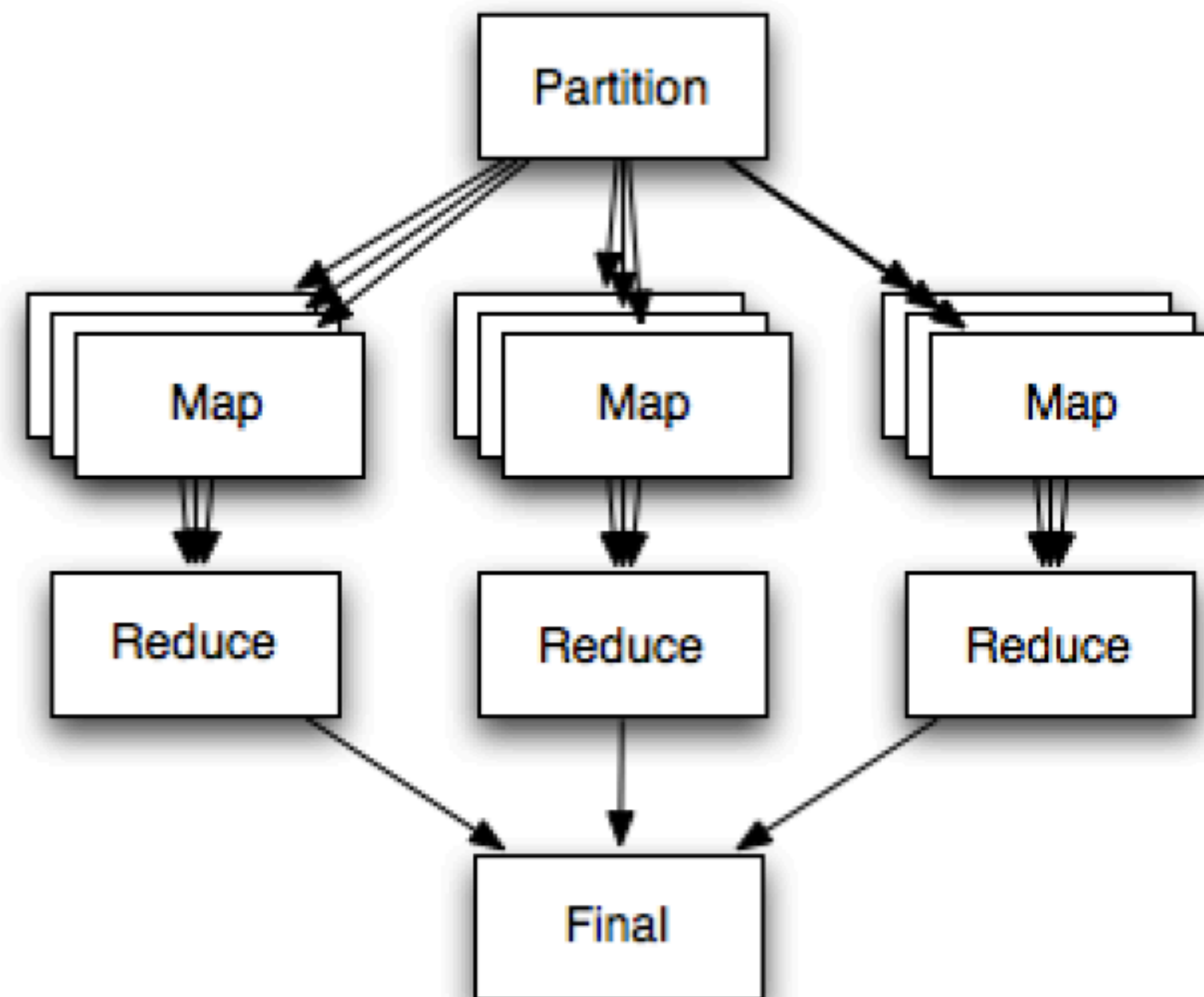


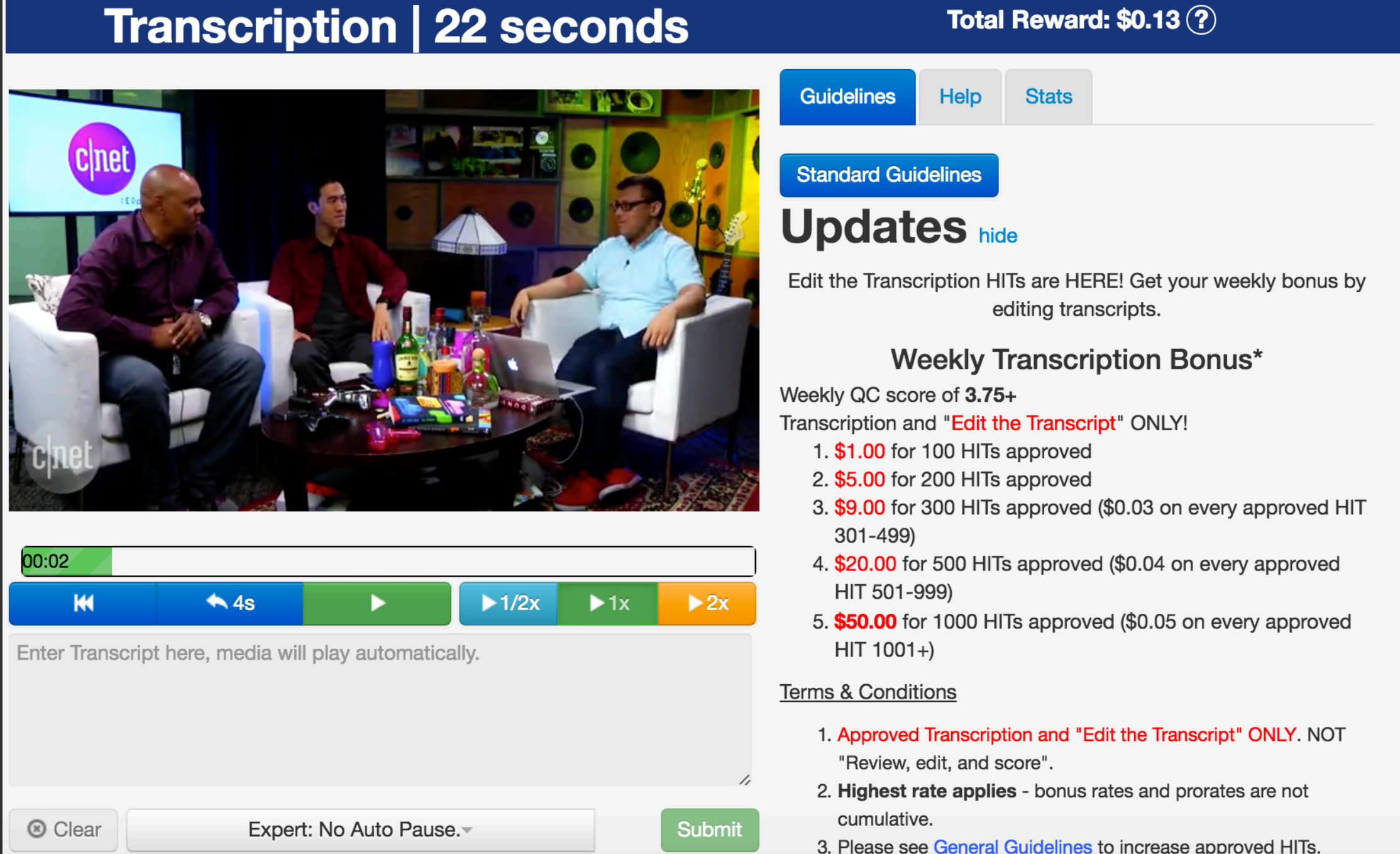
Figure 1: Overview of framework for splitting up and recombining complex human computation tasks.

APPROACH

How to become a Freelance Transcriber for CastingWords

We post all of all transcriptions on Amazon's Mechanical Turk (mTurk). We usually have a good variety of jobs posted broken up into 4-10 minute chunks. You accept one of these, transcribe the file, then submit the text to us via mTurk, we receive it and review it and approve it.

- The disaggregation of an audio file into smaller transcription tasks and the use of a second wave of workers to verify the work done by the transcriptions. The results of multiple workers' outputs are voted on and the best sent to new workers, whose work is then voted on, and so force.
- The transcription task can be broken up into the following elements:
 - breaks up the audio into smaller subtasks
 - A flow that controls the sequencing of the tasks
 - A quality control phase
 - Automatic aggregation of the results



Transcription | 22 seconds Total Reward: \$0.13 ?

[Guidelines](#) [Help](#) [Stats](#)

[Standard Guidelines](#)

Updates [hide](#)

Edit the Transcription HITs are HERE! Get your weekly bonus by editing transcripts.

Weekly Transcription Bonus*

Weekly QC score of 3.75+

Transcription and "Edit the Transcript" ONLY!

1. **\$1.00** for 100 HITs approved
2. **\$5.00** for 200 HITs approved
3. **\$9.00** for 300 HITs approved (\$0.03 on every approved HIT 301-499)
4. **\$20.00** for 500 HITs approved (\$0.04 on every approved HIT 501-999)
5. **\$50.00** for 1000 HITs approved (\$0.05 on every approved HIT 1001+)

[Terms & Conditions](#)

1. **Approved Transcription and "Edit the Transcript" ONLY.** NOT "Review, edit, and score".
2. **Highest rate applies** - bonus rates and prorates are not cumulative.
3. Please see [General Guidelines](#) to increase approved HITs.

APPROACH

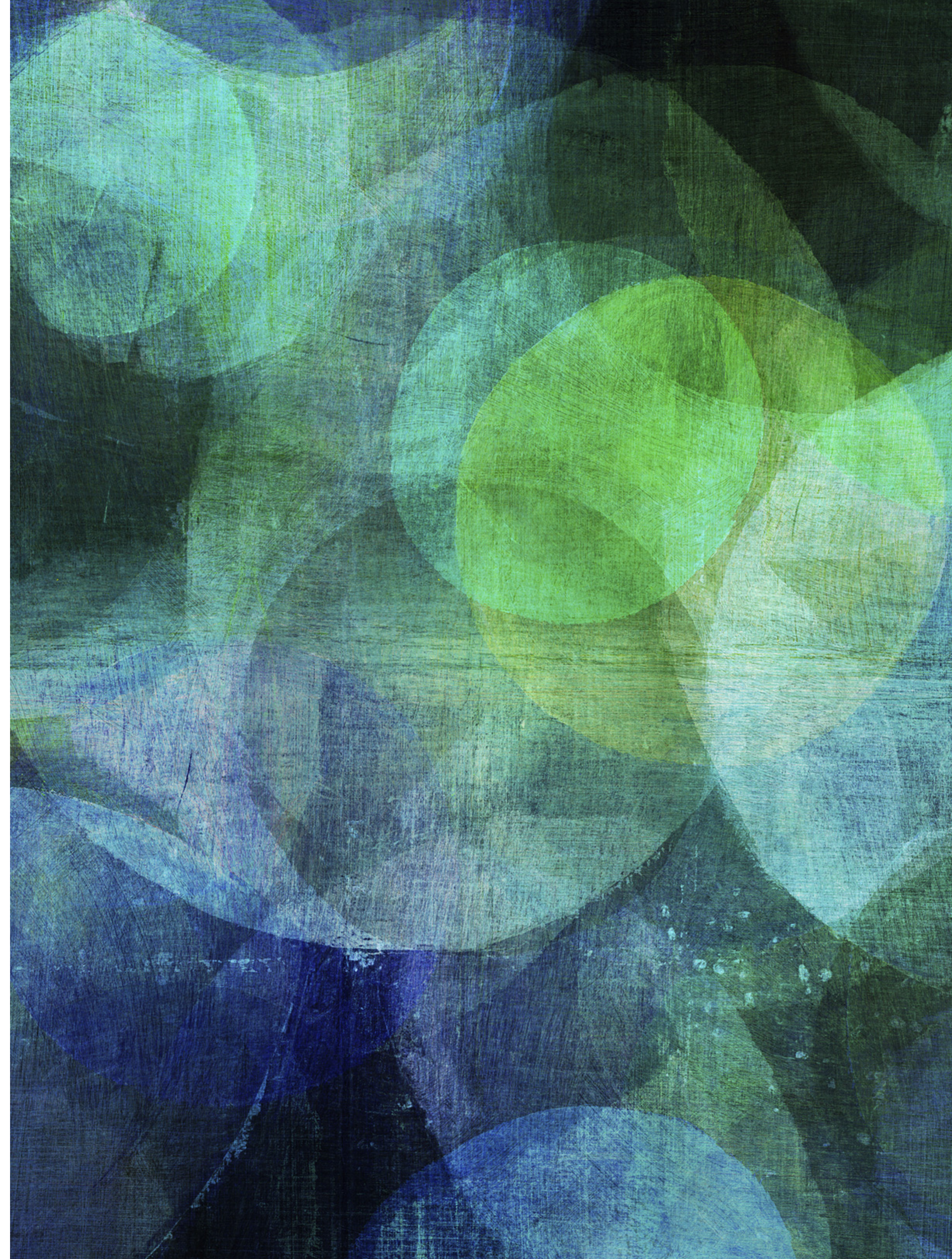
- Specifically, our framework aims to support:
 - **Dynamic partitioning** so that workers themselves can decide a task partition, with their results in turn generating new subtasks (rather than the task designer needing to fully specify partitions beforehand)
 - **Multi-level partitions** in which a task can be broken up by more than one partition
 - **Complex flows** involving many tasks and workers
 - **A variety of quality control methods** including voting, verification, or merging items
 - **Intelligent aggregation of results** both automatically and by workers.
 - A simple method for **specifying and managing tasks and flows** between tasks

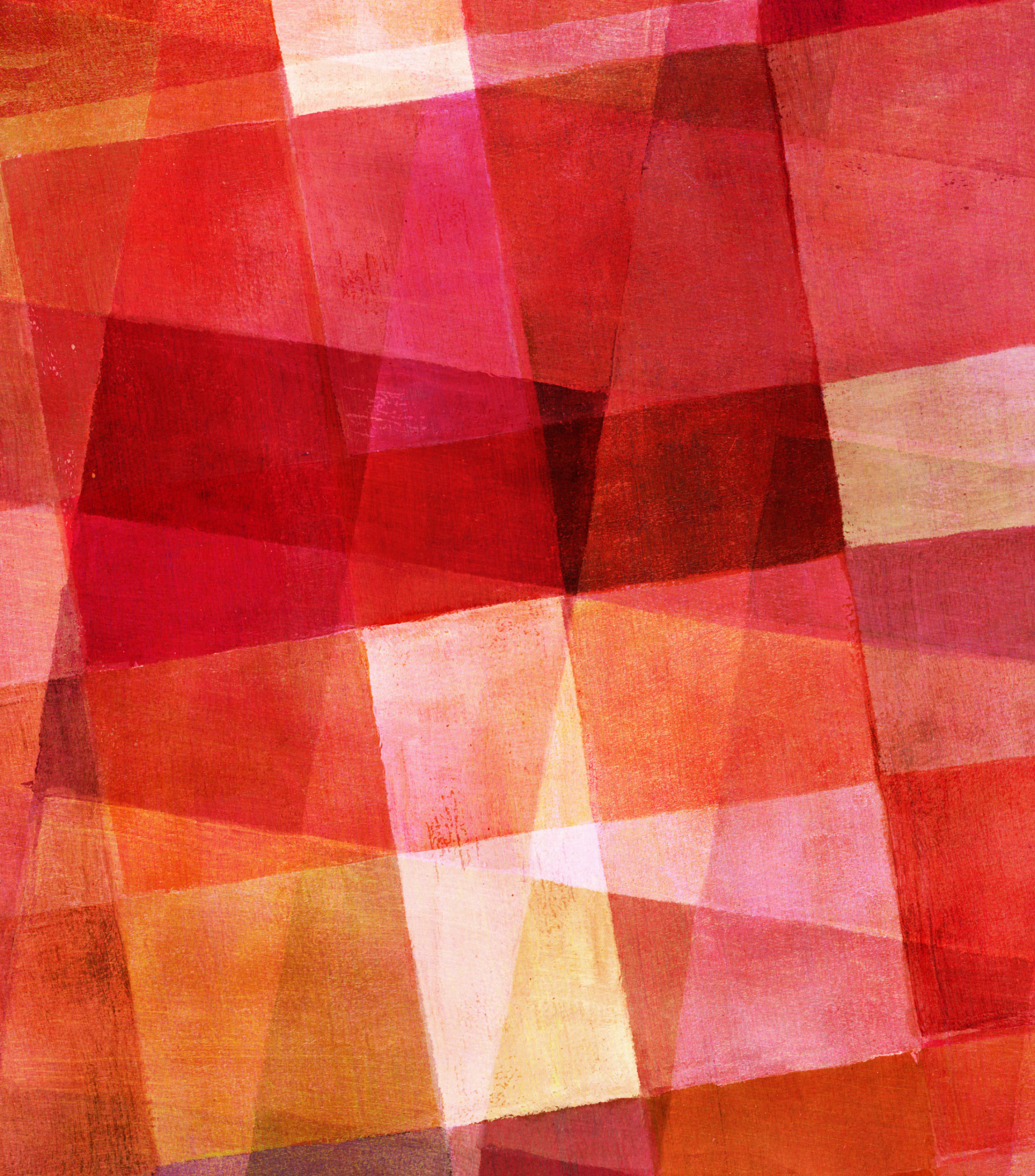
APPROACH

- MapReduce was inspired by functional programming languages in which a large array of data is processed in parallel through a two step process: first, key/value pairs are each processed to generate a set of intermediate key/value pairs (the Map phase). Next, values with identical intermediate keys are merged (the Reduce phase).
- **Partition tasks**, in which a larger task is broken down into discrete subtasks
- **Map tasks**, in which a specified task is processed by one or more workers
- **Reduce tasks**, in which the results of multiple workers' tasks are merged into a single output

Case studies:

- *Article writing*
- *Researching a purchase*
- *Crowdsourcing Science Journalism*





Article writing

ARTICLE WRITING: PROBLEM

- Individuals may not be willing to spend the large amount of effort needed to be a leader and may not be able to **communicate with** others in order to coordinate or influence their behavior.
- Many of the subtasks involved, such as **assembling the relevant information** or doing the actual writing, can be **time consuming and complex**.

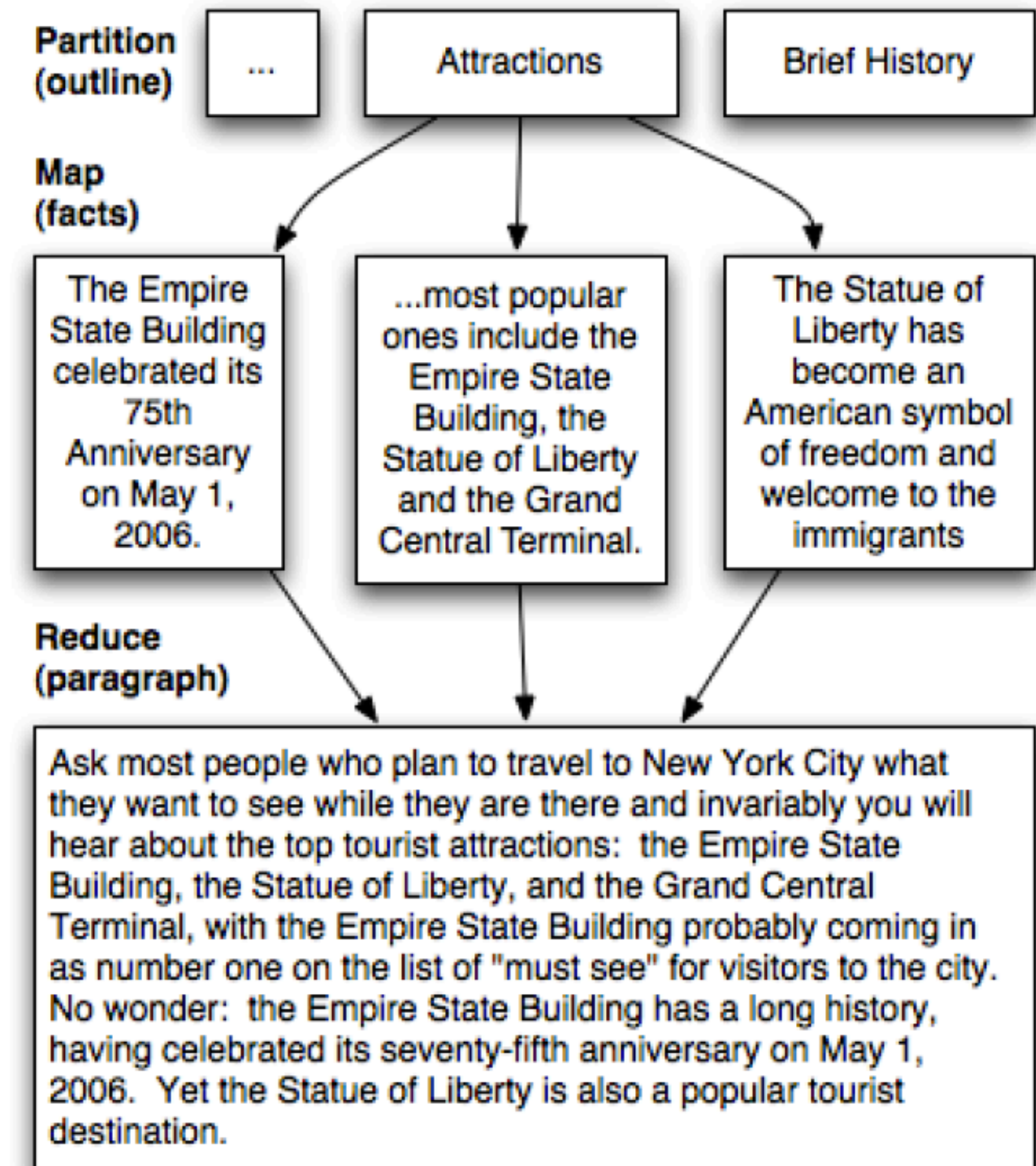


Figure 2: Partial results of a collaborative writing task.

ARTICLE WRITING

- We used this approach to create five articles about New York City. Articles cost an average of \$3.26 to produce, and required an average of 36 subtasks or HITs, each performed by an individual worker.
- A fragment of a typical article is shown in Figure 2. This article consisted of 955 words and 7 sections. It was completed via 36 different HITs for a total cost of \$3.15.

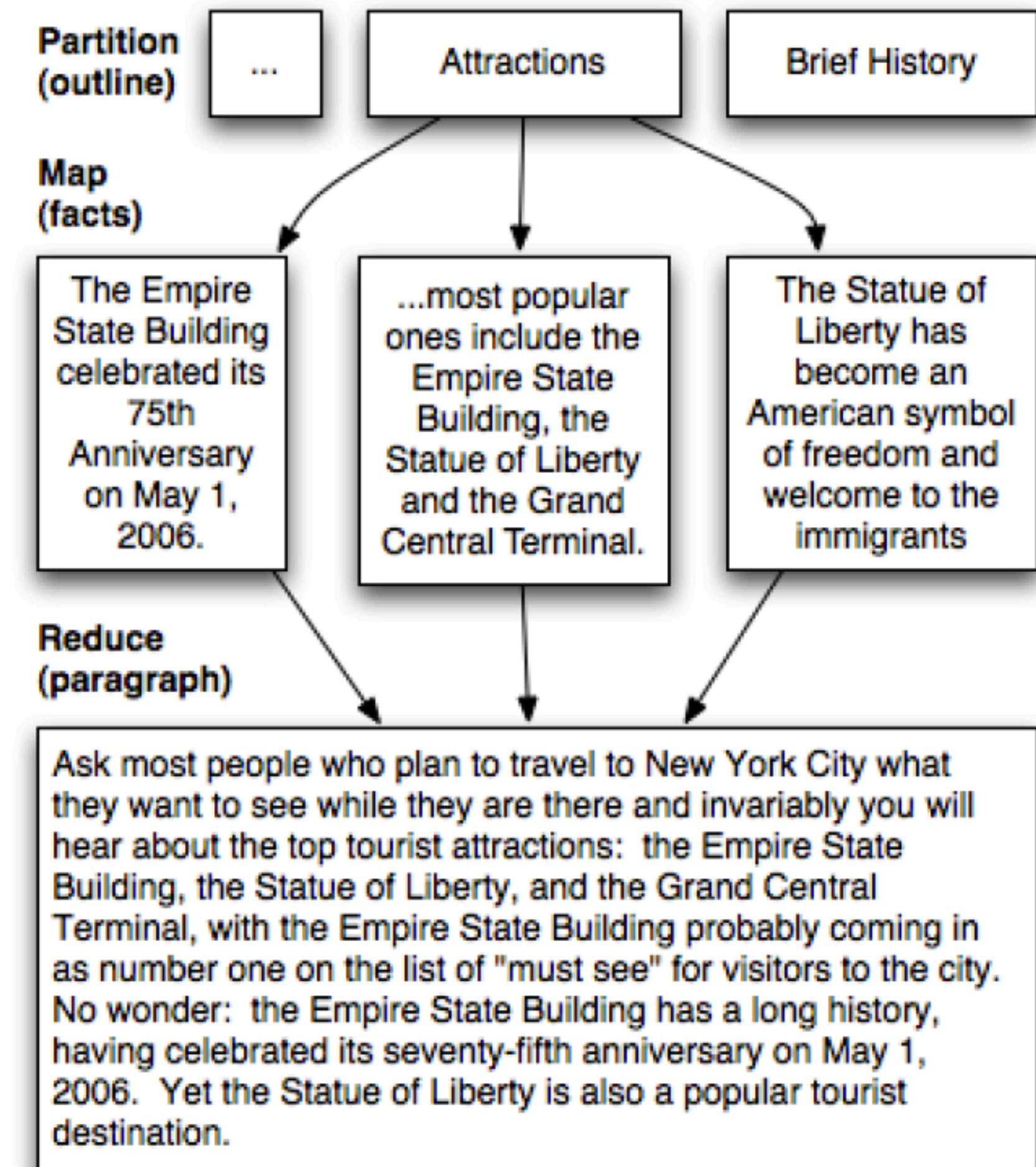


Figure 2: Partial results of a collaborative writing task.

ARTICLE WRITING

- To verify the quality of these collaboratively written articles, we compared them to **articles written individually** by workers and to the **entry from the Simple English Wikipedia** on New York City.
- Individual source is the **same** as the **group payment**.
- The resulting articles consisted of an average of **393 words**, approximately **60% the length** of the collaborative written articles.

ARTICLE WRITING

- We then evaluated the quality of all articles by asking a new set of workers to each rate a single article
- On average the articles produced by the group were of higher quality than those produced individually: mean quality for group-written articles = 4.01 versus 3.75 for individually-written ones (Wikipedia quality=3.95)
- The variability of group writing was lower as well.

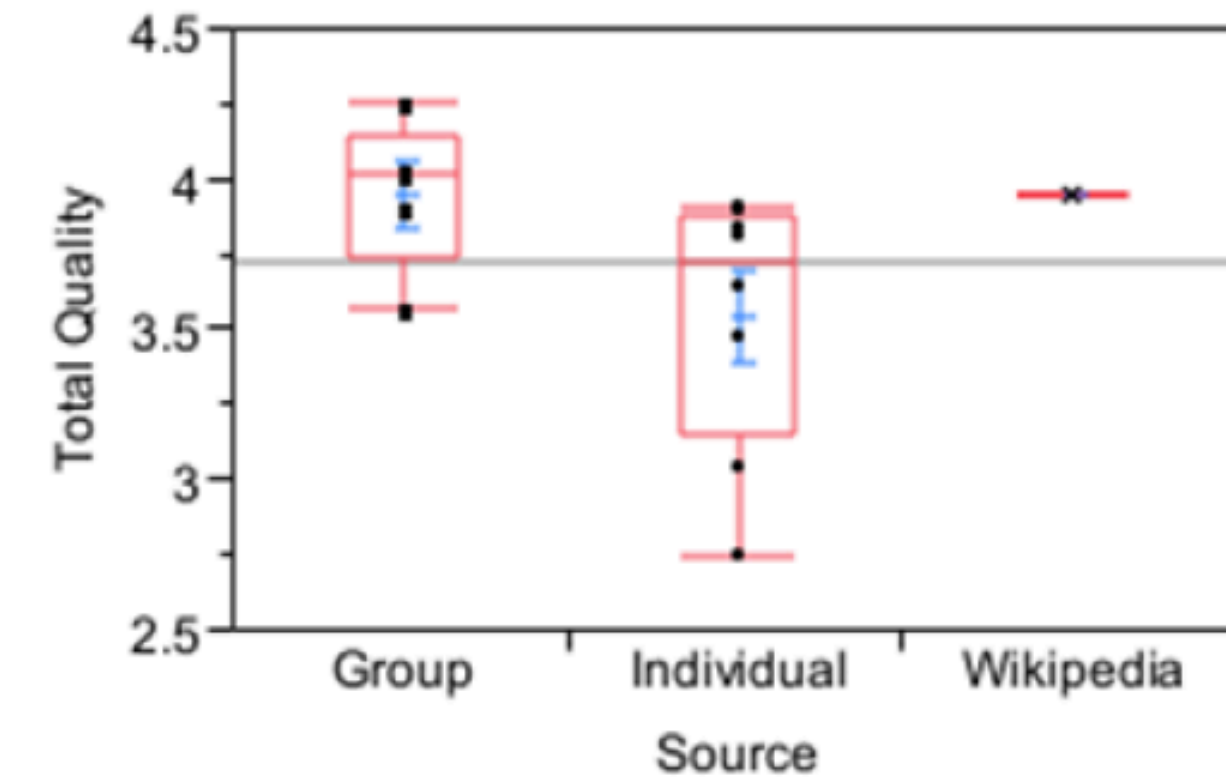


Figure 3: Rated quality of articles about New York City produced by Mechanical Turk workers acting individually or as a group using our framework compared to the quality of the same article on the Simple English Wikipedia.

QUALITY CONTROL

- This creates the possibility that a **single bad partition could have a large negative effect** on the whole task.
- Our approach to dealing with this challenge is to utilize additional Map or Reduce tasks to supporting fault tolerance and quality control.
 - Workers verify the result of other workers' outputs can be represent as a Map task that applies a verification function to each value.
 - Voting on the best choice can be represented as a Reduce task in which a single is chosen from multiple workers' outputs based on vote.

Whether more complex quality control methods would work better than methods such as simple voting ???

- To test these hypotheses we ran an experiment on quality control of article outlines. In the first phase we asked **20 workers to each independently** generate an outline for an article on the recent Gulf of Mexico oil spill using the same procedure as in the article writing case study above.
- We then took these 20 outlines and randomly assigned them to **20 different sets of three outlines**.
- For evaluation we crossed the 20 initial and 20 merged outlines, and asked workers to choose which outline would result in a better article.

Whether more complex quality control methods would work better than methods such as simple voting ???

- Merged outlines were rated higher than the initial outlines: 61% of merged outlines were chosen compared to 39% of initial outlines.
- Merged outlines also had fewer poor outlines.
- The best merged outlines were considered better than the best initial outlines.

Conclusion: Complex quality control tasks can be more effective than simple voting!

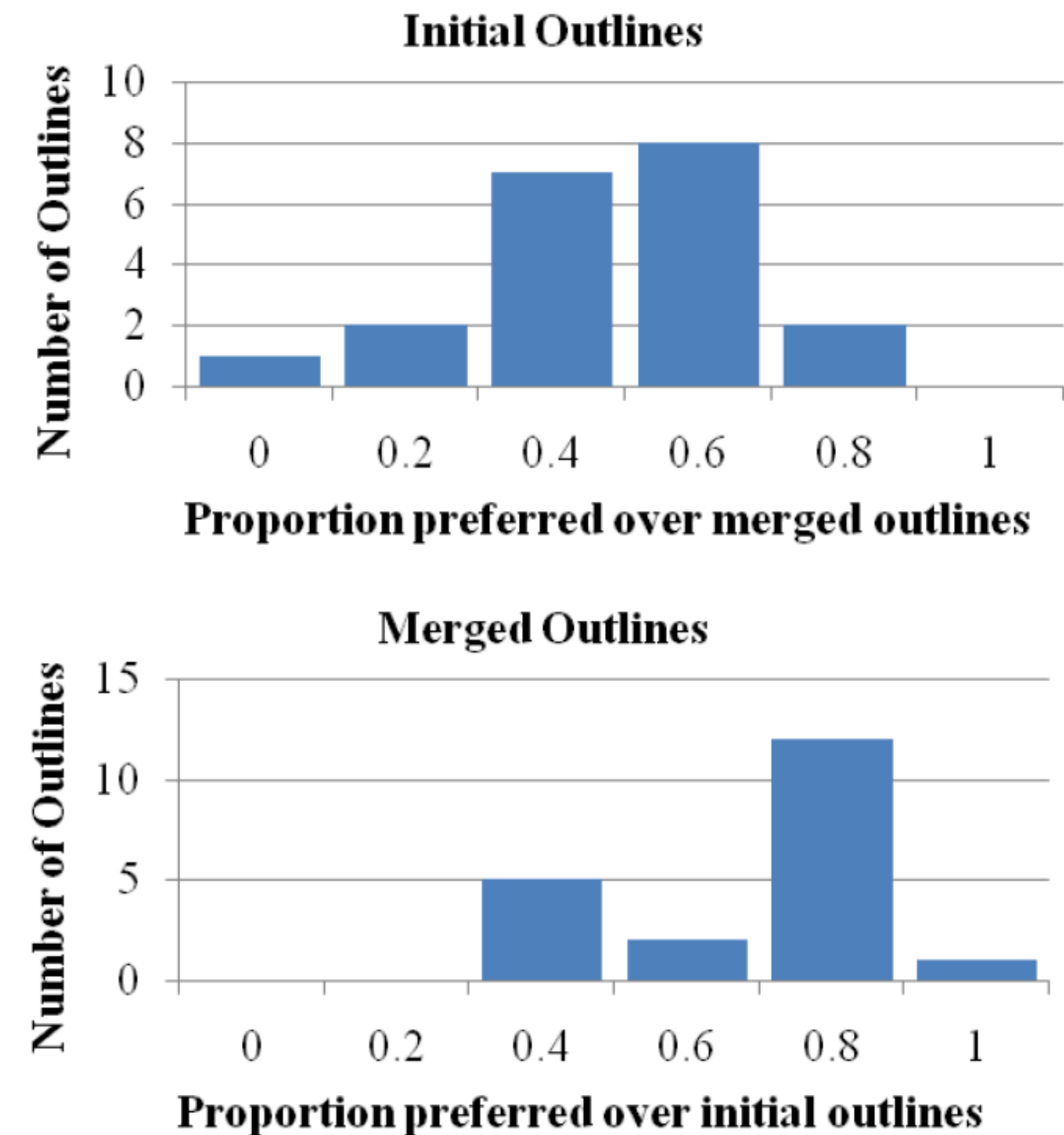
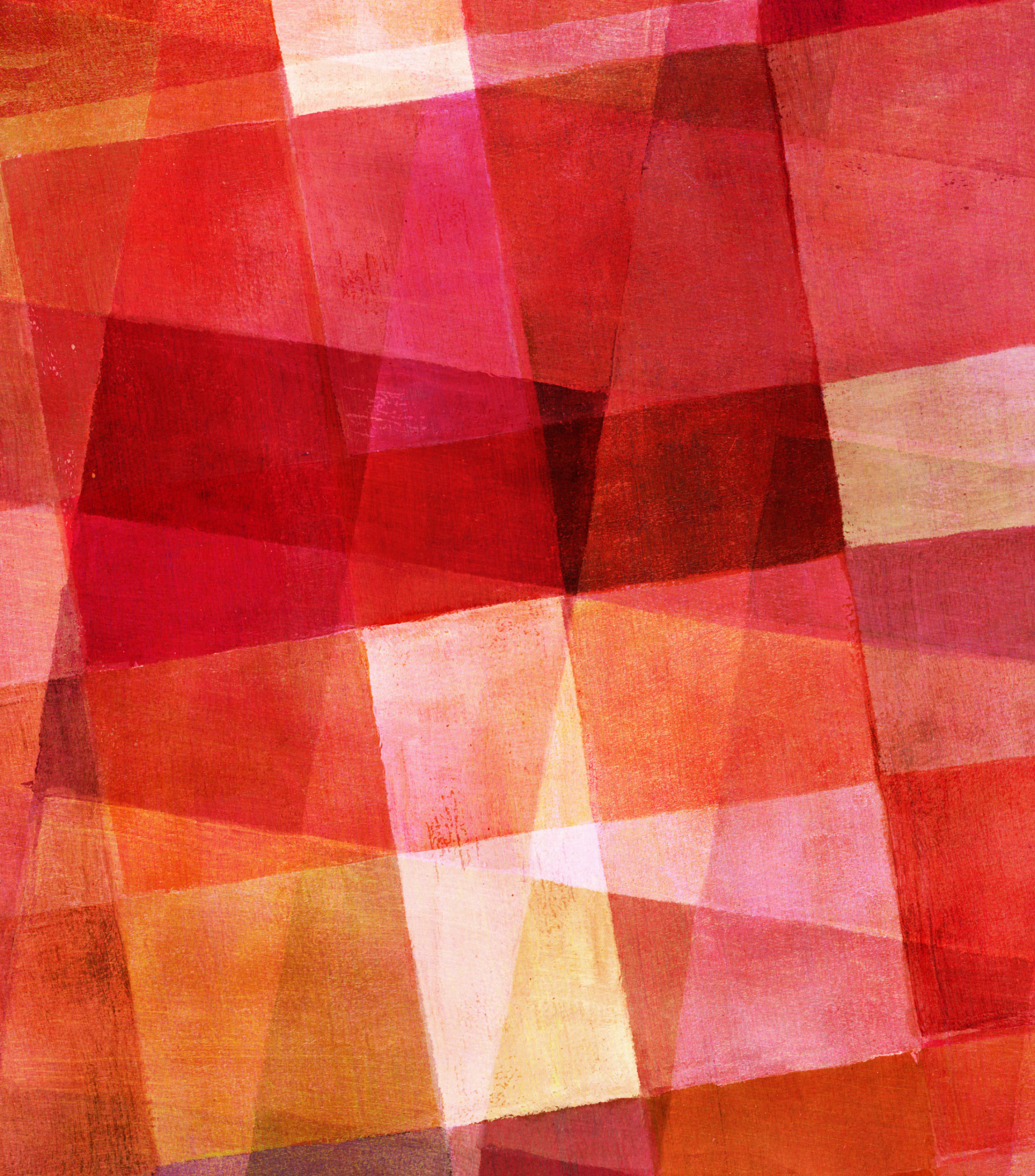


Figure 4. Histograms of participants preference choices for initial and merged outlines.



Researching a purchase

RESEARCHING A PURCHASE

We applied our framework to commission decision matrices.

- This example extends the framework by **showing how one can partition the initial task on multiple dimensions.**
 - In the partition HIT for this problem, one worker was given a short description of a consumer and asked to submit **criteria they would evaluate a car on.**
 - Another worker was given the same description and asked to submit a list of **potential competitors.**
- Combining the resulting lists yielded a matrix resembling a **product comparison table.**
 - In the map step, workers were asked to submit facts for one cell in the table
 - Finally, in the reduce step workers were given all the facts for a cell collected by workers in the map step

	Honda Odyssey	Ford Escape	Nissan Pathfinder
Safety Rating	Honda's Odyssey scored 5 stars for front and side impact in tests conducted by the NHTSI, and features anti-lock brakes for added stability in turns when braking.	Ford's Escape model scored high marks in front and side crash testing, with a safety rating of 9.9 and pretensioner seatbelt feature that tightens automatically in a crash.	Nissan's Pathfinder gets high marks for safety in NHTSI tests and scored 4 stars for front crash test and is offered in a couple different body styles.

Figure 5: An excerpt from the product comparison table

PROTOTYPE

- We implemented a software prototype to test our approach by allowing task designers to indirectly use MTurk to solve complex problems. The prototype allows task designers to **break complex problems down into sub-problems**, to specify the **relationship** between the sub-problems, and to **generate a solution using MTurk**.
- The system abstracts the entire process as a *problem*, which **tracks the state of the current complex task**. A problem references multiple *HIT Templates* and a *flow* that defines the dependencies between the HIT Templates.

Home > Mapreduce > Problems > Add problem

Add problem

Name:	<input type="text" value="Car Buying"/>
Step:	<input type="button" value="Start"/>
Partition:	<input type="text" value="Come up with criteria for buying a new car"/> +
Partition2:	<input type="text" value="Come up with a list of car makes and models"/> +
Mapper:	<input type="text" value="Collect one fact about the %(0)s of the %(1)s"/> +
Reducer:	<input type="text" value="Write a sentence using given facts"/> +

Figure 6: Creating a problem with the web user interface.

NOTIFICATION-BASED FLOW CONTROL MECHANISM

- When a user creates a new problem, they **specify which flow to use to solve that problem**. Flows are implemented as python classes:
 - `on_stage_completed(self, stage)`
 - `crowdforge.flows.Flow`
 - `crowdforge.flows.register`
 - `SimpleFlow`
- The system uses a **notification-based flow control mechanism** to manage which tasks and templates are posted. Every few minutes the system monitors **active problems for four kinds of events**, and fires notifications as needed.
 - *re-sult retrieved*
 - *HIT expired*
 - *HIT complete*
 - *stage complete*

COMPLEX FLOW

- For more general cases, subtasks can be themselves be broken down into partition, map and reduce phases. The notification-based architecture of the CrowdForge prototype allows this kind of nesting to be implemented as a custom flow.

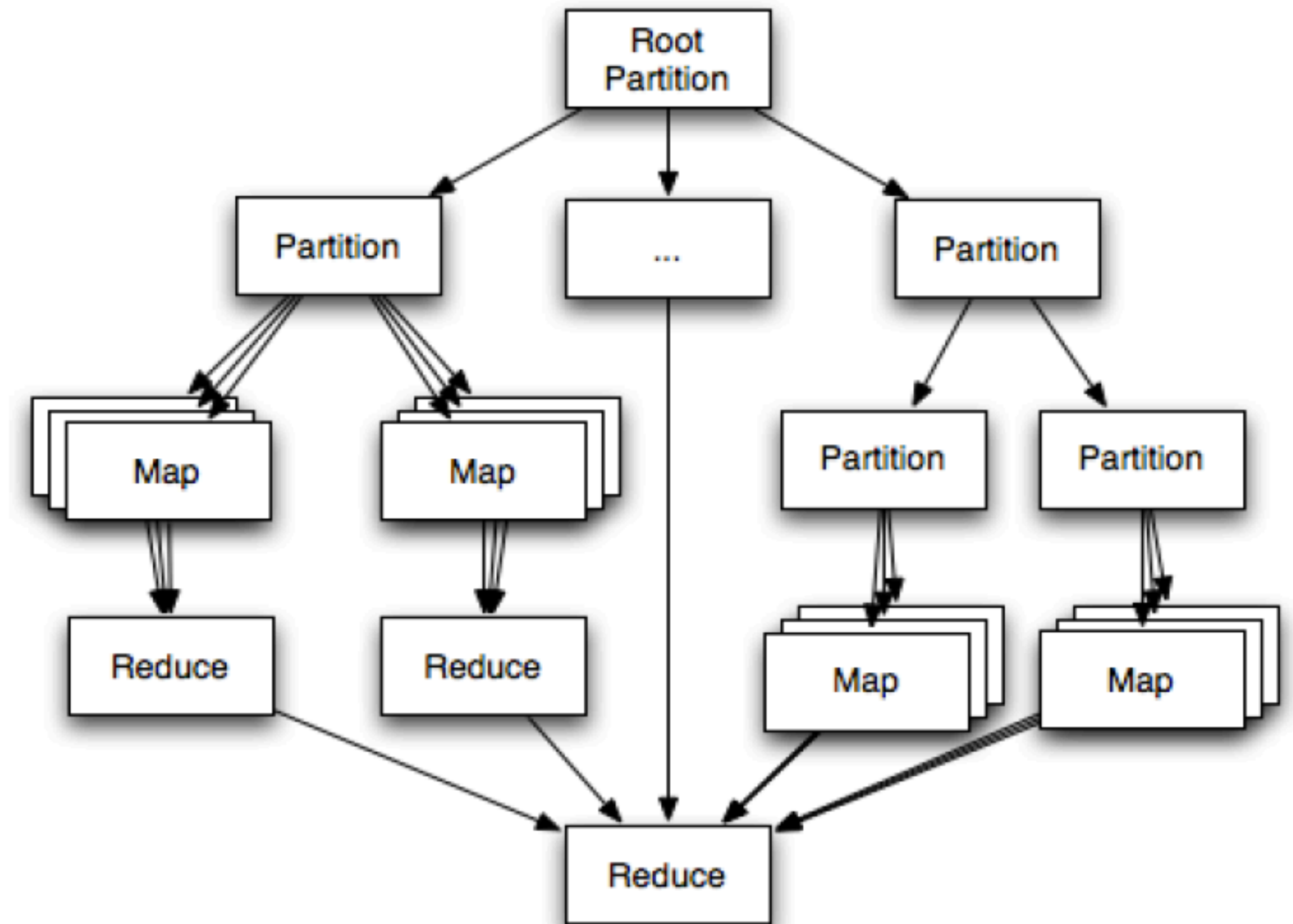
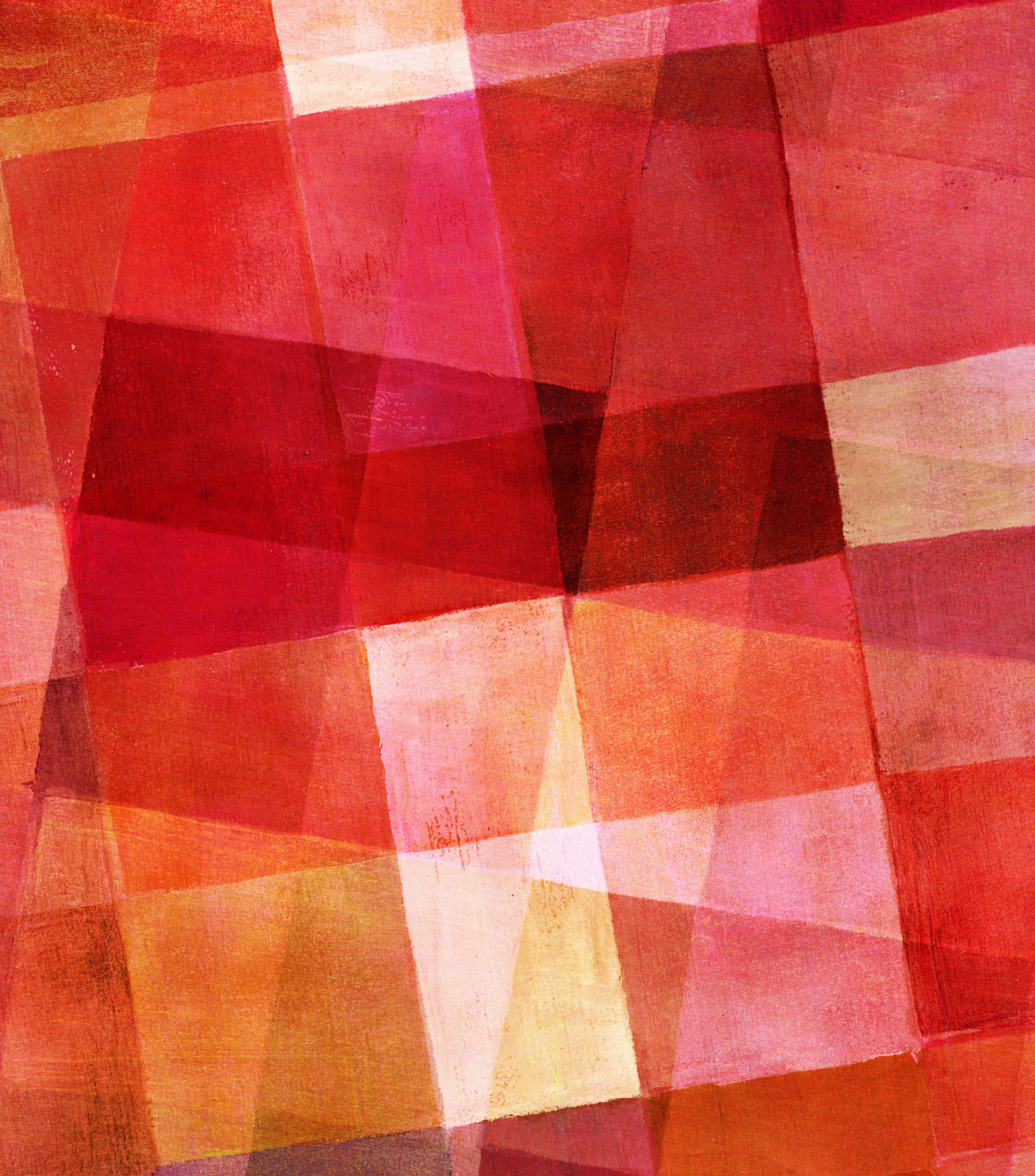


Figure 7. Nested subtasks forming a complex flow.



Crowdsourcing Science Journalism

EXAMPLE

Turning a paper published in an **academic venue** (such as *Science*) into a **newspaper article for the general public**.

- Just reading a complete academic paper **may require more motivation and expertise** than any single crowd worker possesses, let alone any subsequently writing.
- Furthermore, a science **article has a particular structure** to it that the crowd output would need to adhere to; enforcing this structure provides additional constraints.
- Finally, the task may simply **require more expertise than available** in the task market.

EXAMPLE: WHETHER SCIENCE JOURNALISM COULD BE CROWDSOURCED?

- We worked with the journalists to identify a typical **structure** for a popular science article:

Creating a news lead

Describing what scientists did, what they found

Getting a quote from a relevant expert and an author of the study

Describing implications and future work

EXAMPLE: WHETHER SCIENCE JOURNALISM COULD BE CROWDSOURCED ??

- For each of these **sections** we worked to develop **subflows** that would produce them.
- In total our article generation task involved **11 subflows** comprising **31 subtasks**, which represent **262 worker** judgments and a total cost of approximately **\$100**.

EXAMPLE: WHETHER SCIENCE JOURNALISM COULD BE CROWDSOURCED ??

➤ **Generating a news lead** requires quickly and succinctly conveying what the article is about in an engaging way that draws the reader in. First

➤ First, we used a “**consolidate**” process pattern in which the results of a large set are consolidated down to an input that better matches the limited attention profile of the worker.

➤ Second, we provided workers with concrete **examples** of what we desired from them.

➤ **Generate a description of the research procedures.**

➤ First, a partition task asked workers to **extract** the sections from the article that corresponded to different experiments.

➤ Then, for each experiment workers were asked to **summarize** what the researchers actually did, again providing them with a sample experiment and description.

EVALUATION

- To evaluate the quality of the resulting items we enlisted experts with complementary skills, including a professional journalist, the first author of the *Science* paper, a graduate student doing research on social computing, and one of the authors of this study.
- Overall, the results were **surprisingly good**.

“

It was a bit below what you would see in a high-quality publication like the NY Times, but the best were not totally different (although the worst were pretty bad.

-The author of the paper



I'm really impressed by the quality of the answers. The abstract from the Salganik paper is not that technical by the standards of scientific literature, but the key news point -- that social influence helps determine success -- is contained in just one line. Yet 7 of the 10 workers who did the task put that point in their lead.

-The professional journalist

THE BEST AND WORST RATED:



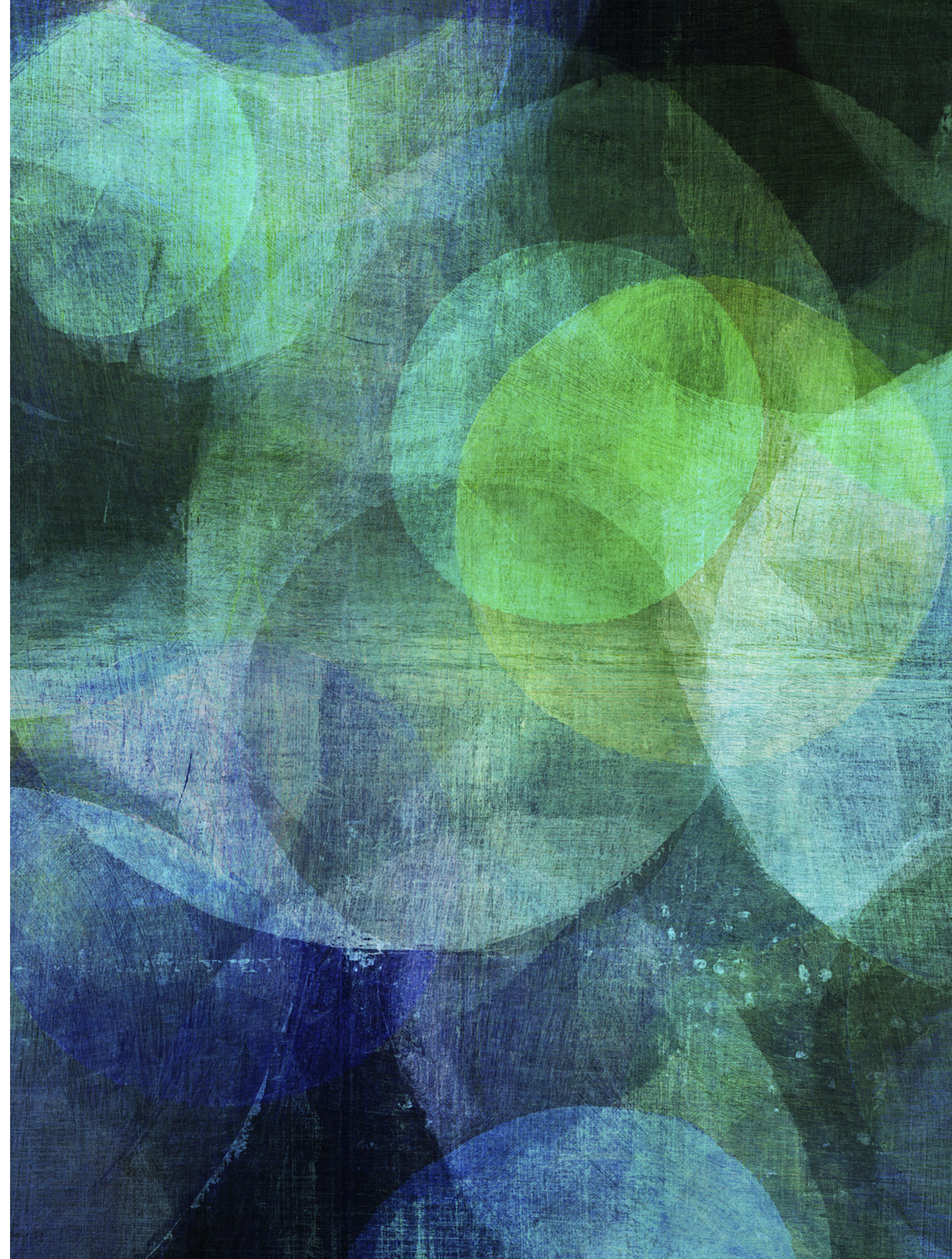
“Blockbusters, bestsellers, hit songs - the main variable that makes these things more popular than their lesser-known counterparts is not quality differences, according to a recent study. The answer lies in **social influence** - the more people know a certain movie is the one to watch, the more likely it will become popular.” (Best)

“The psychology of song preference. Song quality isn't the best predictor of song success.” (Worst)

EVALUATION

- Together these results suggest that despite the lack of expertise, limited time and effort, and limited context provided by crowd workers, assembling the output of many small judgments through the **proper coordination** can result in **highly complex artifacts of surprisingly good quality**.

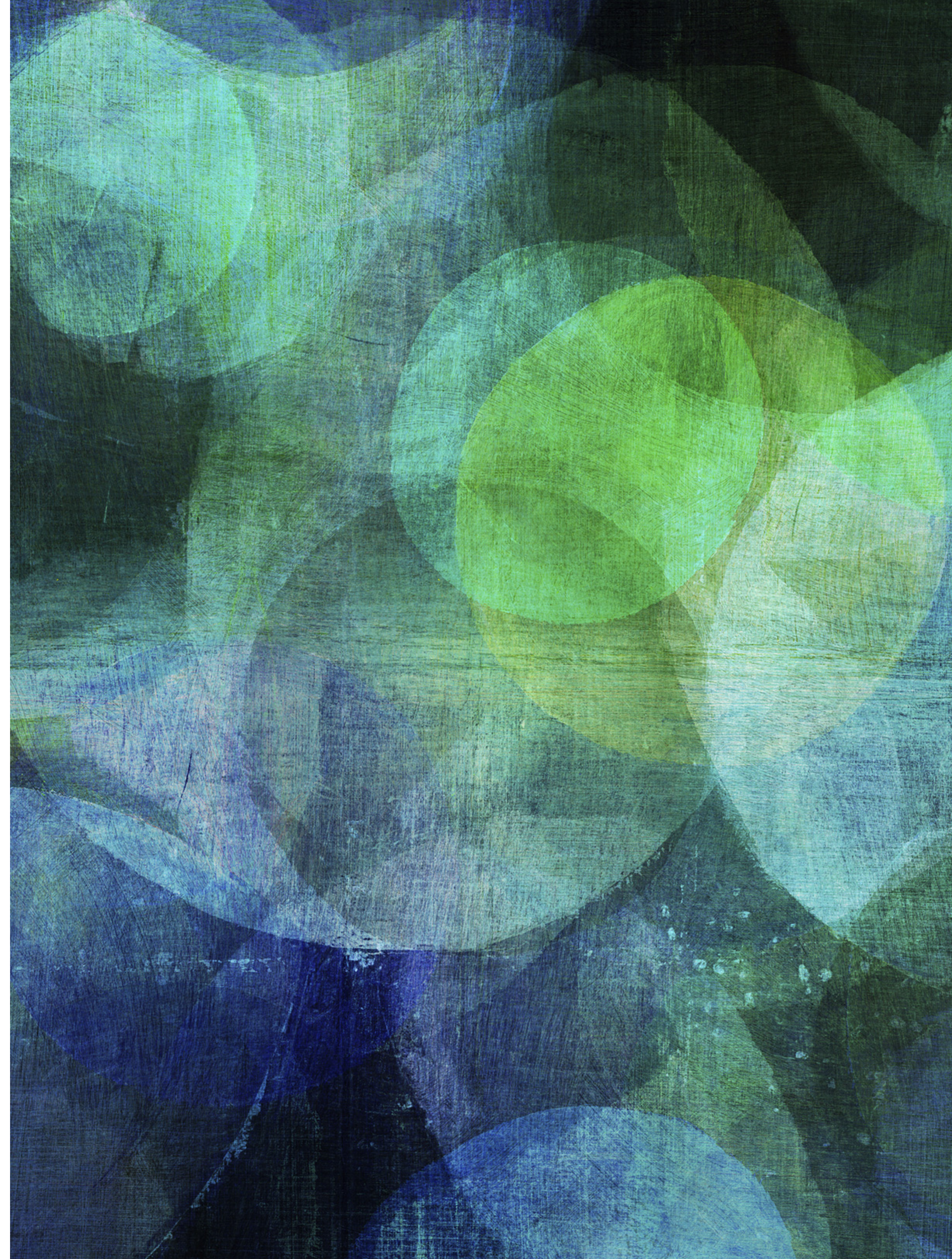
Limitations



LIMITATIONS

- First, the system currently **does not support iteration or recursion**, requiring the task designer to specify each stage in the task flow.
- Second, it is possible that **some work may not be easily decomposable** into units small enough to match the task capacity of the work-force.
- Another possibility is that the decomposition and recomposition of tasks, along with necessary intermediate quality control steps, could **introduce more overhead and cost than they are worth**.
- Understanding the appropriate times and ways to **provide context and visibility into the work of others** is an important area for future crowdsourcing research.

Conclusion



CONCLUSION

- Based on concepts from coordination science and distributed computing, the CrowdForge framework provides a systematic and dynamic way **to break down tasks into subtasks and manage the flow** and dependencies between them.
- We demonstrate through three **case studies** and multiple experiments
- Furthermore, as the nature of work itself becomes more distributed, enabling many more people to be involved in solving complex problems. The CrowdForge framework **reduces this need for predefinition by allowing for subtasks to be dynamically generated by the market itself.**
- Using humans instead of machines as processors provides both distinct benefits and challenges.

FUTURE WORK

- One challenge is **extending our GUI to support more complex, nested flows** so that task designers with no programming experience can complete arbitrarily complex work that involves high coordination dependencies.
- Exploring the possibilities of the CrowdForge framework in very **different kinds of task markets**.

Thank you!
Any questions?

