

# UC Irvine

## UC Irvine Electronic Theses and Dissertations

### Title

A Comparative Study of the Role of Examples in Microtask Crowdsourcing for Software Design

### Permalink

<https://escholarship.org/uc/item/2qg8k46v>

### Author

Spanghero, Fernando Previtalli

### Publication Date

2016

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA,  
IRVINE

A Comparative Study of the Role of Examples in Microtask Crowdsourcing for Software  
Design

THESIS

submitted in partial satisfaction of the requirements  
for the degree of

MASTER OF SCIENCE

in Software Engineering

by

Fernando Spanghero

Thesis Committee:  
Professor André van der Hoek, Chair  
Associate Professor James A. Jones  
Assistant Professor Thomas LaToza

2016



## **DEDICATION**

I dedicate this thesis to my wife, Talitha, for being my best friend and biggest supporter. Her love, patience and comforting words were invaluable. I love you baby.

I also dedicate this thesis to my parents, Fernando and Lucia, for providing me the best education they could and for encouraging me to pursue this endeavor, even if it meant having a son living thousands of miles away.

# TABLE OF CONTENTS

LIST OF FIGURES .....	iv
LIST OF TABLES .....	v
ACKNOWLEDGMENTS.....	vi
ABSTRACT OF THE THESIS .....	vii
1 Introduction .....	1
2 Background.....	7
2.1 Crowdsourcing .....	7
2.2 Crowdsourcing and Microtasking.....	12
2.3 Quality Issues in Crowdsourcing .....	13
2.4 Shepherding the crowd .....	15
3 EXPERIMENT DESIGN .....	17
3.1 Overview .....	17
3.2 Worker Qualification .....	18
3.3 CrowdDesign .....	19
3.4 Decision Points .....	20
3.5 Solution Alternatives Sketching .....	21
3.6 Exit Survey.....	22
3.7 Compensation .....	22
3.8 Participation.....	23
3.9 Review Procedure.....	25
4 DATA ANALYSIS.....	29
4.1 Quantity .....	29
4.2 Diversity .....	31
4.3 Quality.....	37
4.4 Completeness.....	40
4.5 Difficulty.....	43
4.6 Borrowing Ideas.....	48
5 DISCUSSION .....	53
6 LIMITATIONS AND FUTURE WORK.....	57
7 CONCLUSION .....	61
8 REFERENCES.....	63
9 APPENDIX A: EXPERIMENT ARTIFACTS .....	71
10 APPENDIX B: EXAMPLES OF DESIGNS CREATED BY THE CROWD .....	81

## LIST OF FIGURES

Figure 1.1: Example of a morphological chart.....	3
Figure 2.1: Summary of concepts (Source: Schenk and Guittard, 2009, P. 13) .....	10
Figure 3.1: Experiment high-level workflow .....	18
Figure 3.2: Map building decision point prompt.....	20
Figure 3.3: CrowdDesign Sketching Interface.....	22
Figure 3.4: CrowdDesign real-time workers list interface .....	25
Figure 3.5: CrowdDesign worker submission review interface .....	27
Figure 4.1: Distribution percentages of produced solutions in all experiments.....	30
Figure 4.2: Correlation between completeness and quality scores (R = 0.83).....	42
Figure 4.3: Worker feedback.....	45
Figure 4.4: Correlation between time and quality scores (R = 0.23) .....	48
Figure 4.5: Correlation between time and completeness scores (R = 0.15) .....	48
Figure 4.6: Average quality and completeness scores over time .....	52
Figure 9.1: Amazon Turk HIT prompt page as seen by workers .....	71
Figure 9.2: Worker consent form .....	72
Figure 9.3: Worker demographics form .....	73
Figure 9.4: Qualification test 1 .....	74
Figure 9.5: Qualification test 2 .....	75
Figure 9.6: Qualification test 3 .....	76
Figure 9.7: Qualification test 4 .....	77
Figure 9.8: Map building decision point prompt.....	78
Figure 9.9: Visualizing traffic decision point prompt.....	78
Figure 9.10: Setting traffic lights timings decision point prompt .....	79
Figure 9.11: Traffic flows decision point prompt .....	79
Figure 9.12: Exit survey .....	80
Figure 10.1: Solution alternative accepted as example - Map building .....	81
Figure 10.2: Solution alternative rejected as example - Map building.....	82
Figure 10.3: Solution alternative accepted as example – Visualizing traffic .....	83
Figure 10.4: Solution alternative rejected as example – Visualizing traffic.....	84
Figure 10.5: Solution alternative accepted as example – Setting traffic lights timings .....	85
Figure 10.6: Solution alternative rejected as example – Setting traffic lights timings .....	86
Figure 10.7: Solution alternative accepted as example – Traffic flows .....	87
Figure 10.8: Solution alternative rejected as example – Traffic flows .....	88
Figure 10.9: Solution alternative with highest quality score.....	89
Figure 10.10: Solution alternative not compensated.....	90

## LIST OF TABLES

Table 2.1: Design-time quality control approaches (Source: Allahbakhsh et al., 2013, P. 79).....	14
Table 2.2: Run-time quality control approaches (Source: Allahbakhsh et al., 2013, P. 79) .....	15
Table 3.1: Worker participation in the qualification process .....	23
Table 3.2: Worker participation in CrowdDesign .....	24
Table 3.3: Occupation of workers who completed the design task .....	25
Table 3.4: Workers submissions review summary .....	28
Table 4.1: Distribution of compensated solutions per decision point.....	30
Table 4.2: Distribution of accepted solutions per decision point.....	31
Table 4.3: Number of solutions and workers in all experiments.....	31
Table 4.4: Identified solution categories per decision point in all experiments.....	32
Table 4.5: Solution categories per worker - Map building decision point.....	33
Table 4.6: Solution categories per worker - Visualizing traffic decision point .....	34
Table 4.7: Solution categories per worker - Setting traffic lights timings decision point .....	35
Table 4.8: Solution categories per worker - Traffic flows decision point .....	36
Table 4.9: Generation of solution categories by workers in all experiments .....	37
Table 4.10: Distribution of quality scores per decision point.....	38
Table 4.11: Distribution of quality scores in all experiments .....	39
Table 4.12: Distribution of quality scores in our experiment.....	39
Table 4.13: Correlation between worker occupation and quality scores.....	40
Table 4.14: Distribution of the number of requirements met per decision point .....	41
Table 4.15: Distribution of requirements met in all experiments .....	41
Table 4.16: Correlation between worker occupation and completeness scores.....	43
Table 4.17: Number of workers who did not complete the task in all experiments .....	44
Table 4.18: Worker feedback in all experiments .....	46
Table 4.19: Difficulty scores in all experiments.....	47
Table 4.20: Use of CrowdDesign's copy and duplicate features.....	49
Table 4.21: Use of CrowdDesign's copy and duplicate features in experiment 2.....	50
Table 4.22: Chronological distances of solutions of the same category in all experiments .....	51
Table 5.1: Results summary of all experiments .....	53

## ACKNOWLEDGMENTS

This thesis became a reality with the kind support and help of many individuals. I want to extend my deepest gratitude to all of them.

André van der Hoek, my research advisor, for his full support and for imparting his knowledge and expertise in this work. Without his patience and timely counsel, this work would have been a frustrating pursuit.

The Brazilian government and CAPES (Coordination for the Improvement of Higher Education Personnel), for fully funding my graduate program at UCI through the Science without Borders program. Without their support, this work would not have been possible.

CrowdDesign team members Consuelo Lopez, Edgard Weidema and Sahand Nayebaziz. Our teamwork, synergy and insightful discussions were essential to build the foundation of this work.

All UCI SDCL members for their precious contribution, especially Lee Martie, Christian Adriano and Thomas Kwak for their precious suggestions and help in many activities.

Mengyao Zhao and Leah Horgan for their time and effort to grade all sketches.

The University of California, Irvine for all support, resources and funds



## **ABSTRACT OF THE THESIS**

A Comparative Study of the Role of Examples in Microtask Crowdsourcing for Software Design

By

Fernando Spanghero

Master of Science in Software Engineering

University of California, Irvine, 2016

Professor André van der Hoek, Chair

Crowdsourcing is gradually becoming an accepted form of work across different disciplines. Not surprisingly, it has attracted the attention of the software engineering community as well. Previous work started exploring the feasibility of crowdsourcing for software design by conducting experiments in which workers from Amazon Mechanical Turk were asked to engage in a set of software design tasks. It was found that, when workers are exposed to examples of previous designs, they generate overall lower quality contributions. The intuition is that, since these experiments displayed all previous contributions as examples to workers, the presence of low quality examples may have negatively influenced workers.

This thesis compares the designs produced in the previous experiments to designs obtained in a new experiment in which examples were evaluated against pre-defined quality criteria before being displayed to workers. Only examples that were of sufficient quality were shared with workers, with the hope of stimulating them to provide higher quality designs.

We report results from an analysis in which we compare the designs from the current and previous experiments in terms of quantity, diversity of ideas, quality, completeness, perceived task difficulty, and how often workers borrow elements from examples. The major findings are twofold. First, workers who were exposed to sufficient quality examples produced better quality work as compared to workers exposed to all examples. Second, the quality of the designs they produced still did not reach the quality of the designs produced by workers who were not exposed to examples at all.

**Keywords:** Crowdsourcing, software design, alternatives, examples

# 1 Introduction

The ability to easily reach millions of individuals across the globe has enabled crowdsourcing to become an increasingly popular work model for many different types of disciplines [1]. Defined as *'the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call'* [2], crowdsourcing has found success in areas such as graphical design [3], language translation [4], and others [5].

Crowdsourcing has been a subject of interest in both industry and academia as a way of supporting software engineering work [6]. A range of web platforms exists that tackle highly focused software engineering activities by employing different models of crowdsourcing. As examples, TopCoder [7] uses a competition model to find solutions for programming problems, 99designs [3] uses a similar model for user interface design, StackOverflow [8] provides a community-managed Q&A platform for all sorts of software development topics, and uTest [9] allows software vendors to reach a network of testers that can debug their applications in a model similar to freelancing.

However, as of today, these platforms cover a limited set of software engineering activities and only recently research has started to attempt to broaden that set. Complex tasks such as requirements gathering and software architecture are now being studied [10]. Even so, skepticism exists as to whether such complex tasks are suitable to be performed with crowdsourcing [10, 11].

To contribute to this discussion, this thesis explores the use of crowdsourcing in software design. Platforms such as TopCoder [7] and 99designs [3] have had some success

employing a competition model, though its effectiveness has been questioned [10], with the major limitation being that it prevents collaboration: designers work independently and possibly many good ideas are lost because a single winning solution must be chosen while all other solutions are discarded. A recent study [70] began to evaluate the use of a microtasking model, which breaks a problem into small tasks that are distributed to a crowd [68]. Instead of choosing a single winning contribution and discarding possibly good ideas, this model relies on harnessing small contributions made by different individuals [68]. Another recent study [73] showed that microtasking facilitates collaboration when combined with platforms that are designed to stimulate collaboration among participants.

Our study uses an approach based on the concept of a morphological chart [12] (also known as a concept combination table [13] or function-means table [14]), a technique widely used in other engineering disciplines. A morphological chart breaks a problem into smaller, self-contained sub-problems, called decision points, and stimulates the generation of multiple solution alternatives for each decision point. An overall solution is then obtained by selecting one solution alternative from each decision point in a way that the full set is both as compatible and functional as possible. Figure 1.1 shows an example of a morphological chart.


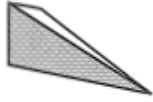


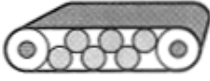

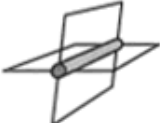

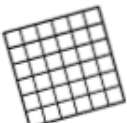



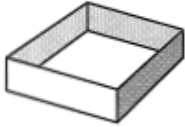

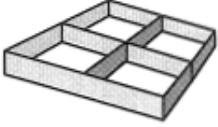



	Option 1	Option 2	Option 3	Option 4
Vegetable picking device		 Triangular plow	 Tubular grabber	 Mechanical picker
Vegetable placing device	 Conveyor belt	 Rake	 Rotating mover	 Force from vegetable accumulation
Dirt sifting device	 Square mesh	 Water from well	 Slits in plow or carrier	
Packaging device				
Method of transportation		 Track system	 Sled	
Power source	Hand pushed	Horse drawn	Wind blown	Pedal driven

Figure 1.1: Example of a morphological chart

Three major research questions were identified by the former study with respect to the use of a morphological chart to support crowdsourcing software design:

1. *Can a crowd identify key decision points?* Given a set of requirements, can the crowd identify the major decision points that represent the essence of the design problem to be addressed? Can they specify the decision points in a clear manner?

2. *Can a crowd identify solution alternatives?* Can the crowd generate a diverse set of solutions for each decision point? Are these solutions considered of good quality?
  
3. *Can a crowd identify a complete solution?* Given all decision points and solution alternatives, can the crowd obtain a complete solution where each individual solution alternative addresses the key requirements of the design problem and is compatible with the others?

All experiments done so far by the previous study focused on the second question. Specifically, a crowd of workers from Amazon Mechanical Turk [5] was asked to perform a set of design tasks of an educational traffic simulation software system. One of the experimental conditions was whether were presented with the designs produced by previous workers. The result was surprising: exposing previous designs to workers did not improve the solution alternatives in terms of quality, quantity, or diversity. In fact, the average quality of solutions was noticeably worse compared to the quality of the designs produced when workers had no examples at all.

This thesis is based on the intuition that the quality went down because workers were exposed to all of the previous designs; whether of high quality, low quality, or anywhere in between. Specifically, it is known that workers from Amazon Mechanical Turk do not want to spend too much time on tasks [61, 74] and thus might gauge their work toward the quality of previous work. If some of the previous designs are of low quality, then perhaps it is not surprising that this effect happens.

This thesis' main focus is to explore this effect in further detail. Specifically, it is based on the hypothesis that we believe it is possible to eliminate the effect of bad examples by only showing workers examples that are of a certain minimum quality. To explore this hypothesis, we reproduce the experiment in which workers were asked to design the user interface of the traffic simulation software system while being exposed to examples, with one key difference: designs submitted by the crowd are evaluated against a pre-defined set of criteria that determine whether they have the necessary quality to be displayed as examples. That is, a reviewer examines each submitted design as the experiment unfolds, and manually decides which designs are shown as examples, and which ones are not. The results are then compared to results from the previous experiments in terms of quantity, diversity of ideas, quality, completeness, how workers perceive task difficulty, and how often workers borrow elements from the examples presented to them.

The major findings are twofold. First, workers who were exposed to sufficient quality examples produced better quality work as compared to workers exposed to all examples. Second, the quality of the designs they produced still did not reach the quality of the designs produced by workers who were not exposed to examples at all.

This thesis is organized as follows. Chapter 2 provides background material regarding crowdsourcing, how crowdsourcing relates to software engineering, microtasking, and crowdsourcing quality issues. Chapter 3 details the experimental setting by describing the tool accessed by workers to create and submit their work contributions and detailing the criteria used to assess which designs are to be displayed as examples. Chapters 4 and 5 present the results, compare them with previous work, and discuss the

findings. Chapter 6 discusses limitations of the experiment and introduces an outlook at future work. Chapter 7 concludes this thesis.



## 2 Background

### 2.1 Crowdsourcing

#### What is crowdsourcing?

Multiple definitions for crowdsourcing can be found in the literature. The term was coined by Howe, in 2006, who defines it as *‘the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call’* [2]. Greengard defines it as *“a powerful mechanism for outsourcing tasks, which are traditionally performed by a specialist or small group of experts, to a large group of humans”* [15]. Brabham describes it as *“an online, distributed problem solving and production model that leverages the collective intelligence of online communities for specific purposes set forth by a crowdsourcing organization—corporate, government, or volunteer”* [16].

Based on these definitions, we can identify common features related to crowdsourcing: it gives open access to the crowd to the production environment; there is flexibility in the workforce; the workers have free will to participate and there are mutual benefits among stakeholders [6]. Along with these features, some challenges need to be addressed. Doan et al. explain that any crowdsourcing system faces four main challenges: recruiting users; providing an infrastructure through which users can make contributions; assessing the quality of contributions; and combining contributions into a single, coherent, solution that solves the problem [1].

Crowdsourcing has been used extensively in various disciplines. Applications can be found in protein structure prediction [25, 26], drug discovery [27, 28], transportation

planning [29, 30], and information retrieval [31, 32], among others. Besides, many applications can be found in software engineering [10, 33-37], which is the discipline that is subject of focus of this thesis.

## **Related concepts**

Since crowdsourcing is a relatively new phenomenon, there may be confusion with other concepts that share similar features. In the following paragraphs, I highlight the differences and similarities between these concepts and crowdsourcing.

**Open innovation** is a concept developed by Chesbrough [20], where companies distribute their knowledge among each other. It means they do not rely single handedly on their own research and development, but also on that of other companies. Although open innovation shares the idea of knowledge distribution with crowdsourcing, important differences exist. While crowdsourcing focuses on the interaction between a company and a crowd, open innovation focus on the interaction between companies. Besides, open innovation involves the idea of buying and selling knowledge by companies (for example through patents), whereas crowdsourcing harnesses knowledge from the crowd [17].

**User innovation** is an approach developed by von Hippel [21], where innovation is led by users who have specific needs. They face issues when using a given product and are motivated to make modifications or make a new product that fits their needs. Both crowdsourcing and user innovation have the participation of individuals outside of professional companies, though in crowdsourcing demand comes from a company while user innovation is user driven [17].

**Open source innovation** is a concept most commonly known in software development. A core group of contributors first develop a rough version of a product and then it is made freely available so that others can collaboratively improve and distribute it [22] [23]. Because of this distribution model, the product is not owned by any particular individual and belongs to the community [23]. The core group members that initially developed the product can become managers and maintainers of the project, although these roles are not limited to them [22]. Crowdsourcing and open source innovation both make use of external individuals. Their biggest difference, however, is that, while crowdsourcing is driven by a company that owns the project or idea, open source innovation has no sense of particular ownership nor enforces it by the use of patents [17, 23].

**Outsourcing** is a broad term that is generally defined as a practice in which an organization looks for goods and/or services from outside companies [24]. Although crowdsourcing can be considered a form of outsourcing, the main difference is that it is directed to a crowd rather than to companies.

Figure 2.1 summarizes all these concepts and how they are related.

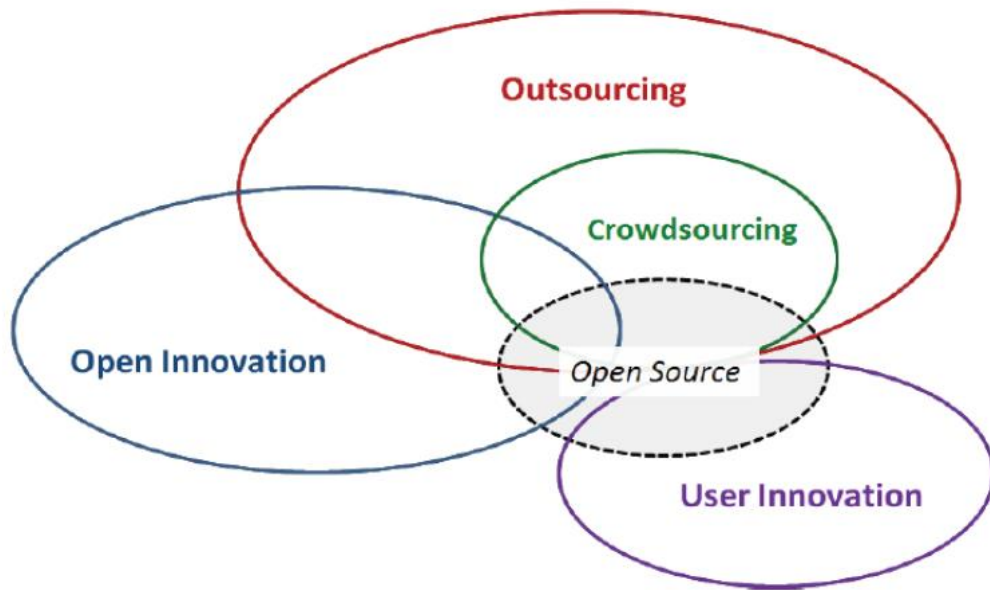


Figure 2.1: Summary of concepts (Source: Schenk and Guittard, 2009, P. 13)

### Benefits of crowdsourcing

The following are the main benefits of crowdsourcing described in the literature:

1. **Cost:** although cost may vary greatly depending on the crowdsourcing model, it is generally lower than in traditional work models. Since workers are typically amateurs or individuals that only want to practice their skills, they are willing to be paid lower remunerations [10, 17]. Besides, a formal employment contract between requesters and workers is not established, meaning lower costs to requesters [10].
2. **Quality:** quality is achieved through broad participation. That is, requesters have access to a large and diverse pool of workers who voluntarily select the tasks on the basis that they possess the necessary skills to provide contributions of sufficient quality [10].

3. **Time-to-market:** two main factors may help reducing in time-to-market in crowdsourcing. First, work can be performed at any time a day since workers are geographically distributed across multiple time-zones [10]. Second, depending on the nature of the work, it can be parallelized across a large number of workers [18, 19].
4. **Creativity and innovation:** one of the key characteristics of crowdsourcing is the diversity of the crowd. Workers come from many different backgrounds and possess a variety of skills. This intellectually rich environment stimulates the production of creative and innovative approaches to solve problems [10].
5. **Motivation:** voluntary work and autonomy are likely factors that encourage worker participation. Additionally, creative and/or problem-solving tasks require a diverse range of skills and therefore motivate participants that are interested in practicing their abilities [17].

### **Crowdsourcing models**

As mentioned previously, crowdsourcing faces four main challenges related to recruiting users, providing access to users, assessing contribution quality, and combining contributions. Over the years, different authors have developed models that address these challenges using different approaches.

Howe [41] defines four primary types of crowdsourcing: **crowd wisdom**, where there is an attempt to harness knowledge from as many people as possible to solve a problem or predict future outcomes (i.e., idea jams [66], prediction markets [67]); **crowd funding**, where a crowd offers financing for a product or service in which they are interested that might otherwise be denied by traditional credit channels (i.e., kickstarter

[38]); **crowd voting**, which leverages the collective judgment of a crowd to organize, filter and rank any kind of content (i.e., LEGO™ ideas [39]); and **crowd creation**, where individuals are asked to generate content of varying complexity (i.e., TopCoder [7], iStockPhoto [40]). Howe argues that requesters should carefully assess their objectives in order to select a crowdsourcing strategy that best fits their needs [41].

Saxton et al. [42] has studied several organizations that employ crowdsourcing and defined a different set of crowdsourcing categories based on the business model employed: intermediary model, citizen media production model, collaborative software development model, digital goods sales model, product design model, peer-to-peer social financing model, consumer report model, knowledge based building model and collaborative science project model.

In the context of software engineering, LaToza and van der Hoek [43] propose three main crowdsourcing models for software engineering tasks: **peer production**, in which workers collaborate towards a project or product (open source development is the most prominent example); **competitions**, in which workers compete against each other and a single winning contribution is selected from all of the submitted solutions (TopCoder [7] and 99designs [3] are examples of competition-based web platforms); **microtasking**, in which a complex problem is broken down into small, self-contained tasks that can be more easily distributed and parallelized among a crowd.

## 2.2 Crowdsourcing and Microtasking

Sarasua et. al define microtasking in crowdsourcing as a *“problem-solving model in which a problem is outsourced to a distributed group of people by splitting the problem space*

*into smaller sub-problems, or tasks, that multiple workers address independently in return for a (financial) reward” [44]. In this model, workers are recruited using an open call and work on one or more small, self-contained tasks that typically can be completed within minutes [43, 45]. By breaking down a complex problem into microtasks, a requester is able to harness the combined effort of the crowd to obtain a solution [43].*

Perhaps the most popular application of this model is the Amazon Mechanical Turk platform [5], which provides a virtual labor marketplace for microtasks as well as the infrastructure for task design, publication, assignment, and payment. The majority of the microtasks published in Amazon Turk are simple activities that can be easily divided and distributed to a large number of workers [46]. Examples are information finding on the Web, content labeling, categorization, ranking, and language translation [46-48].

Recently, microtasking has been finding success when combined with complex tasks. Examples are the organization of complex information [49, 50], graphical perception [51], user interface design [52], and product design [53]. Some other studies showed success in microtasking complex work using Amazon Mechanical Turk, such as writing articles [54], verifying statements in an ontology [55], and providing feedback for Wikipedia articles [56].

### **2.3 Quality Issues in Crowdsourcing**

Even though quality is claimed as one of the benefits of crowdsourcing, it can be negatively affected by a number of issues. As an example, workers might have insufficient skills for performing certain tasks [57]. As another, they may have malicious intentions such as sabotaging a task or quickly finishing a task for monetary gains, usually by abusing

the system or providing extremely low quality, disposable contributions [58]. Additionally, tasks can be ill-defined and not provide sufficient information about their requirements to workers, which may cause confusion and therefore impact the overall quality of contributions [59].

In the context of microtasking crowdsourcing platforms, Ipeirotis et al. raise the problem of quality control in Amazon Mechanical Turk [46]. Kazai et al. analyze the behavior of workers in Amazon Mechanical Turk and categorize them as either sloppy, spammer, incompetent, competent, or diligent [60].

Some counter-measures have been developed to address quality issues in crowdsourcing systems. Allahbakhsh et al. present a comprehensive compilation of quality control approaches in crowdsourcing systems [61]. They divide these into **design time approaches** and **run-time approaches**, as shown in Table 2.1 and Table 2.2, respectively.

Quality-control approach	Subcategories	Description
Effective task preparation	Defensive design	Provides an unambiguous description of the task; task design is defensive — that is, cheating isn't easier than doing the task; defines evaluation and compensation criteria
Worker selection	Open to all	Allows everybody to contribute to the task
	Reputation-based	Lets only workers with prespecified reputation levels contribute to the task
	Credential-based	Allows only workers with prespecified credentials to do the task

**Table 2.1: Design-time quality control approaches (Source: Allahbakhsh et al., 2013, P. 79)**



Quality-control approach	Description
Expert review	Domain experts check contribution quality.
Output agreement	If workers independently and simultaneously provide the same description for an input, they are deemed correct.
Input agreement	Independent workers receive an input and describe it to each other. If they all decided that it's a same input, it's accepted as a quality answer.
Ground truth	Compares answers with a gold standard, such as known answers or common sense facts to check the quality.
Majority consensus	The judgment of a majority of reviewers on the contribution's quality is accepted as its real quality.
Contributor evaluation	Assesses a contribution based on the contributor's quality.
Real-time support	Provides shepherding and support to workers in real time to help them increase contribution quality.
Workflow management	Designs a suitable workflow for a complex task; workflow is monitored to control quality, cost, and so on, on the fly.

Table 2.2: Run-time quality control approaches (Source: Allahbakhsh et al., 2013, P. 79)

## 2.4 Shepherding the crowd

Dow & Kulkarni's original idea of shepherding the crowd consists of exploring "*the value of providing real-time assessment to help motivate and teach online workers to produce high-quality results*" [62]. They developed a system called *Shepherd*, which provides a feedback infrastructure to crowdsourced work. In order to implement such a system, they argue that only someone with sufficient domain knowledge is able to provide external feedback [62, 63]. They found that workers who received feedback produced contributions of higher quality. While this thesis does not employ the idea of providing real-time feedback to workers, it borrows the concept of guiding the crowd by displaying existing contributions that are reviewed against certain minimum quality standards.

The idea of using review to enhance quality in crowdsourcing has been the subject of study of different authors. Chan et al. call it **expert facilitation** and offer a system called

*IdeaGens*, where “experts monitor incoming ideas through a dashboard and offer high-level “inspirations” to guide ideation” [64]. Hung et al. call it **expert guidance** and also offer a tool called *ERICA* that supports expert review of crowdsourced work by collecting input, estimating quality, and optimizing allocation of crowd contributions to experts [65].

To date, no studies exist that combine the use of crowdsourced work with expert review for complex software engineering tasks, such as software architecture, requirements gathering, or software design. Dow & Kulkarni study a crowd that performs product reviewing microtasks [62]. Chan et al. observe workers that generate ideas for a social problem [64]. Hung et al. do not clearly state what type of tasks workers are asked to perform, although they are related to answering multiple-choice questions [65]. While these studies show promise, they tackle relatively straightforward problems that do not require significant complexity to be distributed to a crowd and reviewed.

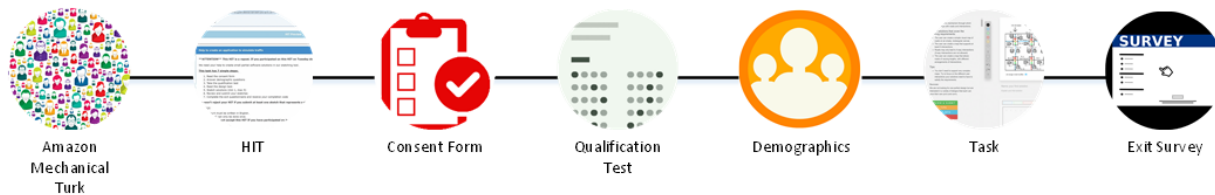
## 3 EXPERIMENT DESIGN

### 3.1 Overview

We conducted an experiment on Amazon Mechanical Turk [5], for which we posted one human intelligence task (HIT) requesting workers to create between one and five solution alternatives for a small user interface design task of an educational traffic simulation software system. When workers decided to participate in the HIT, they had to click on a link in the HIT description that took them to the experiment platform. The traffic simulation problem was broken down into four decision points, each exploring a different aspect of the user interface. These decision points were selected during the previous experiments after the examination of existing complete designs previously created by professional software designers (see Petre and van der Hoek for the detailed design prompt [71]). Once in the platform, workers had to undergo a qualification process divided in three steps. In the first step they had to read and ‘sign’ a consent form. In the second step, they had to provide basic demographic information. In the third step, they had to qualify for the experiment by passing a test consisting of five multiple choice questions covering user interface design principles. Finally, if they passed the test by answering at least three of the questions correctly, they were given access to the actual task.

The task asked workers to provide at least one and up to five different solution alternatives for a given decision point of the traffic simulation software system. During the task, workers could see contributions submitted by previous workers, specifically those contributions that were deemed of sufficient quality. After the worker submitted their solution alternatives, they were asked to complete a questionnaire about their experience

and to provide optional feedback. Workers were then provided with a unique completion code to be submitted on Amazon Mechanical Turk to show that they had completed the task. Finally, the solution alternatives were reviewed and workers were compensated according to the quantity and quality of their contributions. After completing the HIT, workers were not allowed to take it again. Figure 3.1 shows an overview of the entire workflow.



**Figure 3.1: Experiment high-level workflow**

The following sections describe each workflow step in more detail. All the artifacts used during the experiment are available in Appendix A, including the consent form, the demographics form, the qualification test questions, the task decision points, and the exit survey.

## **3.2 Worker Qualification**

In order to access the design task, workers were required to first go through a qualification process divided in three steps: reading and signing a consent form, filling a basic demographic data form and passing a qualification test. The consent informed workers about the purpose of the experiment, eligibility requirements, compensation details, privacy concerns, expected participation duration and contact information. The

demographics form asked workers about their occupation, years of work experience, education level, gender, age, and country of residence.

The qualification tests asked workers to answer five multiple-choice questions about general user interface design principles. Workers were required to answer three questions correctly in order to be considered qualified for the design task. Each worker was randomly assigned a qualification test from a pool containing four tests in total. The reason for this was to make it somewhat harder for workers to obtain the correct answers before taking the Amazon Mechanical Turk HIT.

### **3.3 CrowdDesign**

When workers passed the qualification test, they were redirected to an online sketching tool called **CrowdDesign**. CrowdDesign contains three main features: it allows the creation of decision points for a major design problem (to be performed by the researchers); it provides a sketching tool to design solution alternatives (to be performed by the workers); and it provides an administrative interface to view work status and review submitted solution alternatives (to be performed by the researchers).

When workers first access CrowdDesign, they are randomly assigned a design task related to one of the decision points entered by the researchers. Workers cannot make any changes to decision points and are only able to see the decision point related to their assigned task.

### 3.4 Decision Points

In order to facilitate result comparison, the same four decision points from previous experiments were reused: (1) design an interface to build the traffic simulator map; (2) design an interface to visualize traffic; (3) design an interface to set traffic lights timings; and (4) design an interface to control traffic flows. All tasks presented to workers contained a brief description of the decision point, precisely four requirements, a few tips, and a reminder of the overall goal of the HIT. Figure 3.2 shows the detailed task prompt for the map building decision point, which was displayed to workers in CrowdDesign. Appendix A contains the prompts for all four decision points.

#### Task:

Design an interface mechanism through which users build maps with roads and intersections.

#### Sketch solutions that cover the following requirements:

- The user can create a simple visual map of roads on an empty, rectangular canvas.
- The user can create a map that supports at least 6 intersections.
- Roads may only lead to 4-way intersections (3-way intersections are not allowed).
- The user can create a map that allows roads of varying lengths, with different arrangements of intersections.

#### Tips:

- You don't need to support very complex maps. Try to focus on the different user interactions your solutions need to have to satisfy the requirements.

#### Reminder:

We are not looking for one perfect design but are interested in a variety of designs that each can have their own pro's and con's.

**Figure 3.2: Map building decision point prompt**

### 3.5 Solution Alternatives Sketching

Figure 3.3 shows CrowdDesign's sketching interface. On the left of the screen (1), workers were provided with all the instructions necessary to complete the task (i.e., the decision point). In the middle, the platform provides a set of basic sketching features (2), which allow the worker to produce a sketch illustrating their solution alternative on an empty canvas on the right (3). Each canvas has two associated textual fields: one for the name of the solution alternative, and the other for an explanation of the solution alternative (4). Scrolling down reveals four additional canvases where workers can provide up to four additional solution alternatives (5). Once workers are happy with their work, they can use the "Review & Submit" button to submit their work and access the exit survey (6). In the bottom left, workers can scroll down to see previous workers contributions, from which they can copy elements to their own sketches or duplicate the entire solution as a starting point for their work (7).

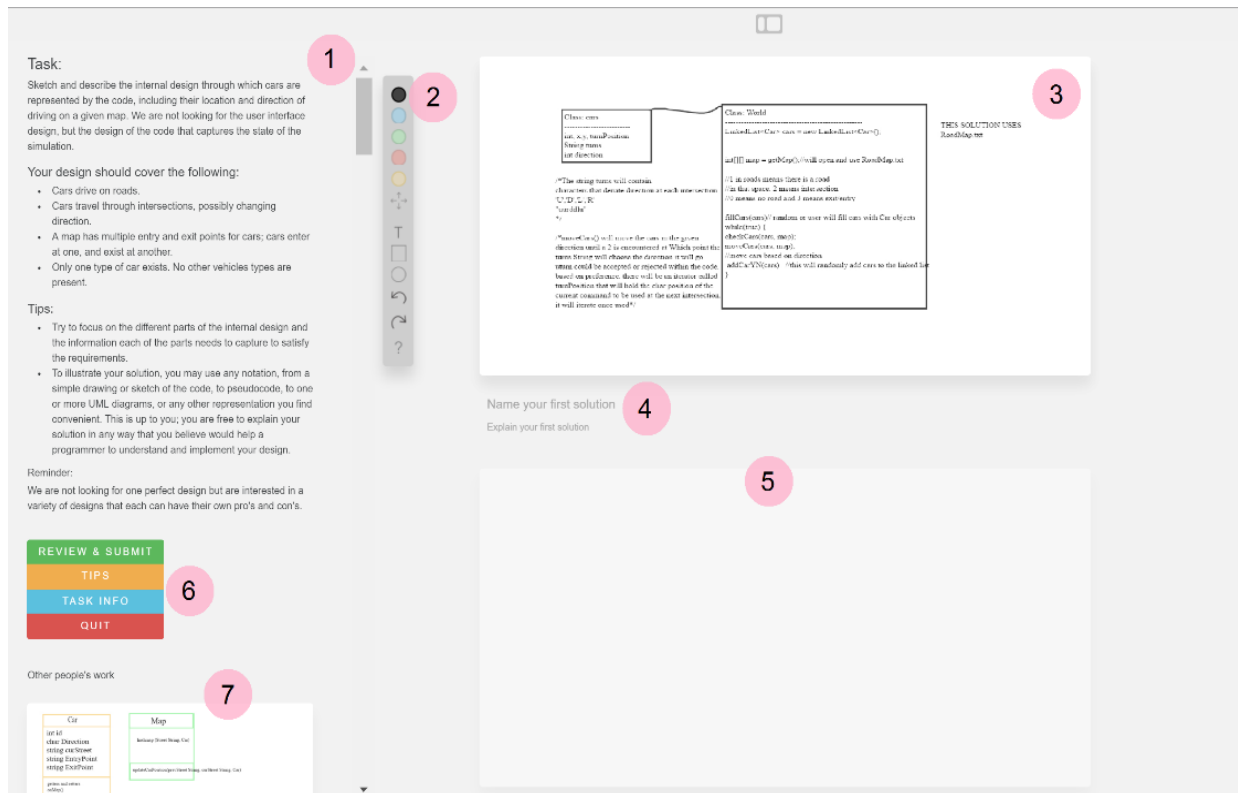


Figure 3.3: CrowdDesign Sketching Interface

### 3.6 Exit Survey

After submitting their solution alternatives, workers were asked to complete an exit survey with four questions. The first three questions asked them to rate on a one (easy) to seven (difficult) scale three aspects: ability to complete the entire task, challenge level of the decision point, and adequacy of support by the tool. The fourth question was optional and open ended, and asked workers for any general feedback they might have.

### 3.7 Compensation

Workers were paid \$2.00 if they provided a valid completion code at the Amazon Turk HIT and clearly tried to provide solutions that addressed the design problem.



Additionally, they were given a bonus of \$0.50 per each submitted sketch that demonstrated honest effort. We considered workers to demonstrate honest effort when they provided at least one solution that genuinely attempted to address the design problem, regardless of whether they are successful or not. They were also given an extra \$1.00 bonus for each solution alternative that met at least three out of four task requirements. Therefore, workers were able to earn up to \$9.50 per task by providing five complete sketches. This compensation is based on the California minimal wage (\$9.00 per hour) and the fact that we expected workers to complete on average five solution alternatives in approximately one hour.

### 3.8 Participation

Table 3.1 shows worker participation in the qualification process. Out of a total of 668 workers who signed the consent form, 123 (18%) quit before taking the qualification test, 362 (54%) workers failed the test, and 183 (27%) passed the test.

Action	# Workers	%
Quit	123	18%
Failed test	362	54%
Passed test	183	27%
<b>Total</b>	<b>668</b>	<b>100.00%</b>

Table 3.1: Worker participation in the qualification process

Table 3.2 shows the outcome for workers who passed the qualification test. Three worker quit right before accessing CrowdDesign, leaving 180 (27%) who were assigned the design task. Out of these 180, 70 (39%) submitted a completion code on Amazon

Mechanical Turk, but 3 did not submit any work of honest effort, leaving 67 (37%) workers whose work was accepted. Out of these 70 workers, eight (11%) also have participated in previous experiments. Note that 110 workers (61%) started the task in some way, but did not complete it and left the tool. From these workers who left, only 12 (7%) took a quit survey providing the reason why they did not complete the task.

Action	# Workers	%
Quit before the design task started	3	2%
Submitted solution alternatives	70	39%
Did not complete the design task and took the quit survey	12	7%
Did not complete the design task and did not take the quit survey	98	54%
<b>Total</b>	<b>183</b>	<b>100 %</b>

**Table 3.2: Worker participation in CrowdDesign**

Table 3.3 shows the occupation of workers who completed the design task. During the qualification process, workers were asked to provide their primary occupation in the demographics form. The majority of the workers, 38 (54%), were hobbyists, followed by nine (13%) professional software developers, eight (11%) undergraduate students, two (3%) graduate students and one (1%) professional UI/UX designer. If any worker did not feel represented by any of the previous options, they had the chance to select “Other” and detail their occupation. 12 (17%) workers selected this option and the most frequent occupations they declared were system administrator, researcher, software tester, and data analyst.

Occupation	# Workers	%
Undergraduate student	8	11%
Graduate student	2	3%
Professional UI/UX designer	1	1%
Professional software developer	9	13%
Hobbyist	38	54%
Other	12	17%
<b>Total</b>	<b>70</b>	<b>100%</b>

**Table 3.3: Occupation of workers who completed the design task**

### 3.9 Review Procedure

Workers' solution alternatives were reviewed after submission. Figure 3.4 shows CrowdDesign's administrative main interface, used internally during the review process. It provides a real-time list of workers who access the tool, allowing the monitoring of work status and distribution. Each worker's submission can be individually accessed for review.

time	worker	session	# of solutions	decision point	status	# accepted	# rejected	actions
03/04/2016 20:29:11	6611a111a4-79	314CA8g1e-8-3-7	1	Visualizing Traffic	reviewed	1	0	review delete
03/04/2016 20:26:02	660Ei-5A4C-90-7	315ee-7g-1E5-8-7	5	Map Building	reviewed	5	0	review delete
03/04/2016 20:19:06	659AC6C-6A928	316ac7g1102-5	0	Traffic Flows	in-progress			review delete
03/04/2016 16:44:40	650IE-8c7e5-70	317ce-9e7G100	0	Setting traffic light timings	in-progress			review delete
03/04/2016 15:44:35	647ce-1e0a12-5	318la1E-3g7-59	0	Visualizing Traffic	in-progress			review delete
03/04/2016 14:25:49	6411a-5A0e-1-3-9	319cC-2E-5G0-76	3	Map Building	reviewed	3	0	review delete
03/04/2016 13:40:28	640Al-4a8a29-1	320EI8C-3E-38-5	3	Traffic Flows	reviewed	3	0	review delete
03/04/2016 12:52:19	637cA-2G0G0-20	321gC9I-9E30-7	2	Setting traffic light timings	in-progress			review delete

**Figure 3.4: CrowdDesign real-time workers list interface**

The solution alternatives reviewing procedure had two steps. In the first step, every time a decision point accumulated submissions from five different workers, they were

reviewed in the submission review interface in CrowdDesign. A reviewer used this interface to accept or reject solutions to be displayed as examples to other workers (see criteria below). In the second step, which happened right after the first, solutions were reviewed again in order to be accepted for compensation in Amazon Mechanical Turk. A solution was eligible for compensation if it demonstrated honest effort in attempting to address the design task. Solutions accepted as examples were considered of honest effort and thus were automatically compensated.

Figure 3.5 shows the submission review interface. It displays all submission information, including the detailed solution alternatives, the time the worker has spent on certain activities, exit survey answers, what examples of previous work were available to the worker and whether the worker borrowed elements from these examples by using features provided by the tool. For each solution alternative, the reviewer has the option to accept or reject it. Accepted solution alternatives were displayed as examples to new workers while rejected solution alternatives were not included in the examples.

ticket: 5eb0EjaauriLGAQu	task difficulty: 5
worker: 636gi-2e3i08-7	decision difficulty: 5
session: 322ei3e0c900	tool difficulty: 4
decision point: Visualizing Traffic	exit feedback: Took a little time to get used to the interface and draw it out, but overall not too difficult.
# of solutions: 2	
time on home: 00:00:27	
time in tool: 00:10:55	
time on info: 00:00:00	
reviewed: yes	
accepted: 1	
rejected: 1	

**Task:** Status of roads

**Description:** The solution shows a green, yellow or red circle to show the status of the upcoming light and lights in the immediate areas. A green, yellow, or red arrow can show their current path and if upcoming red lights or accidents can impact the time of their current path. Text boxes or highlighting of the road can show if there is traffic ahead or an accident.

**Time spent:** 00:10:21  
**Time spent home:** 00:00:48  
**Time spent in tool:** 00:00:01

**Figure 3.5: CrowdDesign worker submission review interface**

In order to be accepted as an example, solutions should meet the following quality criteria:

1. The solution begins to address at least one requirement
2. The solution is not an exact copy of a previous solution
3. The solution is understandable
  - a. It has a drawing that illustrates the idea
  - b. It has a description that explains the drawing

The main objective of these criteria was to prevent workers from seeing completely wrong or unclear solutions. Even if a solution only contained incomplete ideas to address one or more requirements, these ideas could serve as inspiration for other workers who could improve them and possibly conceive better quality, more complete solutions.

Table 3.4 summarizes submissions review. Out of 70 submissions, 45 (64%) had all solutions accepted as examples, 11 (16%) had all solutions rejected as examples and 14 (20%) had both accepted and rejected solutions. Out of the 11 totally rejected submissions, three workers (4%) demonstrated no honest effort and therefore were not compensated.

	# Workers	%
<b>Workers who got all solutions accepted as examples</b>	45	64%
<b>Workers who got all solutions rejected as examples</b>	11	16%
<b>Workers who got both accepted and rejected solutions</b>	14	20%
<b>Total</b>	<b>70</b>	<b>100%</b>

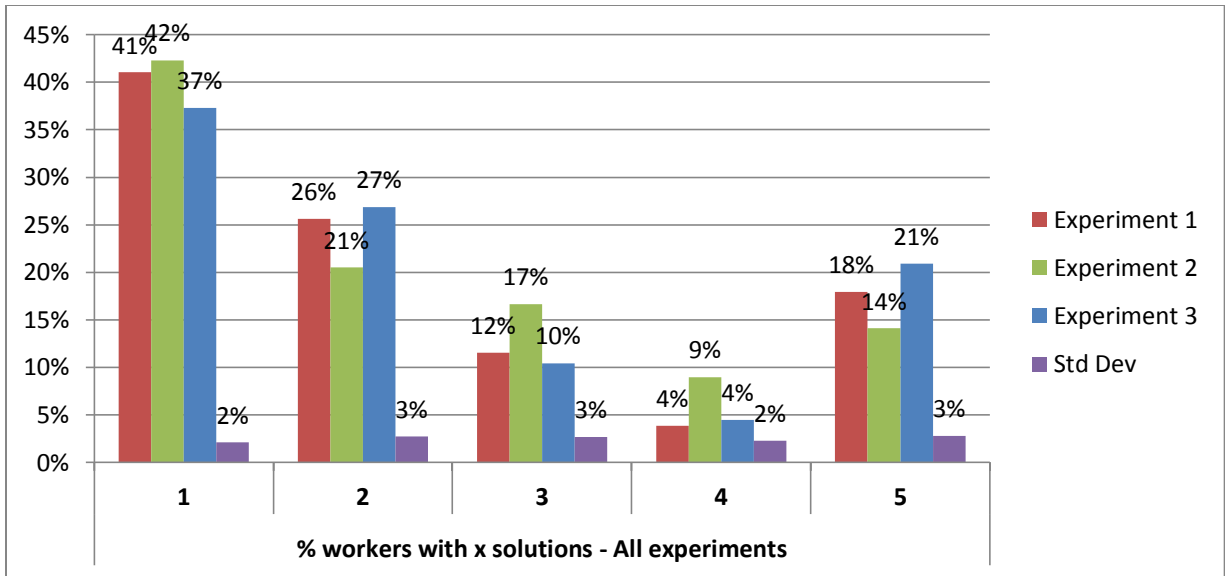
**Table 3.4: Workers submissions review summary**

## 4 DATA ANALYSIS

All submitted solution alternatives that demonstrated honest effort and for which workers were compensated were analyzed in terms of quantity, diversity of ideas, quality, requirements fulfillment, task difficulty perceived by workers and how often workers borrow ideas from others. Results were compared to two previous experiments that asked workers from Amazon Mechanical Turk to engage with the same set of user interface design tasks. The first experiment, henceforth called **Experiment 1**, displayed no examples of existing designs to workers. The second experiment, henceforth called **Experiment 2**, displayed all compensated solutions as examples and did not employ any type of quality control.

### 4.1 Quantity

Figure 4.1 compares the distribution percentages of produced solution alternatives in all experiments. We can see that there were no major differences.



**Figure 4.1: Distribution percentages of produced solutions in all experiments**

In our experiment, 67 workers produced 164 solution alternatives for which they were compensated, an average of 2.4 per worker. Out of these, 59 workers produced 123 solution alternatives that passed the quality review and were displayed as examples to other workers, an average of slightly below two per worker. Table 4.1 shows the distribution of how many compensated solution alternatives were produced by workers. We can see that the majority of workers (64%) submitted only one or two compensated solution alternatives, though a relevant portion (21%) submitted all five solutions.

Decision Point	# workers with x solutions compensated					total solutions
	1	2	3	4	5	
Map building	5	3	2	1	6	51
Traffic visualization	9	7	1	0	3	41
Setting traffic lights timings	4	5	1	2	3	40
Traffic flows	7	3	3	0	2	32
<b>Total</b>	<b>25</b>	<b>18</b>	<b>7</b>	<b>3</b>	<b>14</b>	<b>164</b>
<b>% of total</b>	<b>37%</b>	<b>27%</b>	<b>10%</b>	<b>4%</b>	<b>21%</b>	

**Table 4.1: Distribution of compensated solutions per decision point**



Table 4.2 shows the same distribution for solution alternatives that were displayed as examples only. The majority of workers (63%) also submitted only one or two of such solution alternatives, though a lower portion (7%) had all five submitted solutions shown as examples.

Decision Point	# workers with x solutions accepted					total solutions
	1	2	3	4	5	
Map building	6	3	1	1	3	34
Traffic visualization	9	7	1	1	2	40
Setting traffic light timings	6	4	3	1	0	27
Traffic flows	5	2	3	1	0	22
<b>Total</b>	<b>26</b>	<b>16</b>	<b>8</b>	<b>4</b>	<b>5</b>	<b>123</b>
<b>% of total</b>	<b>44%</b>	<b>27%</b>	<b>14%</b>	<b>7%</b>	<b>8%</b>	

Table 4.2: Distribution of accepted solutions per decision point

Table 4.3 compares the number of solutions and workers in all experiments. Though our experiment had slightly fewer workers, the average number of solution alternatives each worker submitted was nearly identical ( $p = 0.86$ ).

Experiment	total solutions	total workers	Avg solutions / worker
Experiment 1	181	78	2.3
Experiment 2	187	80	2.3
Experiment 3	164	67	2.4

Table 4.3: Number of solutions and workers in all experiments

## 4.2 Diversity

In order to examine the diversity of ideas in solution alternatives, all solution alternatives were printed and grouped through the use of affinity diagramming [69], for each decision point. Three researchers unrelated to the project as well as the author of this

thesis iteratively grouped solution alternatives that seemed alike, creating categories representing conceptually different solutions. For example, “Click and drag”, “Grid Drawing”, and “Blocks” were identified as different approaches to address the map building decision point.

Table 4.4 presents the number of identified solution categories per decision point in all experiments. Overall, the number of categories is quite high in each of the experiments ( $p = 0.13$ ). It is interesting to note that experiment 1 and our experiment had the same number of categories (48), though our experiment had a higher average of categories per workers (0.72). Experiment 2 had fewer categories (37). The results suggest that showing bad examples to workers reduces solution diversity as compared to when no examples are displayed at all. On the other hand, showing examples of sufficient quality only did not impact diversity when compared to not displaying examples.

Decision point	Experiment 1	Experiment 2	Experiment 3	Std Dev
Map building	11	11	12	0.47
Visualizing traffic	10	7	15	3.30
Setting traffic light timings	13	10	12	1.25
Traffic flows	14	9	9	2.36
Total	48	37	48	5.19
Avg / Workers	0.62	0.46	0.72	0.10

Table 4.4: Identified solution categories per decision point in all experiments

Table 4.5 through Table 4.8 show the categories of each worker’s solutions, with each table representing one of the four decision points. Each category is highlighted with a unique color and solutions are displayed in the order they were submitted, from left to right. For example, in Table 4.5, worker MB1 produced two solution alternatives: the first

one was categorized as “*Drag and Drop Only*” and the second was categorized as “*Click and Drag*”.

Worker Id	Unique Categories	Created New Categories?	Solution Categories - Map Building					
			Drag and Drop Only	Click and Drag	Grid Drawing	UI layout and extra features	Drag and Drop Only	UI layout and extra features
MB1	2	Yes	Drag and Drop Only	Click and Drag				
MB2	3	Yes	Grid Drawing	UI layout and extra features	Drag and Drop Only	UI layout and extra features	Drag and Drop Only	
MB3	2	No	Drag and Drop Only	Grid Drawing				
MB4	4	Yes	Assisted Drawing	Isolated Road Properties	Drag + Resize	Touch Grid		
MB5	1	No	UI layout and extra features					
MB6	1	No	Grid Drawing					
MB7	3	No	Assisted Drawing	Assisted Drawing	Grid Drawing	Click and Drag	Grid Drawing	
MB8	2	No	Isolated Road Properties	UI layout and extra features	Isolated Road Properties	Isolated Road Properties	UI layout and extra features	
MB9	1	No	Grid Drawing					
MB10	1	No	Drag and Drop Only					
MB11	4	No	UI layout and extra features	UI layout and extra features	Isolated Road Properties	Assisted Drawing	Click and Drag	
MB12	3	Yes	Blocks	Grid Drawing	Pencil-like (draw)			
MB13	2	No	Grid Drawing	Assisted Drawing				
MB14	1	Yes	Fixed Nodes					
MB15	2	Yes	Map Only	Isolated Road Properties	Map Only	Map Only	Isolated Road Properties	
MB16	1	No	Map Only	Map Only	Map Only			
MB17	4	No	Drag and Drop Only	Pencil-like (draw)	Touch Grid	Assisted Drawing	Pencil-like (draw)	

**Table 4.5: Solution categories per worker - Map building decision point**

Worker Id	Unique Categories	Created New Categories?	Solution Categories - Visualizing traffic					
VT1	1	Yes	Bars Indicating Queues					
VT2	2	Yes	Color Encoded and Traffic Light Timing and Flow	Color Encoded and Traffic Light Timing and Flow	Color Encoded Only	Color Encoded and Traffic Light Timing and Flow	Color Encoded and Traffic Light Timing and Flow	Color Encoded and Traffic Light Timing and Flow
VT3	2	Yes	Color Encoded Only	GPS				
VT4	1	No	Color Encoded and Traffic Light Timing and Flow					
VT5	2	Yes	Color Encoded Only	Notifications Only				
VT6	1	Yes	First Person Design					
VT7	1	Yes	Flow Rate Graph					
VT8	2	Yes	Route Time Only	Intersection Info and Color Density				
VT9	2	Yes	Traffic Light Rules	GPS				
VT10	3	Yes	Color Encoded Only	Lights + Flow Rule Setting	Tips + Color			
VT11	2	No	GPS	GPS	Tips + Color	GPS	GPS	
VT12	3	Yes	Automated Traffic Lights Timing	Traffic Light Status	Automated Traffic Lights Timing	Notifications Only	Notifications Only	
VT13	2	No	Tips + Color	Notifications Only				
VT14	1	Yes	Notification Threshold + Color					
VT15	2	No	Traffic Light Status	GPS				
VT16	1	No	Intersection Info and Color Density	Intersection Info and Color Density				
VT17	1	No	Color Encoded and Traffic Light Timing and Flow					
VT18	1	No	Intersection Info and Color Density					
VT19	1	No	Tips + Color					
VT20	1	No	Color Encoded Only					

**Table 4.6: Solution categories per worker - Visualizing traffic decision point**

Worker Id	Unique Categories	Created New Categories?	Solution Categories - Setting traffic light timings					
STLT1	1	Yes	Input Boxes for Times	Input Boxes for Times				
STLT2	3	Yes	Sensor Description + Timing Description	Input Boxes for Times	Traffic Lights Ration + Error Prevention	Input Boxes for Times		
STLT3	1	Yes	Map Only	Map Only	Map Only	Map Only	Map Only	
STLT4	2	Yes	Input Boxes for Times	Input Box + Error Prevention				
STLT5	3	Yes	Sensor Description + Timing Description	Sensor Description + Timing Description	Input Boxes for Times	Sensor Description		
STLT6	3	Yes	Input Box + Error Prevention	Input Boxes for Times	Click to Change Light State	Click to Change Light State	Input Boxes for Times	
STLT7	2	Yes	Traffic Light Timing Description	Automated Traffic Lights				
STLT8	2	Yes	Traffic Light Builder (no time)	Sensor Description + Timing Description				
STLT9	5	Yes	Input Boxes for Times	Click to Change Light State	Traffic Light Timing Description	Sensor + Traffic Lights + Error Prevention	Sensor + Timing + Traffic Load	
STLT10	1	No	Traffic Light Timing Description					
STLT11	1	No	Input Box + Error Prevention					
STLT12	3	No	Sensor + Timing + Traffic Load	Sensor Description + Timing Description	Traffic Light Builder (no time)			
STLT13	1	No	Sensor Description + Timing Description					
STLT14	2	No	Sensor + Timing + Traffic Load	Sensor Description + Timing Description				
STLT15	1	No	Sensor Description + Timing Description					

**Table 4.7: Solution categories per worker - Setting traffic lights timings decision point**

Worker Id	Unique Categories	Created New Categories?	Solution Categories - Traffic flows					
TF1	1	Yes	Entry/Exit Points					
TF2	1	Yes	Traffic Light					
TF3	2	Yes	Input Direction + Flow on Map	Input Direction + Flow on Map	Input Direction + Flow on Map		Traffic Light	Traffic Light
TF4	1	Yes	Map Only					
TF5	2	Yes	Traffic Light	Flow Only				
TF6	1	Yes	Car Behavior	Car Behavior				
TF7	1	Yes	Advanced Settings Menu for All Map Elements	Advanced Settings Menu for All Map Elements	Advanced Settings Menu for All Map Elements			
TF8	2	Yes	Traffic Light	Unknown Map Settings	Unknown Map Settings		Traffic Light	Traffic Light
TF9	1	No	Advanced Settings Menu for All Map Elements					
TF10	2	No	Map Only	Input Direction + Flow on Map				
TF11	1	No	Advanced Settings Menu for All Map Elements					
TF12	2	Yes	Car + Route Definition	Flow Only	Car + Route Definition			
TF13	1	No	Advanced Settings Menu for All Map Elements					
TF14	1	No	Car + Route Definition					
TF15	2	No	Input Direction + Flow on Map	Map Only	Map Only			

**Table 4.8: Solution categories per worker - Traffic flows decision point**

Table 4.9 summarizes the generation of solution categories by workers across all three experiments. In our experiment, workers produced 1.8 conceptually different solutions on average and 35 workers (52%) were responsible for creating new categories. This represents an improvement when compared to experiments 1 and 2, in which workers produced 1.6 and 1.7 conceptually different solutions ( $p = 0.39$ ), respectively, and 49% and 34% of the workers were responsible for creating new categories, respectively. Results suggest that bad examples discourage workers from producing new categories. Although good examples made workers produce more categories in our experiment than in

experiment 2, they still produced the same number of categories when compared to experiment 1.

Experiment	Avg Unique Categories / Worker	# Workers Created New Categories	% Workers Created New Categories
Experiment 1	1.6	38	49%
Experiment 2	1.7	27	34%
Experiment 3	1.8	35	52%
Std Dev	0.1	4.6	8%

Table 4.9: Generation of solution categories by workers in all experiments

### 4.3 Quality

An independent panel of four reviewers assessed the quality of solution alternatives. It was composed of the two researchers who conducted experiments 1 and 2, and two other researchers unrelated to the project. All reviewers had a background in user interface design and three of them were extensively familiar with the traffic simulation design problem. They gave a score from one (lowest quality) to seven (highest quality) for each one of the 164 compensated solution alternatives. They individually scored solutions in terms of **understandability** (is the solution clear?), **feasibility** (is the solution technically viable?), and **usability** (is the solution intuitive for the users?). The reviewer's quality score for a solution alternative was calculated from the average of these individual scores. A solution alternative final quality score was then obtained by calculating the average of all reviewers' scores.

Table 4.10 presents the distribution of quality scores per decision point. For each decision point, solution alternatives are counted across the score range of 0-1 to 6-7. It is

readily observed that there are no solutions of the highest quality and that only a few were rated 5-6 (3%). The average of 2.8 indicates that overall quality is medium to low.

Decision point	# solution alternatives with quality x-y							Average quality
	0-1	1-2	2-3	3-4	4-5	5-6	6-7	
Map building	6	7	11	13	12	2	0	3.0
Visualizing traffic	2	14	13	8	2	2	0	2.6
Setting traffic lights timings	6	8	4	17	5	0	0	2.7
Traffic flows	7	6	5	9	4	1	0	2.7
<b>Total</b>	<b>21</b>	<b>35</b>	<b>33</b>	<b>47</b>	<b>23</b>	<b>5</b>	<b>0</b>	<b>2.8</b>
<b>%</b>	<b>13%</b>	<b>21%</b>	<b>20%</b>	<b>29%</b>	<b>14%</b>	<b>3%</b>	<b>0%</b>	

Table 4.10: Distribution of quality scores per decision point

Table 4.11 compares the quality score distribution in all three experiments. We can see that the distribution was generally similar, though there were a few noteworthy differences. Experiment 2 had a higher percentage of solutions rated 0-1 (34%), while having less solutions rated 3-4 (10%). Our experiment had the lowest percentage of solutions rated 5-6 (3%). As for the average quality scores ( $p < 0.0001$ ), experiment 1, in which no examples were displayed to workers, obtained the best average quality (3.2), although our experiment average quality (2.8) was better than that in experiment 2 (2.3). It appears that showing a subset of examples of sufficient quality to workers indeed stimulated them to produce better quality solutions.

Experiment	% solution alternatives with quality x-y							Average quality
	0-1	1-2	2-3	3-4	4-5	5-6	6-7	
Experiment 1	11%	22%	14%	26%	24%	11%	2%	3.2
Experiment 2	34%	29%	18%	10%	16%	6%	1%	2.3
Experiment 3	13%	21%	20%	29%	14%	3%	0%	2.8
Std Dev	11%	3%	3%	8%	4%	3%	1%	0.4



**Table 4.11: Distribution of quality scores in all experiments**

Table 4.12 compares the quality score distribution in our experiment for all solution alternatives and for solution alternatives accepted as examples only. The average quality of examples is better when analyzed separately (3.2) and matches the average quality score of solution alternatives in experiment 1 (as showed in Table 4.11). We can see that examples of scores 0-1 and 1-2 are less frequent (2% and 15%, respectively) when compared to all solutions (13% and 21%, respectively). Further, examples of scores 3-4 and 4-5 (37% and 19%, respectively) are more frequent. Still, the quality improvements found in the examples are only modest. This is not surprising because the quality review criteria only rejected solution alternatives of extremely low quality that did not even begin to address the design task appropriately.

Solution Alternative	# solution alternatives accepted as examples with quality x-y							Average quality
	0-1	1-2	2-3	3-4	4-5	5-6	6-7	
All	13%	21%	20%	29%	14%	3%	0%	2.8
Examples only	2%	15%	22%	37%	19%	4%	0%	3.2

**Table 4.12: Distribution of quality scores in our experiment**

Table 4.13 shows a correlation between workers occupations and quality scores. Since the majority of workers identified themselves as hobbyists (38, 54%) and some occupations, such as graduate students, consisted of only a few workers (2, 3%), it is not possible to conclude whether different occupations affect quality. Professional UI/UX designers and graduate students had the highest scores (3.50 and 3.42, respectively),

although only three workers listed these occupations. Undergraduate students and professional software developers had the lowest scores (2.31 and 2.37, respectively).

Worker Occupation	# Workers	# Solutions	Avg Quality
Hobbyist	38 (54%)	98	2.85
Other	12 (17%)	16	2.90
Professional Software Developer	9 (13%)	20	2.37
Undergraduate Student	8 (12%)	22	2.31
Graduate Student	2 (3%)	4	3.42
Professional UI/UX Designer	1 (1%)	4	3.50
		<b>Std Dev</b>	<b>0.46</b>

Table 4.13: Correlation between worker occupation and quality scores

#### 4.4 Completeness

The reviewers who gave quality scores also judged whether solution alternatives fulfilled their requirements. They had to indicate if each requirement was met by choosing **yes** or **no** (we did not include “partial” since each requirement was small, clear, and explicitly stated). If more than 50% of the reviewers were in agreement (i.e., 3 or more) that a requirement was met, we counted it as such. If two or fewer, we counted it as not met. A solution alternative final completeness score was then obtained by simply counting how many requirements out of the four it fulfilled.

Table 4.14 presents the distribution of the number of requirements met per decision point. The majority of the solutions (67, 41%) did not fulfill any requirements, while almost half of them met 2 (32, 20%) to 3 (44, 27%) requirements. On average, workers fulfill less than half of requirements (1.6).

Decision point	# requirements met					Average completeness
	0	1	2	3	4	
Map building	15	2	4	28	2	2.0
Visualizing traffic	21	2	9	6	3	1.2
Setting traffic lights timings	16	3	12	4	5	1.5
Traffic flows	15	1	7	6	3	1.4
<b>Total</b>	<b>67</b>	<b>8</b>	<b>32</b>	<b>44</b>	<b>13</b>	<b>1.6</b>
<b>%</b>	<b>41%</b>	<b>5%</b>	<b>20%</b>	<b>27%</b>	<b>8%</b>	

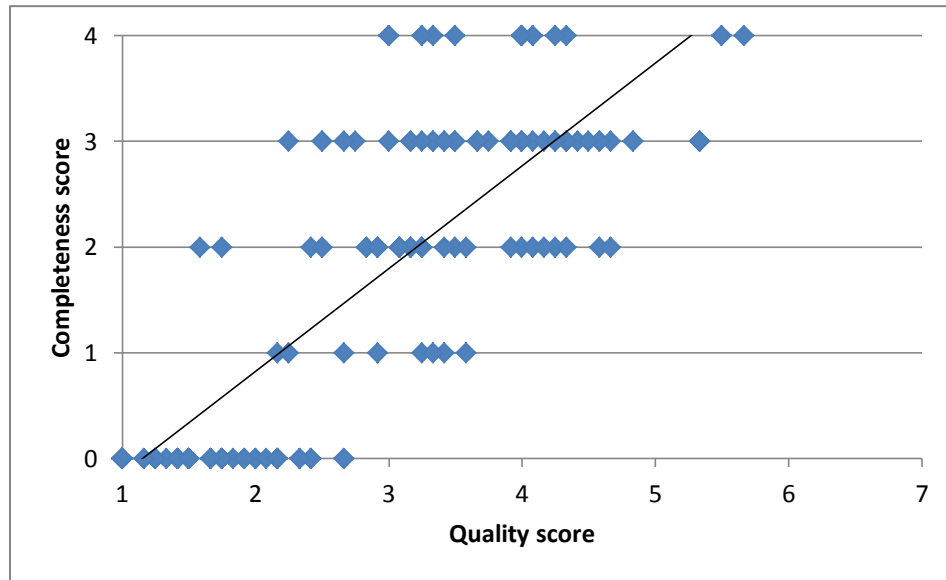
Table 4.14: Distribution of the number of requirements met per decision point

Table 4.15 shows the distribution of completeness scores for all three experiments. We can see that experiment 1 has the lowest percentage of solutions with 0 or 1 requirements met (10% and 22%, respectively) when compared to experiment 2 (27% and 27%, respectively) and our experiment. (41% and 5%, respectively) A hypothesis of this shift from experiment 1 to the other experiments could be that examples encourage incomplete solutions. Workers could be borrowing from examples without validating that they address the task requirements. As for the average scores, experiment 1 had the highest (2.3) while experiment 2 and our experiment had the same (1.6), a difference of 0.7 ( $p < 0.0001$ ). These numbers suggest that examples are hindering workers from fulfilling requirements, regardless whether all solutions are shown as examples or only those which were deemed of sufficient quality.

Experiment	% requirements met					Average completeness
	0	1	2	3	4	
Experiment 1	10%	22%	15%	39%	14%	2.3
Experiment 2	27%	27%	15%	27%	5%	1.6
Experiment 3	41%	5%	20%	27%	8%	1.6
Std Dev	13%	9%	2%	6%	4%	0.3

Table 4.15: Distribution of requirements met in all experiments

Completeness scores displayed high correlation to quality scores (Pearson correlation coefficient ( $R$ ) = 0.83). In other words, higher quality solutions also addressed more requirements in general. Figure 4.2 displays this correlation. Quality scores are on the x axis while completeness scores are on the y axis.



**Figure 4.2: Correlation between completeness and quality scores ( $R = 0.83$ )**

Table 4.13 shows a correlation between worker occupation and completeness scores. Workers with other occupations (such as system administrators, researchers, and data analysts) had slightly higher scores (2.17) as compared to hobbyists (1.66). Professional developers and undergraduate students had lower scores (1.15 and 1.09 respectively). We can see that results corroborate the high correlation between quality scores and completeness scores. As observed in Table 4.13, which shows the correlation between worker occupation and quality scores, professional UI/UX designers and graduate

students had the highest number of requirements met (2.25 and 2.00 respectively), although only three workers listed this occupation. On the other end, undergraduate students and professional developers once again had the lowest number of requirements met (1.09 and 1.15, respectively).

Worker Occupation	# Workers	# Solutions	# Reqs Met
Hobbyist	38 (54%)	98	1.66
Other	12 (17%)	16	1.81
Professional Software Developer	9 (13%)	20	1.15
Undergraduate Student	8 (12%)	22	1.09
Graduate Student	2 (3%)	4	2.00
Professional UI/UX Designer	1 (1%)	4	2.25
		<b>Std Dev</b>	<b>0.42</b>

**Table 4.16: Correlation between worker occupation and completeness scores**

## 4.5 Difficulty

Of the 180 workers who came into the tool, 110 (61%) did not finish the HIT. 98 (54%) left the task in-progress and did not provide any feedback, though 12 (7%) took the time to answer a survey asking why they quit. Of those who did provide feedback, 10 expressed that the task was not clear or too hard. The remaining two workers provided different reasons for quitting. One commented: *“I quit the task because I realized I would not have enough time to complete it because I did not check the time when I hit accept on MTurk.”*. The other commented: *“I’m a programmer, not a U.I. designer, and I have absolutely no idea how your program is supposed to actually work anyway.”*.

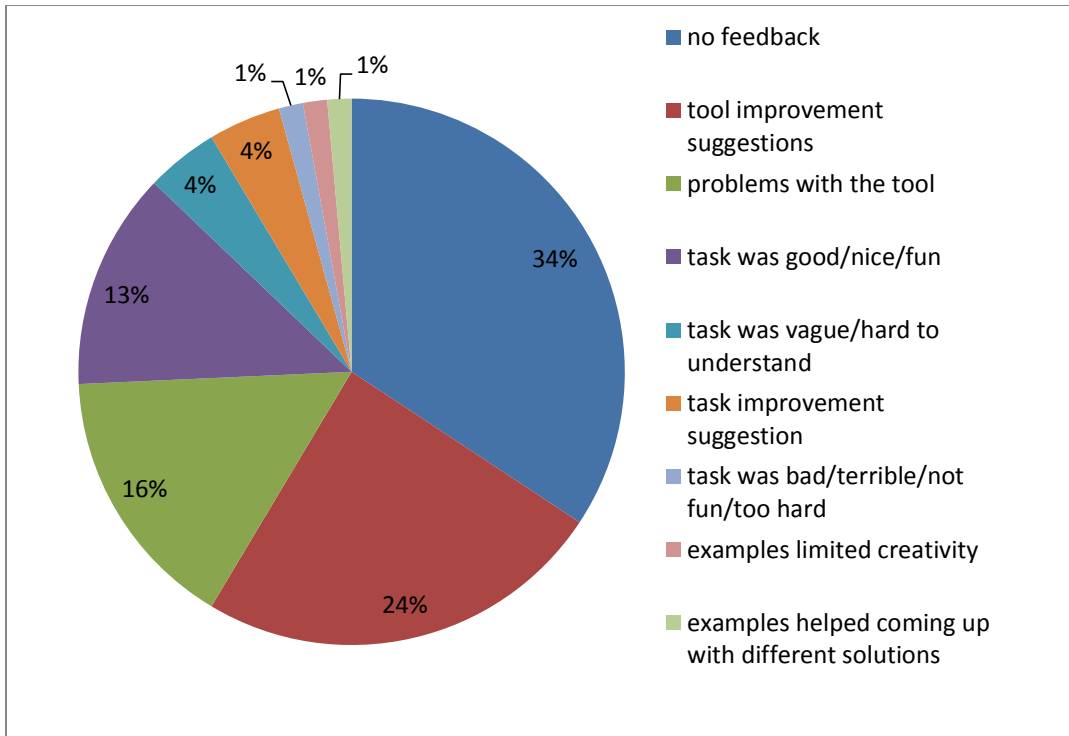
Table 4.17 shows the number of workers who did not complete the task across all three experiments. Even though the number of workers who accessed the tool varies between experiments, the percentage of workers who quit leaving no reasons is exactly the

same in all experiments (54%). As for workers who quit and took the survey, experiment 2 and our experiment share identical percentages (7%) while experiment 1 was slightly higher (15%). In total, the percentage of workers who quit the task is a little lower in the experiments that displayed examples (61%) when compared to experiment 1 (69%).

Experiment	# workers accessed tool	# workers left - no reason	# workers left - took quit survey	# workers left - total
<b>Experiment 1</b>	<b>284</b>	153 (54%)	43 (15%)	<b>196 (69%)</b>
<b>Experiment 2</b>	<b>225</b>	122 (54%)	16 (7%)	<b>138 (61%)</b>
<b>Experiment 3</b>	<b>180</b>	98 (54%)	12 (7%)	<b>110 (61%)</b>

**Table 4.17: Number of workers who did not complete the task in all experiments**

As for the workers who completed the task, Figure 4.3 shows the different types of feedback they provided during the exit survey.



**Figure 4.3: Worker feedback**

Table 4.18 compares worker feedback across the experiments. There were only a few noteworthy differences. First, the number of workers who did not provide feedback was higher in experiment 2 (42%). Second, fewer workers reported that the task was hard to understand in experiment 2 (3%) and our experiment (4%), which might suggest that examples help workers in understanding what is expected of them.

Feedback type	Experiment 1	Experiment 2	Experiment 3	Std Dev
no feedback	26%	42%	34%	7%
tool improvement suggestions	25%	19%	24%	3%
problems with the tool	20%	20%	16%	2%
task was good/nice/fun	9%	7%	13%	2%
task was vague/hard to understand	15%	3%	4%	5%
task improvement suggestion	2%	3%	4%	1%
task was bad/terrible/not fun/too hard	3%	0%	1%	1%
examples limited creativity	0%	1%	1%	1%
examples helped coming up with different solutions	0%	2%	1%	1%
Tool was good/nice/easy	0%	2%	0%	1%

Table 4.18: Worker feedback in all experiments

Numerically, workers rated their ability to complete the entire task at a difficulty level of 4.69 (out of 7), the challenge level of the decision point at 4.10 (out of 7), and adequacy of tool support at 4.40 (out of 7). These numbers show that, in workers' opinion, the overall task was not easy and that the tool could be improved, as highlighted in their written feedback. One worker commented that *"if the design tool were better, this would be a lot faster, and you would get more creative design"*, while another said *"I could not erase one precise element, the "undo" function was inefficient. I could not include text next to the drawing as in examples, very confusing"*.

Table 4.19 shows difficulty scores across all three experiments. Once again, there were no significant differences, although workers who were exposed to examples found the task generally easier than those who were not ( $p = 0.0009$  for average task difficulty,  $p = 0.002$  for average decision point difficulty, and  $p = 0.11$  for average tool difficulty).



Experiment	Avg task difficulty	Avg decision point difficulty	Avg tool difficulty
Experiment 1	5.17	4.94	4.72
Experiment 2	4.47	4.47	4.19
Experiment 3	4.69	4.10	4.40
Std Dev	0.29	0.35	0.22

Table 4.19: Difficulty scores in all experiments

During the experiments, we measured the time workers spent using CrowdDesign's sketching features for each solution alternative. Additionally, we measured the overall time they spent, from the moment they entered the tool until they left. For the sketching time, workers took 6:27 minutes in experiment 1, 5:28 minutes in experiment 2, and 5:26 minutes on average in our experiment ( $p = 0.18$ ). For the overall time, workers took 17:31 minutes on average in experiment 1, 15:27 minutes in experiment 2 and 15:24 minutes in our experiment. Workers who were exposed to examples took on average about one minute less to sketch a solution alternative and spent about 2 minutes less in overall.

Figure 4.4 and Figure 4.5 show the correlation between time and quality scores, and between time and completeness scores, respectively. No correlation was found in both cases ( $R = 0.23$  and  $0.15$ , respectively).

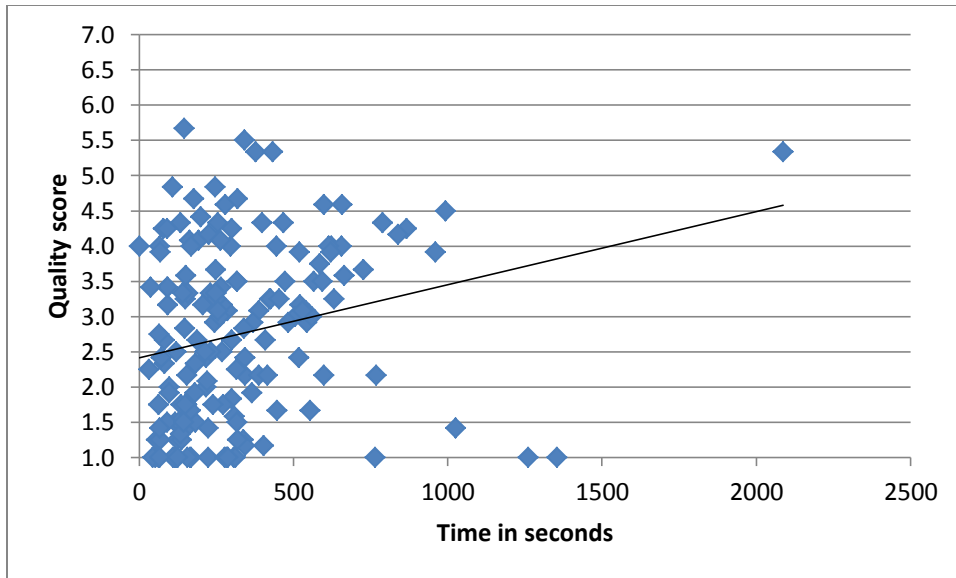


Figure 4.4: Correlation between time and quality scores ( $R = 0.23$ )

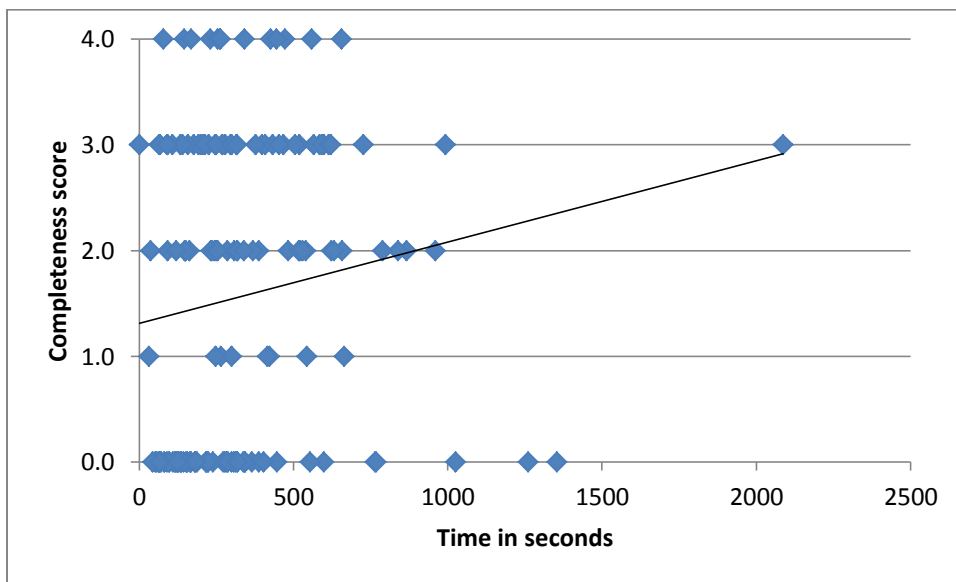


Figure 4.5: Correlation between time and completeness scores ( $R = 0.15$ )

## 4.6 Borrowing Ideas

Participants had the chance to copy or duplicate any of the solutions, partially or wholly, in working on their solution. Table 4.20 shows how many workers used

CrowdDesign’s copy/duplicate features, and compares the quality scores and number of requirements met by the original and destination solutions. Only three workers out of 70 (4%) used the duplicate feature while no workers used the copy feature. In all cases, workers produced inferior designs, of bad quality and not fulfilling any of the requirements. By looking at the solution alternatives, two workers explicitly mentioned that they liked the original idea, although they did not provide any visible improvements and simply drew additional shapes that were unintelligible. The remaining worker seemed to provide an improved solution, but did not include any description that made it clear. It is interesting to observe that all three workers duplicated recent examples and selected reasonably good solutions (4.1 quality and 2.7 completeness on average).

Original Solution			Destination Solution				Quality Score Delta	# Reqs Met Delta
Worker Id	Quality Score	# Reqs Met	Worker Id	Distance From Original	Quality Score	# Reqs Met		
STFL9	3.5	3	STFL12	3	1.7	0	-1.8	-3
TF7	4.2	2	TF8	1	1.0	0	-3.2	-2
STFL5	4.6	3	STFL8	3	1.3	0	-3.3	-3
<b>Average</b>	<b>4.1</b>	<b>2.7</b>	<b>Average</b>	<b>2.3</b>	<b>1.3</b>	<b>0.0</b>	<b>-2.8</b>	<b>-2.7</b>

Table 4.20: Use of CrowdDesign’s copy and duplicate features

Table 4.21 shows the same analysis for experiment 2. 10 workers out of 76 (13%) used copy and duplicate features. As in our experiment, workers did not use the copy function, with worker VT21 being the lone exception. Differently from our experiment, workers produced solutions of identical average quality (2.2) and addressing slightly fewer requirements on average (-0.3). When looking at the original solutions from which workers borrowed, the absolute quality and completeness average scores were worse than in our

experiment (as showed overall in Table 4.20). However, this can be explained by the fact that examples were not reviewed beforehand as in our experiment, and were of all types of quality. Separately, we also note that workers generally borrowed from recent examples, except for MB12, who borrowed from an example contained in the sixth previous submission, and VT21, who borrowed from an example contained in the eighth previous submission.

Original Solution			Destination Solution				Quality Score Delta	# Reqs Met Delta
Worker Id	Quality Score	# Reqs Met	Worker Id	Distance From Original	Quality Score	# Reqs Met		
MB1	1.0	1	MB2	1	1.0	0	0.0	-1
MB3	1.0	0	MB4	1	1.0	0	0.0	0
MB6	1.0	0	MB12	6	1.2	0	+0.2	0
MB9	4.6	3	MB10	1	4.8	3	+0.2	0
MB18	2.5	3	MB21	3	4.1	3	+1.6	0
SL13	1.2	1	SL15	2	1.9	1	+0.7	0
	4.7	2		2	3.4	3	-1.3	+1
VT5	1.7	2	VT7	2	1.4	1	-0.3	-1
VT6	3.7	3	VT8	2	1.9	1	-1.8	-2
	2.3	1	VT10	4	2.0	1	-0.3	0
VT13	1.0	2	VT21 (C)	8	1.8	1	+0.8	-1
Average	2.2	1.6	Average	2.3	2.2	1.3	0.0	-0.3

Table 4.21: Use of CrowdDesign's copy and duplicate features in experiment 2

We also analyzed the similarity between the categories of workers' solutions and of the examples. In order to do so, we calculated the chronological distance between sequential solutions of the same category. For example, if the category "Drag 'n Drop" were identified in a solution from submission #2 and next in a solution from submission #5 of the map building decision point, the distance between these solutions would be three. In

other words, it took three submissions for this category to be identified again in another solution alternative.

Table 4.22 shows the results for all experiments. The distances in experiment 2 and our experiment are mostly between one and three (71% and 66% respectively), which means that workers generally produce solutions of the same categories of recent examples ( $p = 0.001$ ). Distances in experiment 1 have a different pattern, being mostly homogeneous, with exception of distance six, which accounts for 43% of the solutions. A high correlation exists between experiment 2 and our experiment ( $R = 0.8$ ), a low correlation between experiment 1 and our experiment ( $R = 0.09$ ), and a low correlation between experiment 2 and 1 ( $R = 0.06$ ). We believe this might be an indication that workers are indeed borrowing ideas, mostly from recent examples, without explicitly using the copy and duplicate features.

Experiment	Dist = 1	Dist = 2	Dist = 3	Dist = 4	Dist = 5	Dist >= 6
Experiment 1	20 (18%)	13 (12%)	9 (8%)	13 (12%)	9 (8%)	48 (43%)
Experiment 2	47 (33%)	37 (26%)	17 (12%)	10 (7%)	13 (9%)	19 (13%)
Experiment 3	24 (25%)	19 (20%)	20 (21%)	11 (11%)	7 (7%)	15 (16%)
Std Dev	6.13%	5.85%	5.36%	2.14%	0.74%	13.42%

**Table 4.22: Chronological distances of solutions of the same category in all experiments**

Figure 4.6 shows how the average quality and completeness scores progressed over time throughout the experiment as well as workers individual quality and completeness scores. We can see that there were no significant changes over time in general. The average quality started slightly over 3.0 in the beginning of the experiment, quickly dropping to nearly 2.5, and then remaining stable between around 2.7 and 2.9 until the end of the

experiment. The same pattern is observed for the average completeness score, which corroborates once more the high correlation between quality and the number of requirements met.

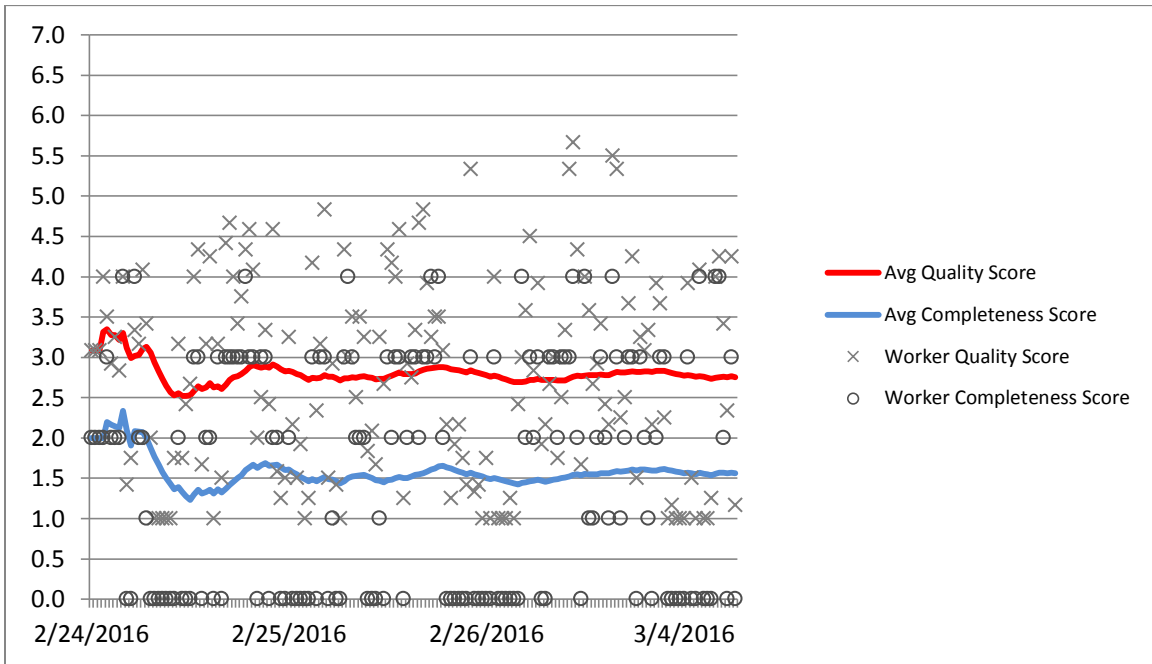


Figure 4.6: Average quality and completeness scores over time

## 5 DISCUSSION

Table 5.1 summarizes the major results of all three experiments, for each analyzed aspect (quantity, diversity, quality, completeness, difficulty, and borrowing ideas), and highlights results from best (green) to worst (red). By analyzing this table, two major findings can be observed.

Result	Experiment 1	Experiment 2	Experiment 3
Avg Solution Quantity	2.3	2.3	2.4
Total Number of Categories	48	37	48
Avg Solution Quality	3.2	2.3	2.8
Avg Solution Completeness	2.3	1.6	1.6
Avg Total Difficulty	14.83	13.13	13.19
Avg Total Time Spent	17:31	15:27	15:24
Borrowing Quality Delta	N/A	0.0	-2.8
Borrowing Completeness Delta	N/A	-0.3	-2.7

Table 5.1: Results summary of all experiments

### 1. Workers who were exposed to sufficient quality examples produced better quality work as compared to workers exposed to all examples

The results in our experiment were equal or better than the results in experiment 2, in almost every aspect. Workers produced 0.1 more solutions on average, a small increase of 4%; 11 more solution categories were identified, 30% more than in experiment 2; the average solution quality score was 0.5 higher, an increase of 22%; the average number of 1.6 requirements met per solution alternatives did not change between the experiments; the total average difficulty scores were almost identical in both experiments, with our experiment being perceived by workers as less than 1% more difficult; and the average

time workers spent on the entire task was similar in both experiments, being just three seconds lower in our experiment. On the other hand, experiment 2 had more workers who borrowed ideas from examples as 13% of its workers (9% more than in our experiment) used CrowdDesign's duplicate and copy features. Workers who copied produced slightly less complete solutions (-0.3 delta), with the same average quality, when compared to the ones from which they borrowed. As for our experiment, the workers who borrowed produced significantly lower quality (-2.8 delta) and less complete (-2.7 delta) solutions. Yet, it is not possible to draw any conclusions regarding how borrowing affects work quality because we consider the number of workers who used the tool's duplicate and copy features not representative enough in both experiments. We suspect that this lack of interest of workers may be an indication that the tool was not intuitive enough, even though there was a tutorial that guided them through the borrowing features. At the same time, they could have just been looking rather than actual copying. Further study needs to be conducted to determine whether this is true, and what the real impact is.

## **2. Workers who were exposed to sufficient quality examples still did not reach the quality of the designs produced by workers who were not exposed to examples at all**

Even though our experiment demonstrated progress over experiment 2, it is still outperformed by experiment 1. In both experiment 1 and our experiment, workers produced an average of 2.3 solution alternatives and had 48 solution categories identified. Our experiment solution alternatives average quality score was 14% lower, a difference of



0.4, and the average completeness was 44% lower, a difference of 0.7. Workers found our experiment easier, with an 11% lower total average difficulty score, and took less time to complete the task, spending 2:31 minutes less on average, which is 12% faster than experiment 1. Interestingly, workers found our experiment easier, yet produced inferior quality work when compared to experiment 1. This seems contradictory, although we suspect that this is a consequence of recruiting workers from Amazon Mechanical Turk, who are known for providing mediocre, but acceptable, work in the least amount of time in order to maximize their financial gains [61, 74].

We conclude that the experiments that showed examples to workers did not help them in producing solution alternatives that were better in terms of quantity, quality and diversity. However, there are two facts that encourage further research on the effects of showing examples to the crowd, rather than concluding that examples should not be displayed at all. First, our experiment yielded results that were equal or better in every analyzed aspect when compared to experiment 2, with the single exception of the usage of the tool's borrowing features. This encourages new experiments that explore different conditions related to the review process, such as defining stricter quality criteria, avoiding displaying too many solutions from same categories, providing better tool support, or adding more reviewers. Second, workers considered the experiment with examples easier and also took less time to complete the task, which is an important factor to attract more people, especially in platforms such as Amazon Mechanical Turk, in which workers are more interested in easier and faster tasks in order to be more productive and, thus, earn more money. Still, there is a considerable challenge in understanding how these workers

can be stimulated to produce better quality work. Even though our experiment had improvements in quality when compared to experiment 2, it still did not reach the quality level of experiment 1.

## 6 LIMITATIONS AND FUTURE WORK

Despite the findings that support the use of examples in microtask crowdsourcing for software design, there are still limitations related to the approach taken that should be addressed in future experiments.

- 1. Crowdsourcing platform.** Workers from Amazon Mechanical Turk are typically interested in maximizing their financial gain through completing as many simple and easy HITs as possible. A plausible consequence of this behavior is that workers may have chosen to not invest a lot of effort in providing high quality solutions that fully addressed the design task, instead submitting mediocre solutions in order to complete the HIT and move on to the next one. A piece of information that corroborates this suspicion is that workers spent in all experiments between 15 to 17 minutes on average to submit their solutions, which is remarkably low. Since we expected workers to spend approximately one hour to submit five solution alternatives (which would be an average of 10-12 minutes per solution alternative), it is clear that they only spent just half of that. Future experiments should recruit workers from different platforms such as TopCoder [7], or even through direct calls in online communities such as Reddit [72], which contains a large number of software professionals and technology enthusiasts. We believe that workers from such platforms have different motivations than those from Amazon Mechanical Turk and may address the design tasks differently. TopCoder workers, for instance, always strive to provide high quality solutions, because of the

competition model TopCoder uses. We believe it could be possible that these workers carry this behavior over to our tasks.

2. **Worker profiles.** The majority of workers (54%) were hobbyists, while the remaining 46% were divided among students, researchers, and professionals. It would be interesting in the future to analyze the attributes of solutions provided by different crowds, especially with more UI/UX designers, which accounted for only 1% of the workers in our experiment. As in the previous item, future experiments should recruit workers from different platforms. It is known, for instance, that TopCoder has a high number of software professionals among its members.
3. **Design problem.** All of the experiments were conducted with the same educational traffic simulation problem. While the researchers from the previous experiments knew this problem in detail and selected the decision points based on existing designs provided by professionals, we cannot discard the possibility that low quality solutions are a consequence of workers' poor understanding of the design task or the problem domain. The opposite could also be true: it is more likely that workers are familiar with traffic elements than other that require more specialized knowledge such as, for instance, space shuttle control. Future experiments should present different problems of similar complexity and study whether solution alternatives attributes differ.
4. **Filtering examples.** The primary objective of filtering examples during the experiment was to prevent bad solutions from serving as examples to workers. Since this was the first experiment that employed a quality review criteria, we purposely designed a

criteria that was not too strict, which only rejected solutions that did not demonstrate honest effort or addressed the design task in a completely wrong manner. Because of that, many of the solutions that were accepted as examples were of mediocre quality, which consequently might have influenced workers to provide generally mediocre quality work, as results suggest. In addition, we found that workers normally borrow from the examples that are first visible (e.g., they tried to not scroll down very far, if at all). In experiment 2 and our experiment, 71% and 66% of workers who borrowed, borrowed from the previous three submissions, respectively. If these examples belong to similar solution categories, it means that workers will not see many diverse solutions and will potentially be less inspired to produce unique solutions. Future experiments should test different review criteria that could either be stricter and/or avoid selecting too many examples of same solution category.

**5. Number of reviewers.** Only one reviewer was responsible for evaluating each submitted solution alternative against the quality criteria. The first problem with this setting is that the reviewer could be biased, being either too strict or too lenient. Having an independent board of reviewers could minimize this effect. Furthermore, having only one reviewer is not scalable. In case future experiments receive a substantially higher number of submissions, which is plausible depending on where workers are recruited from, a single individual will not be able to review all solution alternatives in a timely manner. Lastly, reviewers are prone to make mistakes. Again, a board of independent reviewers would alleviate this problem.

**6. Tool improvements.** 28 workers who submitted solution alternatives (40%) reported having problems using CrowdDesign or reported tool improvements that they believe would make them more productive. Considering that only 46 workers provided feedback, this represents 60% of all feedback received. Workers generally mentioned having issues with the sketching features (*“undo button would undo multiple steps”, “the design tool was a little bit cumbersome. I didn't see any easy way to get back to the tools after I selected an item to delete or copy, etc.”*) or suggested new sketching features (*“I would have liked the ability to use a paint bucket”, “keyboard shortcuts for the design tool and a list of key shortcuts somewhere”* ). Future experiments should address these issues in order to provide a better experience to workers and minimize the possibility of the tool being an obstacle for them to create good solutions. Besides, CrowdDesign could also allow workers to upload designs they produce using external tools they feel more comfortable with.

## 7 CONCLUSION

In this study, we performed an experiment in which a crowd of workers was asked to provide one to five solution alternatives to a set of several small software design tasks. While working on their solutions, workers could see examples of previous designs submitted by other workers. By using a review procedure that filtered out submitted designs, only examples that were of sufficient quality were shared with workers, with the hope of stimulating them to provide higher quality designs. The results were compared to two previous experiments that asked workers to engage in the exact same set of tasks. One of these experiments displayed all examples, regardless of their quality, and the other experiment displayed no examples at all. The results of the three experiments were compared in terms of quantity, diversity of ideas, quality, completeness, perceived task difficulty, and how often workers borrow elements from examples.

Workers were recruited from the Amazon Mechanical Turk microtasking platform and were asked to provide designs for small parts of the user interface of an educational traffic simulation software system. Each design was to address one of four decision points, each exploring a different aspect of the user interface. Workers had to pass a qualification test in order to be eligible for the design task. Those who passed were given access to our CrowdDesign platform, in which they were randomly assigned one of the four decision points and were provided a set of features that enabled them to sketch their designs.

Analysis of the results led to two major findings. While workers who were exposed to sufficient quality examples produced better quality work as compared to workers exposed to all examples, the quality of the designs they produced still did not reach the

quality of the designs produced by workers who were not exposed to examples at all. Additionally, it was found that workers who were exposed to sufficient quality examples found the task easier and spent less time to submit their solutions when compared to the previous experiments. These findings represent a small step toward the broader research agenda we are pursuing and encourage future work that further studies the effects of examples by exploring different experimental conditions, such as employing different design problems, changing the quality review criteria, changing the number of reviewers, among others.



## 8 REFERENCES

- [1] Doan, A. et al., Crowdsourcing Systems on the World-Wide Web. Communications of the ACM, 54(4), 2011, 86–96.
- [2] Jeff Howe, "The rise of crowdsourcing," Wired magazine, vol. 14, no. 6, pp. 1-4, 2006.
- [3] Ricardo Matsumura Araujo, ".99designs: An analysis of creative competition in crowdsourced design," First AAAI conference on Human computation and crowdsourcing, pp. 17-22, 2013.
- [4] Omar F Zaidan and Chris Callison-Burch, "Crowdsourcing Translation: Professional Quality from Non-Professionals," 9th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 1220-1229, 2011.
- [5] Amazon Mechanical Turk: <https://www.mturk.com/>.
- [6] Mao, K. et al. 2015. A Survey of the Use of Crowdsourcing in Software Engineering. Technical Report RN/15/01, University College London (2015).
- [7] TopCoder: <https://www.topcoder.com/>.
- [8] StackOverflow: <https://stackoverflow.com/>.
- [9] uTest, <https://www.utest.com/>.
- [10] Stol, K.-J. and Fitzgerald, B. Two's Company, Three's a Crowd: a Case Study of Crowdsourcing Software Development. ICSE 2014, 187–198.
- [11] Hosseini, M. et al., Towards Crowdsourcing for Requirements Engineering. CEUR Workshop, 2014, 82–87.

- [12] Smith, G. et al. 2012. Concept Exploration Through Morphological Charts: An Experimental Study. *Intl. Conference on Design Theory and Methodology*, 134(5), 2012, 494–504.
- [13] Ulrich, K.T. and Eppinger, S.D. *Product Design and Development* (2003).
- [14] Dym, C. et al., *Engineering Design: a Project-Based Introduction* (2004).
- [15] Greengard, S.: Following the crowd. *Communications of the ACM*. 54, 20–22 (2011).
- [16] D. C. Brabham, “Crowdsourcing as a model for problem solving an introduction and cases,” *Convergence: the international journal of research into new media technologies*, vol. 14, no. 1, pp. 75–90, 2008.
- [17] Schenk, E., & Guittard, C. (2010). Towards a characterization of crowdsourcing practices. *Journal of Innovation Economics*, 7(1), 1-20.
- [18] K. R. Lakhani, D. A. Garvin, and E. Lonstein, “TopCoder(A): Developing software through crowdsourcing,” *Harvard Business School Case*, 610-032, January 2010.
- [19] T. D. LaToza, W. Ben Towne, A. van der Hoek, and J. D. Herbsleb, “Crowd development,” in *Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering*, May 2013, pp. 85–88.
- [20] Chesborough, H.W. (2003). *The era of Open Innovation*, MIT Sloan Management Review
- [21] von Hippel, E. (2005). *Democratizing innovation*, MIT Press.
- [22] von Hippel, E., & von Krogh, G. (2003). Open Source Software and the “Private-Collective” Innovation Model: *Issues for Organization Science*, 14(2), 209-223.
- [23] Brabham, D. (2008). Crowdsourcing as a model for problem solving: An introduction and Cases. *Convergence: The International Journal of Research into New Media Technologies*, 14(1), 75-90

- [24] Gilley, K., & Rasheed, A. (2000). Making More by Doing Less: An Analysis of Outsourcing and its Effects on Firm Performance. *Journal of Management*, 26(4), 763-790
- [25] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popovic, et al., "Predicting protein structures with a multiplayer online game," *Nature*, vol. 466, no. 7307, pp. 756–760, 2010.
- [26] F. Khatib, F. DiMaio, S. Cooper, M. Kazmierczyk, M. Gilski, S. Krzywda, H. Zabranska, I. Pichova, J. Thompson, Z. Popovic, et al., "Crystal structure of a monomeric retroviral protease solved by protein folding game players," *Nature Structural and Molecular Biology*, vol. 18, no. 10, pp. 1175–1177, 2011.
- [27] T. C. Norman, C. Bountra, A. M. Edwards, K. R. Yamamoto, and S. H. Friend, "Leveraging crowdsourcing to facilitate the discovery of new medicines," *Science Translational Medicine*, vol. 3, no. 88mr1, 2011.
- [28] R. Johnson, "Natural products: Crowdsourcing drug discovery," *Nature chemistry*, vol. 6, no. 2, pp. 87–87, 2014.
- [29] D. C. Brabham, T. W. Sanchez, and K. Bartholomew, "Crowdsourcing public participation in transit planning: preliminary results from the next stop design case," *Transportation Research Board*, 2009.
- [30] A. Misra, A. Gooze, K. Watkins, M. Asad, and C. A. Le Dantec, "Crowdsourcing and its application to transportation data collection and management," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2414, no. 1, pp. 1–8, 2014.
- [31] O. Alonso, D. E. Rose, and B. Stewart, "Crowdsourcing for relevance evaluation," in *ACM SigIR Forum*, vol. 42, no. 2. ACM, 2008, pp. 9–15.

- [32] M. Lease and E. Yilmaz, "Crowdsourcing for information retrieval," in ACM SIGIR Forum, vol. 45, no. 2. ACM, 2012, pp. 66–75.
- [33] T. W. Schiller and M. D. Ernst, "Reducing the barriers to writing verified specifications," in Proceedings of the 27th ACM International Conference on Object-Oriented Programming Systems, Languages, and Applications, 2012, pp. 95–112.
- [34] T. D. Breaux and F. Schaub, "Scaling requirements extraction to the crowd: Experiments with privacy policies," in Proceedings of the 22nd IEEE International Requirements Engineering Conference, Aug. 2014, pp. 163–172.
- [35] R. A. Cochran, L. D'Antoni, B. Livshits, D. Molnar, and M. Veanes, "Program boosting: Program synthesis via crowd-sourcing," in Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2015, pp. 677–688.
- [36] M.-A. Storey, C. Treude, A. van Deursen, and L.-T. Cheng, "The impact of social media on software engineering practices and tools," in Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, ser. FoSER '10, 2010, pp. 359–364.
- [37] K. T. Stolee and S. Elbaum, "Exploring the use of crowdsourcing to support empirical studies in software engineering," in Proceedings of the 4th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 2010, pp. 1–4.
- [38] Kickstarter, <https://www.kickstarter.com/>.
- [39] LEGO™ ideas, <https://ideas.lego.com/>.
- [40] iStockPhoto, <http://www.istockphoto.com/>.
- [41] J. Howe, "Crowdsourcing: Why the power of the crowd is driving the future of business," Crown Business, New York, 2008.

- [42] G. D. Saxton, O. Oh, and R. Kishore, "Rules of crowdsourcing: Models, issues, and systems of control," *Information Systems Management*, vol. 30, no. 1, pp. 2–20, 2013.
- [43] T. D. LaToza and A. van der Hoek, "Crowdsourcing in software engineering: Models, motivations, and challenges," *IEEE Software*, vol. 33, no. 1, pp. 74–80, 2016.
- [44] Sarasua, C., Simperl, E., and Noy, N.F. (2012). Crowdmap: crowdsourcing ontology alignment with microtasks. In *The Semantic Web–ISWC 2012*, Edited by P. Cudré-Mauroux, J. Heflin, E.S. Tania, T. Jérôme Euzenat, M. Hauswirth, J.X. Parreira, J. Hendler , 525–541. Berlin: Springer.
- [45] T. D. LaToza, A. Di Lecce, F. Ricci, W. B. Towne, and A. van der Hoek, "Ask the crowd: Scaffolding coordination and knowledge sharing in microtask programming," in *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium on*. IEEE, 2015, pp. 23–27.
- [46] Ipeirotis, P., Provost, F., Wang, J.: Quality management on Amazon Mechanical Turk. In: *Proceedings of the ACM SIGKDD Workshop on Human Computation*. (2010) 64–67.
- [47] Kulkarni, A., Can, M., Hartmann, B.: Turkomatic: automatic recursive task and workflow design for Mechanical Turk. In: *Human factors in computing systems (CHI)*. (2011).
- [48] G. Little, L. Chilton, M.G., Miller, R.: TurKit: tools for iterative tasks on mechanical Turk. In: *Proceedings of the ACM SIGKDD Workshop on Human Computation*. (2009) 29–30.
- [49] P. André, A. Kittur, and S. P. Dow, "Crowd synthesis: Extracting categories clusters from complex data," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 2014, pp. 989–998.

- [50] L. B. Chilton, G. Little, D. Edge, D. S. Weld, and J. A. Landay, "Cascade: Crowdsourcing taxonomy creation," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2013, pp. 1999–2008.
- [51] J. Heer and M. Bostock, "Crowdsourcing graphical perception: using mechanical turk to assess visualization design," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2010, pp. 203–212.
- [52] W. S. Lasecki, J. Kim, N. Rafter, O. Sen, J. P. Bigham, and M. S. Bernstein, "Apparition: Crowdsourced user interfaces that come to life as you sketch them," in Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, 2015, pp. 1925–1934.
- [53] L. Yu and J. V. Nickerson, "Cooks or cobblers?: crowd creativity through combination," in Proceedings of the SIGCHI conference on human factors in computing systems. ACM, 2011, pp. 1393–1402.
- [54] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut, "Crowdforge: Crowdsourcing complex work," in Proceedings of the 24th annual ACM symposium on User interface software and technology. ACM, 2011, pp. 43–52.
- [55] A. Kittur, E. H. Chi, and B. Suh, "Crowdsourcing user studies with mechanical turk," in Proceedings of the SIGCHI conference on human factors in computing systems. ACM, 2008, pp. 453–456.
- [56] N. F. Noy, J. Mortensen, P. R. Alexander, and M. A. Musen, "Mechanical turk as an ontology engineer," Using microtasks as a component of an ontology engineering workflow. Web Sci, 2013.

- [57] A.J. Quinn and B.B. Bederson, "Human Computation: A Survey and Taxonomy of a Growing Field," Proc. 2011 Ann. Conf. Human Factors in Computing Systems, ACM, 2011, pp. 1403–1412.
- [58] U. Gadiraju, R. Kawase, S. Dietze, and G. Demartini. 2015. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In Proceedings ACM Conference on Human Factors in Computing Systems. ACM, 1631–1640.
- [59] J.J. Chen, N. Menezes, and A. Bradley, "Opportunities for Crowdsourcing Research on Amazon Mechanical Turk," Proc. CHI 2011 Workshop Crowdsourcing and Human Computation, 2011.
- [60] G. Kazai, J. Kamps and N. Milic-Frayling, "Worker types and personality traits in crowdsourcing relevance labels," in Proceedings of the 20th ACM international conference on Information and knowledge management. ACM, 2011, pp. 1941–1944.
- [61] M. Allahbakhsh, B. Benatallah, and A. Ignjatovic, "Quality control in crowdsourcing systems," IEEE INTERNET COMPUTING, pp. 76–81, 2013.
- [62] D. Steven, A. Kulkarni, S. Klemmer, and B. Hartmann. 2012. Shepherding the crowd yields better work. Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work, ACM, 1013–1022.
- [63] B.J. Zimmerman. Becoming a self-regulated learner: Which are the key subprocesses? Contemporary Educational Psychology 11, 4 (1986), 307-313.
- [64] J. Chan, S. Dang and S. Down. 2016. Improving Crowd Innovation with Expert Facilitation. Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing, Pages 1223-1235.

- [65] N. Quoc Viet Hung, D. Chi Thang and M. Weidlich. 2015. ERICA: Expert Guidance in Validating Crowd Answers. Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Pages 1037-1038.
- [66] IBM InnovationJam™, <https://www.collaborationjam.com/>.
- [67] PredictIt, <https://www.predictit.org>.
- [68] E. Weidema, C. López, S. Nayebaziz, F. Spanghero and A. van der Hoek. 2016. Toward Microtask Crowdsourcing Software Design Work. Proceedings of the 3<sup>rd</sup> International Workshop on Crowdsourcing in Software Engineering.
- [69] B. Hanington and B. Martin, Universal Methods of Design, 2012.
- [70] C. Lopez. 2016. An Experiment in Microtask Crowdsourcing Software Design. Unpublished master's thesis, University of California Irvine.
- [71] M. Petre and A. van der Hoek. 2013. Software Designers in Action: A Human-Centric Look at Design Work. Chapman and Hall/CRC, 1st edition.
- [72] Reddit: <https://www.reddit.com/>.
- [73] J. Teevan, S. Iqbal and C. von Vech. 2016. Supporting Collaborative Writing with Microtasks. Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. Pages 2657-2668.
- [74] M. Silberman, J. Ross, L. Irani, and B. Tomlinson, "Sellers' problems in human computation markets," Proc. of the ACM SIGKDD Workshop on Human Computation (HCOMP'10), pp. 18-21, 2010.



## 9 APPENDIX A: EXPERIMENT ARTIFACTS

We need your help to create small software interface solutions in our sketching tool.

**This task has 7 simple steps:**

1. Read the consent form.
2. Answer demographic questions.
3. Take the qualification test.
4. Read the design task.
5. Sketch solutions (min 1, max 5).
6. Review and submit your sketches.
7. Complete the exit questionnaire and receive your completion code.

**We won't reject your HIT if you submit at least one sketch that represents a thoughtful solution to the task.**

**Attention:**

- All text must be written in English.
- This HIT can only be done once.

**Technology Requirements:**

- Chrome (v46) or latest version of Safari (OSX version).

**Two bonus criteria:**

- For each sketch that represents a honest attempt to solve the solution you earn an extra \$0.50 bonus.
- In addition, if your sketches covers at least 3 out of 4 of the given requirements you earn an extra \$1.00 bonus for a total of \$1.50.

If you have any question about this HIT please message us.

**Make sure to leave this window open as you complete the task.** When you are finished, you will return to this page to paste the code into the box.

**Survey link:** <http://dellserver.ics.uci.edu:8080/crowddesign/ConsentForm.jsp?UI>

**Provide the task code here:**

Figure 9.1: Amazon Turk HIT prompt page as seen by workers

University of California, Irvine  
Study Information Sheet

Programming Online Study

Faculty Sponsor and Lead Researcher

Professor Adriaan W. van der Hoek  
Department of Informatics  
Donald Bren School of Information and Computer Sciences  
andre@ics.uci.edu  
949-824-6326

- You are being asked to participate in a research study to perform some programming tasks related to software design, coding, debugging, and testing.
- Programming tasks will be performed in an online tool that consists of an external website accessible via a link in a Mechanical Turk task (HIT - Human Intelligent Task).
- The purpose of the study is to better understand the challenges developers face in using tools to answer their questions about code and to help inform the design of new tools that help developers to work more effectively.
- You are eligible to participate in this study if you are at least 18 years of age or older; are fluent in English; and have at least minimal programming skills.
- The research procedures involve using an online software development tool and will last approximately from 5 to 45 minutes.
- There are no risks/discomforts associated with the study. No personal information will be collected.
- There are no direct benefits from participation in the study. However, this study may help us to better understand how programmers work with tools.
- You will be paid the equivalent of 9 dollars per hour, which is California minimal wage, prorated by the expected length of the task to be completed. You will be paid through Amazon Mechanical Turk. At the end of the study, you will be given a code to enter in your HIT (Human Intelligent Task) that confirms that you participated.
- All research data collected will be stored securely and confidentially in encrypted files. At the end of the study, the original answers to demographics questions will be deleted from our files.
- The research team and authorized UCI personnel may have access to your study records to protect your safety and welfare. Any information derived from this research project that personally identifies you will not be voluntarily released or disclosed by these entities without your separate consent, except as specifically required by law.
- If you have any comments, concerns, or questions regarding the conduct of this research please contact the researchers listed at the top of this form.
- Please contact UCI's Office of Research by phone, (949) 824-6662, by e-mail at IRB@research.uci.edu or at 5171 California Avenue, Suite 150, Irvine, CA 92617 if you are unable to reach the researchers listed at the top of the form and have general questions; have concerns or complaints about the research; have questions about your rights as a research subject; or have general comments or suggestions.
- Participation in this study is voluntary. There is no cost to you for participating. You may refuse to participate or discontinue your involvement at any time without penalty. You are free to withdraw from this study at any time. If you decide to withdraw from this study you should notify the research team immediately by clicking on the "No, thanks" button below.

By checking this box I hereby state that "I have read the study information sheet and want to proceed with this study".

No, thanks

Yes, I want to participate

Figure 9.2: Worker consent form

Thank you for your interest in CrowdDesign and for helping us finding out better ways to design software by using the power of the crowd.

Please let us know a little bit more about you. This will help us to design future tasks.

I am currently a:

- Professional software developer
- Professional UI/UX designer
- Graduate student
- Undergraduate student
- Hobbyist
- Other

How many years of experience do you have?

Where did you learn your skills (mark all that apply)?

- High school
- College/University
- On the web
- Other

What is your gender?

- Female
- Male
- Other
- Prefer not to tell

What is your age?

What is your country of residence?

Figure 9.3: Worker demographics form

Before we allow you to continue, we need to evaluate your skills. Please answer the following questions.

1 - Even better than good error messages is a careful design, which prevents a problem from occurring in the first place. Which of the following statements is not an example of preventing errors

- Google auto-complete
- Amazon's "you might also like" product recommendations
- Autocorrect of grammar and spelling
- Conflicting buttons, like "cancel" and "submit", are clearly kept separated

2 - Placeholder text is used in text fields as a temporary solution until a proper value or variable can be assigned. Which of the following statements about placeholder text fields are correct?

I. Placeholder text within a field should be easy to replace

II. The word "default" is a meaningful and responsive term to put in a default text field

- I is correct and II is false
- I is false and II is correct
- Both I and II are correct
- Both I and II are false

3 - When users need to read text in your application it is important that it presented in a way that does not harm its readability. Which of the following could harm the readability of text in your application

- Text with a high contrast
- Text that is colored to stand out
- Unique labels for menu's and buttons
- Font sizes that are large enough to be readable on standard displays

4 - When considering design principles which of the following is true

- Simplicity comes before usability
- Discoverability comes before consistency
- Efficiency comes before learnability
- None of these are true

5 - While using your website an user clicks a button but unfortunately an error occurs. What should your website show to the user

- The user gets a detailed message containing all the information about the error including the stack trace and an option to retry the action
- The user gets shown a blank page with an error code
- The user gets a message that an error occurred and is asked to try again
- The user gets navigated back to the homepage

Quit

Submit answers

Figure 9.4: Qualification test 1

Before we allow you to continue, we need to evaluate your skills. Please answer the following questions.

1 - Giving feedback about the system status to the user is an important part of its design. The system should always keep users informed about what is going on. Which of the following statements is an example of such feedback

- The system sends an email to the user informing about the latest developments
- The system prompts the user for his password when logging in
- The system gives the user a tutorial about how to use a new feature
- The system shows a progress indicator when it is loading

2 - When users need to read text in your application it is important that it presented in a way that does not harm its readability. Which of the following could harm the readability of text in your application

- Text with a high contrast
- Font sizes that are large enough to be readable on standard displays
- Text that is colored to stand out
- Unique labels for menu's and buttons

3 - When considering design principles which of the following is true

- Simplicity comes before usability
- Discoverability comes before consistency
- Efficiency comes before learnability
- None of these are true

4 - When designing an user interface it is important to make your designs consistent. Which of the following statements about consistency in design is correct?

I. Offer users consistent visual cues for a sense of "home"

II. Users do not have to be informed when they face dela

- I is correct and II is false
- I is false and II is correct
- Both I and II are correct
- Both I and II are false

5 - What is a problem, also known as the "Illusion of Simplicity", that can happen when focusing too much on simplicity?

- Creating simplicity will harm the usability
- Hiding complexity, in favour of simplicity, will actually increase it
- Creating optical illusions on your website will mislead the user
- Simplicity improves usage patterns

Quit

Submit answers

Figure 9.5: Qualification test 2

Before we allow you to continue, we need to evaluate your skills. Please answer the following questions.

1 - A good User Interface design can improve the user experience of an application. Which of the following statements about user interfaces are correct?

I. Controls and other objects necessary for the successful use of software have to be visibly accessible at all times

II. Users are capable of learning quickly, therefore after giving instructions once they will not need them again

- I is correct and II is false
- I is false and II is correct
- Both I and II are correct
- Both I and II are false

2 - The aesthetics of an application can influence the user experience a lot. Which of the following principles about aesthetics in design is not true?

Aesthetic design should be left to those schooled and skilled in its application(e.g. graphic and visual designers)

- Fashion should never be put before usability
- Aesthetics should lead the design of software
- Test the visual design as thoroughly as the behavioral design

3 - When considering design principles which of the following is true?

- Efficiency comes before learnability
- Discoverability comes before consistency
- None of these are true
- Simplicity comes before usability

4 - What is a problem, also known as the "Illusion of Simplicity", that can happen when focusing too much on simplicity?

- Creating optical illusions on your website will mislead the user
- Simplicity improves usage patterns
- Creating simplicity will harm the usability
- Hiding complexity, in favour of simplicity, will actually increase it

5 - Giving feedback about the system status to the user is an important part of its design. The system should always keep users informed about what is going on. Which of the following statements is an example of such feedback?

- The system shows a progress indicator when it is loading
- The system prompts the user for his password when logging in
- The system gives the user a tutorial about how to use a new feature
- The system sends an email to the user informing about the latest developments

Quit

Submit answers

Figure 9.6: Qualification test 3

Before we allow you to continue, we need to evaluate your skills. Please answer the following questions.

1 - There are many known common design mistakes. Which of the following is not a design mistake?

- Prompting alert boxes to a user
- Using very small fonts
- Having a confusing navigation
- Not informing the user about a successful submission

2 - What is a problem, also known as the "Illusion of Simplicity", that can happen when focusing too much on simplicity?

- Creating optical illusions on your website will mislead the user
- Simplicity improves usage patterns
- Creating simplicity will harm the usability
- Hiding complexity, in favour of simplicity, will actually increase it

3 - When designing an user interface it is important to make your designs consistent. Which of the following statements about consistency in design is correct?

I. Users should not have to wonder whether different words, situations, or actions mean the same thing.  
II. It is just as important to be visually inconsistent when things act differently as it is to be visually consistent when things act the same

- I is correct and II is false
- I is false and II is correct
- Both I and II are correct
- Both I and II are false

4 - Placeholder text is used in text fields as a temporary solution until a proper value or variable can be assigned. Which of the following statements about placeholder text fields are correct?

I. The word "default" is a meaningful and responsive term to put in a default text field  
II. Placeholder text within a field should be easy to replace

- I is correct and II is false
- I is false and II is correct
- Both I and II are correct
- Both I and II are false

5 - A good User Interface design can improve the user experience of an application. Which of the following statements about user interfaces are correct?

I. Controls and other objects necessary for the successful use of software do not have to be visibly accessible at all times  
II. Users are capable of learning quickly, therefore after giving instructions once they will not need them again

- I is correct and II is false
- I is false and II is correct
- Both I and II are correct
- Both I and II are false

Figure 9.7: Qualification test 4

## Task:

Design an interface mechanism through which users build maps with roads and intersections.

## Sketch solutions that cover the following requirements:

- The user can create a simple visual map of roads on an empty, rectangular canvas.
- The user can create a map that supports at least 6 intersections.
- Roads may only lead to 4-way intersections (3-way intersections are not allowed).
- The user can create a map that allows roads of varying lengths, with different arrangements of intersections.

## Tips:

- You don't need to support very complex maps. Try to focus on the different user interactions your solutions need to have to satisfy the requirements.

## Reminder:

We are not looking for one perfect design but are interested in a variety of designs that each can have their own pro's and con's.

**Figure 9.8: Map building decision point prompt**

## Task:

Design an interface mechanism through which users are informed of the state of traffic and traffic light timings.

## Sketch solutions that cover the following requirements:

- The users must be able to see how their traffic light timings influence the traffic.
- The feedback should inform the user about traffic jams on his/her road system and provide information that helps him/her to avoid these traffic jams.
- The feedback must support a road system with at least 6 intersections and provide both information on the intersection level as on the total road system level.
- Only 4-way intersections are allowed.

## Tips:

- You don't need to support very complex maps. Try to focus on which information the user needs to satisfy all the requirements.
- Think about the different ways you can provide this information to the user.

## Reminder:

We are not looking for one perfect design but are interested in a variety of designs that each can have their own pro's and con's.

**Figure 9.9: Visualizing traffic decision point prompt**



### Task:

Design an interface mechanism through which users set the timing of green, yellow, and red for the traffic lights on an intersection.

### Sketch solutions that cover the following requirements:

- Your intersection should support left hand turns.
- Your solution must avoid letting the user set timings that allow car crashes.
- The intersection should support (optional) use of sensors to detect cars waiting.
- You only have to design a solution for a 4-way intersection.

### Tips:

- Try to focus on what settings the user needs to configure to set traffic lights timings that meet the requirements.
- Think about the different ways the user can manipulate these settings.
- Optionally, you can also think about how your system would work with multiple intersections.

### Reminder:

We are not looking for one perfect design but are interested in a variety of designs that each can have their own pro's and con's.

**Figure 9.10: Setting traffic lights timings decision point prompt**

### Task:

Design an interface mechanism through which users control where traffic goes on a map.

### Sketch solutions that cover the following requirements:

- Your solutions should allow users to control where cars enter and exit on the map, as well as how much traffic flows into the map from each entrance.
- Your solution should support a map of at least six 4-way intersections.
- Your solution should allow user to specify the behavior of the traffic; that is, decide what it does when it encounters a traffic light.
- Your solution should support a large amount of traffic.

### Tips:

- You don't have to support a complex system to drive cars. Try to focus on the settings the user needs to configure the direction of traffic flows.
- Think about the different ways the user can manipulate these settings.

### Reminder:

We are not looking for one perfect design but are interested in a variety of designs that each can have their own pro's and con's.

**Figure 9.11: Traffic flows decision point prompt**

**Exit Questionnaire**  
please answer the following questions

How difficult was it to complete this task?  
very easy -  1  2  3  4  5  6  7 - very hard

How difficult was it to design for this decision point?  
very easy -  1  2  3  4  5  6  7 - very hard

How difficult was it to use this design tool?  
very easy -  1  2  3  4  5  6  7 - very hard

Do you have any feedback to improve this HIT?

[GET YOUR COMPLETION CODE](#)

**Figure 9.12: Exit survey**

## 10 APPENDIX B: EXAMPLES OF DESIGNS CREATED BY THE CROWD

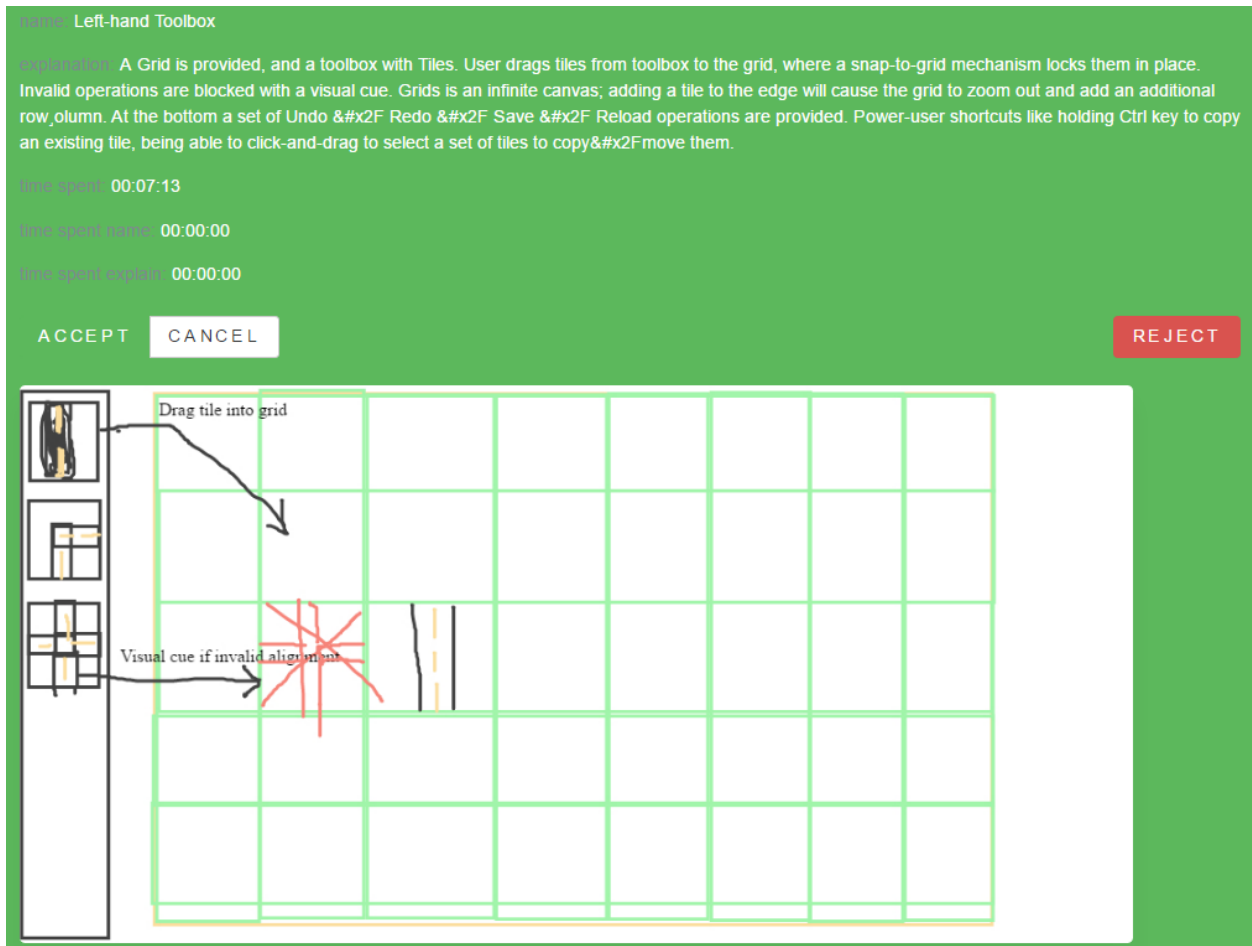


Figure 10.1: Solution alternative accepted as example - Map building

name Highway Through the Country

explanation Is it more efficient to take the highway or the back roads?

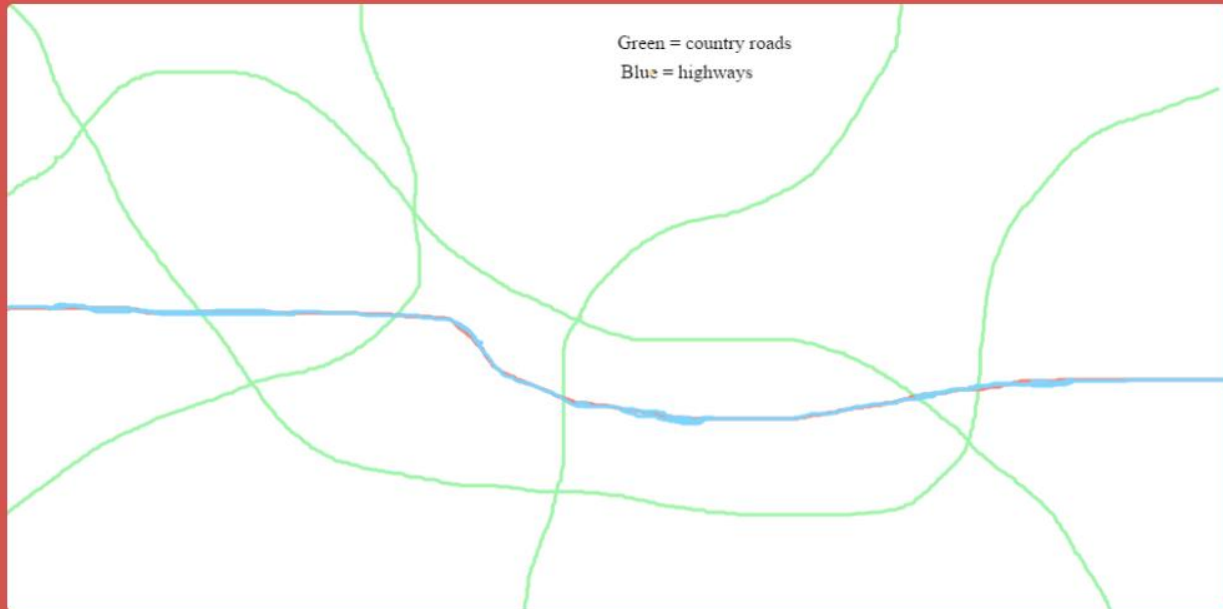
time spent 00:04:48

time spent name 00:00:00

time spent explain 00:00:00

ACCEPT CANCEL

REJECT



**Figure 10.2: Solution alternative rejected as example - Map building**

name Route With Times Per light

description: Program shows the entire route with all 6 intersections, show the main route (S) to (F) and the 6 intersections with their respective delays and time improvements. There is a key for how heavy traffic is (green, yellow, red). At the bottom, you can see how many "lights/intersections" there are in your route and an average of all the time that will be spent in the remaining lights in your route (will update when route is updated/re-routed).

time spent 00:13:09

time spent name 00:00:00

time spent explain 00:00:00

ACCEPT CANCEL

REJECT

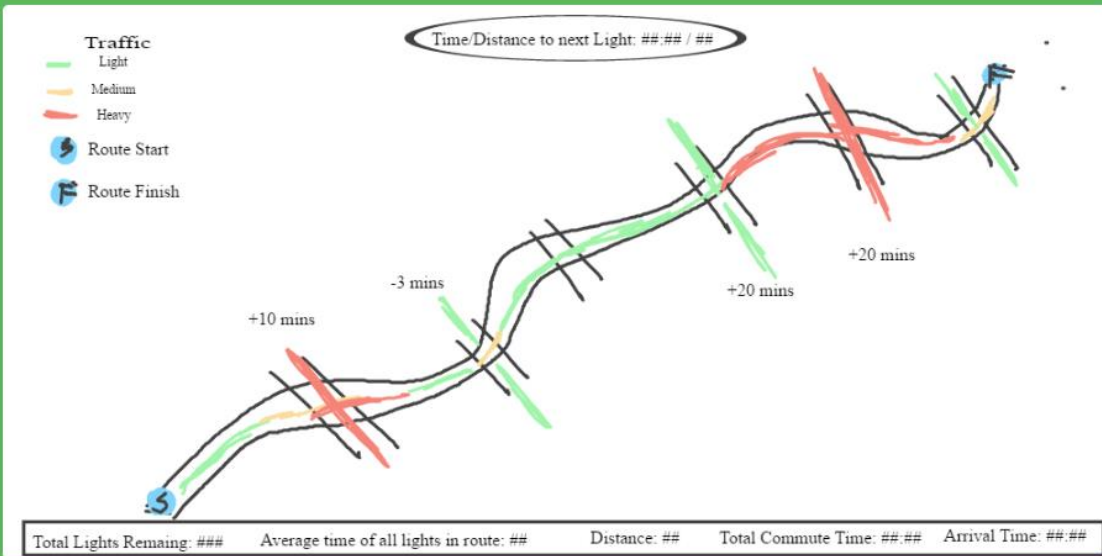


Figure 10.3: Solution alternative accepted as example - Visualizing traffic

name: Signs that encourage motorists

explanation: Since breaking is what causes traffic jams, when streets start to get congested, signs can encourage motors to maintain and steady pace instead of driving so fast that they eventually need to break. These can be the same signs that give motorists live updates.

time spent: 00:00:43

time spent name: 00:00:04

time spent explain: 00:00:00

Please remember:  
Slower drivers go to the left.

Pull over for minor accidents.

It is better to maintain a steady pace then go fast then break.

**Figure 10.4: Solution alternative rejected as example – Visualizing traffic**

There is a camera system mounted on the signal light in the center of the intersection. Each opposing lane (northbound and southbound, and eastbound and westbound) will be green at the same time, while the cross-traffic will be red. Turn lanes are marked in green. When drivers cross the blue sensor plate, a notification shows up on the control panel indicating how many cars are waiting, and also an indicator of how long they have been waiting. The turn lanes will always follow the normal green lights (e.g., when the user wants to allow the cross traffic to go, or allow the left turn lane to go, the normal through traffic will turn from green, to yellow, to red. After it is red, the left turn lane is allowed to go, then the cross traffic will get their turn after the turn lane has ended. There should be a sufficient delay between a light turning red and the next light turning green. The left turn lanes will allow a further buffer of time between the two lanes of cross traffic. Yellow lights will always run for the a fixed period of time (e.g., 5 seconds), but the duration of the green light (and opposing red lights) will be determined by the user. Heavier traffic might warrant a longer green, but a car that has been waiting for a longer time at a red light will begin initiate an indication on the control panel.

Time spent: 00:12:48

Time spent on red: 00:00:00

Time spent on green: 00:00:00

ACCEPT CANCEL

REJECT

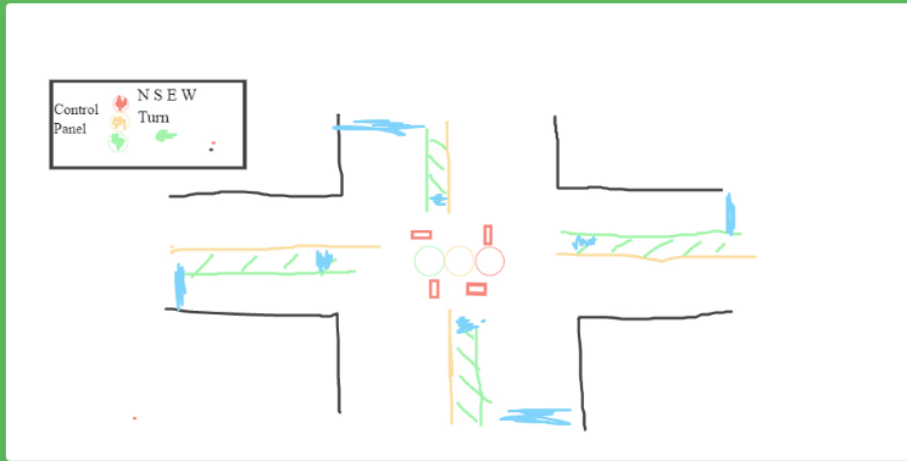
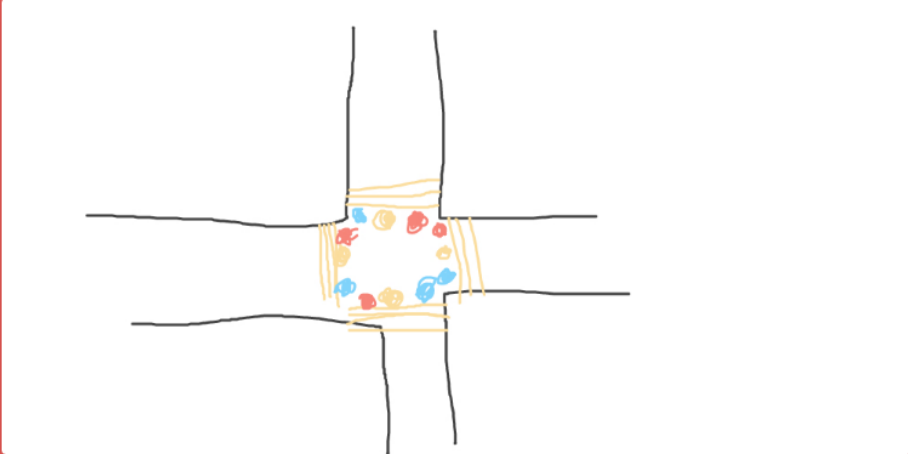


Figure 10.5: Solution alternative accepted as example – Setting traffic lights timings

name: Conventional  
explanation: 4 way intersection with stoplights  
time spent: 00:02:49  
time spent name: 00:00:01  
time spent explain: 00:00:00



**Figure 10.6: Solution alternative rejected as example – Setting traffic lights timings**



name: Touch screen map

description: Touch areas on the map then changes characteristics by entering data on the right.

Time spent: 00:04:23

Time spent on map: 00:00:05

Time spent on input: 00:00:02

ACCEPT CANCEL REJECT

Touch a red circle to modify how much traffic enters the map from that point.

enter amount of traffic in textbox here

Touch a blue x to modify how traffic behaves at that intersection

select right turn/left turn/ect options here

submit button

**Figure 10.7: Solution alternative accepted as example – Traffic flows**

name: Roundabout with lights

explanation: This is a system of 6 roundabouts with added traffic lights. In this way, There is redundancy in the control, and during low usage hours (late night or early morning) the traffic lights turn off to save energy and increase efficiency

time spent: 00:12:44

time spent name: 00:00:04

time spent explain: 00:00:00

ACCEPT CANCEL

REJECT

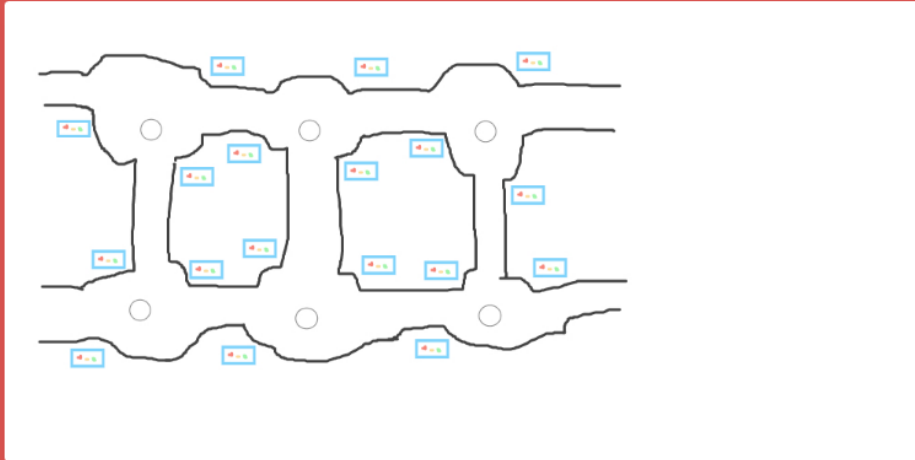


Figure 10.8: Solution alternative rejected as example – Traffic flows

Data on map

Explanation: see diagram

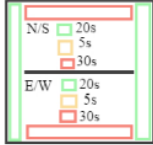
Time spent on map: 00:06:17

Time spent on text: 00:00:01

Time spent on diagram: 00:00:02

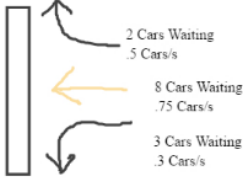
ACCEPT CANCEL REJECT

Each Intersection will be a square, and each side will be color coded to match the current light status. The center of the intersection will contain timing information for that intersection.




Click and Type

At each intersection there would also be 3 arrows indicating the traffic volume waiting to go in separate directions, and the number of cars per second that can make that change.



Depending on the number of cars waiting the arrows can change color to yellow or red


On the side there could also be a quick menu that highlights intersections with high numbers of cars waiting. When these are clicked on it will make the intersection "bounce", making it easy to see which one is having issues and adjusting the time accordingly.





**Figure 10.9: Solution alternative with highest quality score**

name: Two-way traffic  
explanation: Two-way traffic Accounted for  
time spent: 00:00:13  
time spent name: 00:00:03  
time spent explain: 00:00:00

**ACCEPT** CANCEL REJECT

  
Intersections

  
Signals and Signs

  
Directionality

**Figure 10.10: Solution alternative not compensated**