Borrowing from the Crowd: A Study of Recombination in Software Design Competitions

Thomas D. LaToza[1], Micky Chen[2], Luxi Jiang[2], Mengyao Zhao[1], André van der Hoek[1]

[1] Department of Informatics
University of California, Irvine; Irvine, CA, USA
{tlatoza, mengyaz1, andre}@ics.uci.edu

[2] Graduate School of Informatics
University of Amsterdam; Amsterdam, Netherlands
mickychen@yahoo.com, ximielu@gmail.com

***Abstract*—One form of crowdsourcing is the competition, which poses an open call for competing solutions. Commercial systems such as TopCoder have begun to explore the application of competitions to software development, but have important limitations diminishing the potential benefits drawn from the crowd. In particular, they employ a model of independent work that ignores the opportunity for designs to arise from the ideas of multiple designers. In this paper, we examine the potential for software design competitions to incorporate recombination, in which competing designers are given the designs of others and encouraged to use them to revise their own designs. To explore this, we conducted two software design competitions in which participants were asked to produce both an initial and a revised design, drawing on lessons learned from the crowd. We found that, in both competitions, all participants borrowed ideas and most improved the quality of their designs. Our findings demonstrate the potential benefits of recombination in software design and suggest several ways in which software design competitions can be improved.**

***Index Terms*—Crowdsourcing, software design, collective intelligence, collaborative design**

I. Introduction

There is growing interest in the application of crowdsourcing to software engineering. In crowdsourcing, work traditionally done by experts in a single firm is distributed to a large, undefined, distributed group of people in the form of an open call for work [16, 34]. Software engineering has begun to adopt crowdsourcing in several contexts, including open source software development, Q&A sites such as Stack Overflow[1], crowdsourced formal verification games [20], and microtask-based testing websites such as uTest[2].

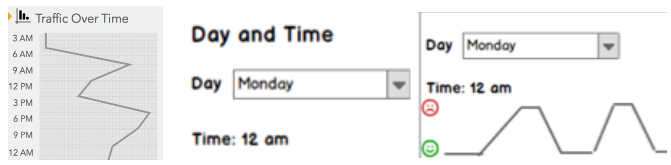
One form of crowdsourcing is the competition, in which participants each independently create a solution and a winner is chosen. Competitions have demonstrated great potential, helping solve problems ranging from rapidly locating balloons dispersed across a country[3] to devising improved social recommendation algorithms[4]. Commercial systems such as TopCoder[5] and 99Designs[6] have begun to explore the application of this model to software development, creating competitions in which workers are asked to author small pieces of code or design user interface elements. However, a recent study of TopCoder found that there are a number of important limitations of current crowdsourcing models, which presume a waterfall process, require clients to be intimately involved, and evaluate quality only late in the process [33].

A central, fundamental limitation of current software competitions is the linear nature of their process: only the best design is used. In crowdsourcing terms, the aggregation mechanism is simply to select a winner. However, in so doing, the diversity of the crowd is thrown away, as only the ideas of a single individual may influence the final design [4]. In contrast, crowdsourcing workflows in other domains have demonstrated that there are important advantages to introducing ideas through recombination. Enabling designers to see the alternative designs of others and iteratively improve their own has been found to increase the creativity of designs [39] and enable designs to grow organically [40]. Moreover, there may be other ways in which increasing communication between individuals within competitions can increase their efficacy. Evaluating designs and selecting a winner imposes a significant burden on clients [33]. Can crowds themselves play a role in evaluation?

Software design is multi-faceted, and ranges from designing the internals of a system to designing user interactions with software. It is well-known that both types of design are needed and influence each other [35]. There are, however, some clear differences including the notations and tools available and the mechanics of explaining 'how it works'. Can a recombination process help support a range of types of design, or are its benefits limited to only certain types of design?

To answer these questions, we conducted two separate software design competitions focusing on (1) user experience design and (2) architecture design. Each participant first independently created an initial design. To explore the potential for recombination, participants were then given several of the competing designs and asked to create a revised design drawing on what they had learned. Finally, participants evaluated the designs of their peers by ranking their revised designs.

We found that, in both competitions, designers were able to evaluate the designs of their peers and borrow ideas from the crowd. Figure 1 shows an example of an improvement made by borrowing an idea. Demonstrating the value of diversity, all designers borrowed, with even the designers of top designs



(a) UE1, Round 1 (b) UE3, Round 1 (c) UE3, Round 2

Fig. 1. Participants borrowed ideas from others such as a 24 hour traffic graph (a), revising their initial designs (b) to adapt and incorporate the ideas (c).

1 http://stackoverflow.com
2 http://www.utest.com/
3 http://archive.darpa.mil/networkchallenge/
4 http://www.netflixprize.com/
5 http://www.topcoder.com/
6 http://99designs.com/

finding ideas in much weaker designs with which they could improve. Overall, the quality of designs in both competitions improved significantly between rounds. Our results also suggest several ways to make design competitions more effective.

II. Related Work

A number of systems have investigated applying ideas from crowdsourcing to software development, seeking to leverage the potential of crowdsourcing to broaden participation, utilize expertise, and reduce time-to-market [19, 33]. In Stack Overflow, developers ask questions, other developers answer them, and yet other developers evaluate the quality of the answers, concurrently curating a knowledge repository of frequent questions [21]. In Collabode, an original programmer requests short programming microtasks which are completed by workers using a shared web IDE [10, 11]. CrowdCode enables crowds of developers to write code, test, debug, and respond to changes through microtasks [18]. Other work has explored the use of crowdsourcing for recommending fixes for bugs [14, 25] and compilation errors [37], and to checking and fixing unit test assertions [26]. To leverage larger pools of workers, some systems enable non-specialists to contribute. For instance, systems have explored gamification of software verification [20] or checking for security vulnerabilities [6].

Due to its increasing prevalence, studies have begun to characterize the impact and success of crowdsourcing approaches on software development. Open source software development is often seen as a form of crowdsourcing, as contributors can come from anywhere in the world, there is a set of tasks that individuals complete to accrue status (e.g., filing bug reports, fixing bugs, specifying new features), and work often happens remotely. Crowston reviews studies of open source software development [5]. A study of Stack Overflow found that crowdsourcing enables questions to be answered fast - in a median of 11 minutes - and that 92% of questions on expert topics are answered [21]. A case study of TopCoder examined the use of software competitions in creating production software to be used by an industry client [33]. It identified several important challenges with the TopCoder competition model, such as its use of a waterfall process, difficulties dealing with complexity and interdependencies, large overhead imposed on the client in preparing specifications and answering questions, and in pushing quality issues late into the lifecycle. Other work has begun to investigate how social networks and trust form in open, online communities for software creation [1].

Psychological theories of creativity emphasize the crucial importance of recombination processes in drawing on many ideas to generate creative ideas [3, 32, 36]. Outside the domain of software, several studies have examined the use of recombination in crowdsourcing workflows. Yu & Nickerson [39] proposed an iterative design process of idea generation, evaluation, and recombination to design a chair for children, demonstrating that creativity increases through recombination. Xu & Bailey [38] found that iteratively authoring design critiques building on the ideas of others enabled quality design critiques to be created fast and accurately. Dow et al. [7, 8] found that separately creating and sharing multiple design prototypes can increase design quality. Other work has investigated remixing communities that allow public repositories of artifacts to be borrowed and adapted by others, finding that adapted artifacts are, on average, not better than the originals [15].

Due to the centrality of sketching to the design process, several tools have investigated ways of supporting and enabling groups of designers to sketch designs together. An early example is Commune, a shared intersurface prototyping tool, supporting distributed design groups [24]. Team Storm enables teams of designers to work efficiently with multiple ideas in parallel [13]. Similarly, Calico enables collaborative, distributed sketching across devices with both synchronous and asynchronous work [22]. IdeaVis [9], a digital pen for paper-based writing, augments traditional sketching to support co-located sketching sessions. Drawing on techniques for controlled brainstorming such as 6-3-5 brainwriting [29] and C-sketch [31], SkWiki enables lightweight branching and merging in collaborative sketch editing, allowing designers in collaborative brainstorming sessions to easily clone and explore changes to existing ideas in parallel [40].

Our study builds on this work, specifically examining if, and in what ways, design competitions with recombination can be used in software design.

III. Method

A. *Study Design*

Two separate but parallel design competitions were conducted: one for software architecture design (AD), and one for user experience design (UX). To observe the process of recombination, each competition consisted of two rounds. Each participants was asked to submit both an initial design (round 1) and revised design (round 2). In round one, participants were provided the design prompt and given one week to produce a design. In between the two rounds, participants were given the opportunity to see the initial designs submitted by other participants in their group and were strongly encouraged to use this as inspiration for their own revised design (i.e., a recombination step). To investigate if performing peer evaluations itself leads designers to more carefully understand designs and adopt ideas more extensively, participants were evenly divided into control and experimental conditions, and participants in the experimental condition were additionally asked to rank the first round designs in their condition. All participants were then given a second week to prepare a revised design, drawing on what they had learned from the other designs. At the conclusion of round two, both control and experimental participants ranked the second round designs of the participants in their group. Figure 2 depicts the structure of the competitions.

B. *Participants*

40 participants (20 per competition) were recruited from computer science, informatics, HCI, and software engineering graduate students at UC Irvine, UC Berkeley, University of Southern California, University of Washington, and Carnegie Mellon University. Participants were only recruited from graduate student populations; the competition was not in any way associated with their course or degree activities, which were many and varied. During the competitions, 10 participants from the AD competition and 8 from the UX competition did not submit the required designs or evaluations and were dropped from the study. We report results only for the final 22 participants. All participants had professional experience in industry,

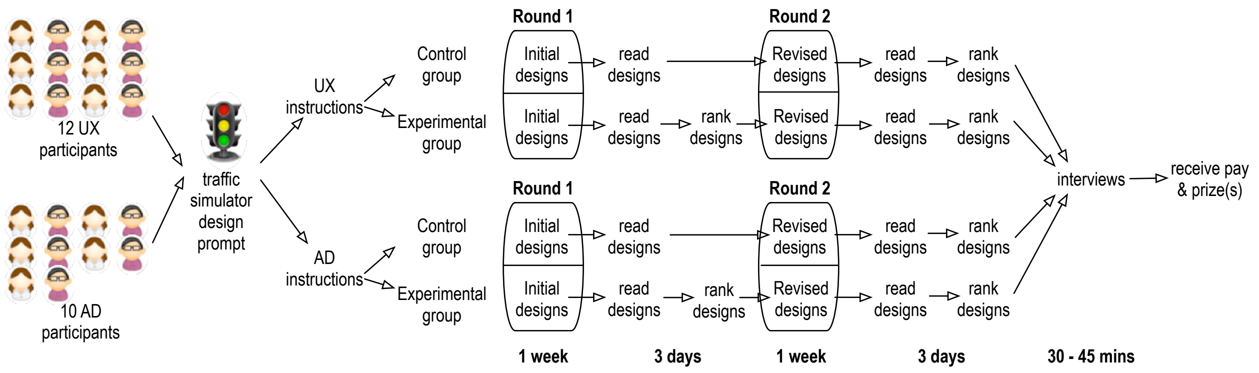


Fig. 2. The structure of the software design competitions.

ranging from 2 to 8 years, with an average of 4.8 years for AD and 3.2 for UX (Table 1). 7 of the 12 UX participants and 3 of the 10 AD participants were female. All participants were paid $100, prorated for those that dropped out. To encourage participants to put forth their best effort throughout the competitions, participants with the winning designs in the first and second rounds of each competition (4 prizes) were awarded $1000.

C. *Tasks*

Participants in both competitions were provided a two page design prompt, used previously in a series of studies of professional software designers [27]. Participants were asked to design an educational traffic flow simulation program to be used by a professor in a civil engineering course to teach students traffic light patterns. The prompt described a set of open-ended goals and requirements, including offering students the ability to: (1) create visual maps of roads, (2) specify the behavior of lights at intersections, (3) simulate traffic flow, and (4) change parameters of the simulation (e.g., traffic density). Participants were asked to produce a design at the level of detail necessary to present "to a group of software developers who will be tasked with implementing it". While the prompt was otherwise identical, participants in the AD competition were instructed to focus solely on the architecture and design of the system, while participants in the UX competition were instructed to focus solely on an interaction design for the user interface. Participants in both competitions were instructed that their design would be evaluated on its elegance, clarity, and completeness. Our materials are publicly available[7].

D. *Procedure*

The study was conducted entirely through electronic communication. Throughout the study, participants communicated only with the experimenters; all other participants were anonymous. At the beginning of the first round, participants were sent an email with the design prompt and given one week to produce an initial design. Participants were allowed to use whichever tools they wished, and simply needed to upload a PDF of their design to Dropbox. Participants were not allowed to exchange ideas and were instructed to work independently. At the conclusion of the one-week period, a recombination step took place. Participants in each of the four groups (e.g., UX control) were given all of the anonymized initial designs in their group and three days to read the designs. Participants in the experimental groups were additionally asked to rank the designs they received. Participants were then given a second one-week period in which to prepare a revised design. At the conclusion of the second one-week period, all participants (both control and experimental) were given three days to read and rank the revised designs within their group. Finally, participants participated in a one-on-one 30-45 minute semi-structured interview with two or three of the authors using Skype. Interview questions focused on processes for peer evaluations, how participants prepared their revised designs, how participants made use of other designs, and their strategy and suggestions for the competition as a whole. After the interviews were concluded, participants were informed whether they had won a prize, and winners were given the option to be publicly recognized on a website. Neither the peer nor expert evaluations were provided to participants during or after the study.

E. *Data Analysis*

All initial and revised designs were first evaluated by a panel, consisting of three authors and one additional panelist. All panelists had a background in design, and three of the panelists had extensive familiarity with the design prompt through past use in a course or research study. Each design was independently scored by each of the four panelists on a 1-7 scale (with 7 being the strongest) for each of the three criteria given to participants: elegance, clarity, and completeness. To enable scores to be compared between rounds and reduce potential bias, the expert panel was double-blind and did not see any of the designs beforehand. All designs were mixed together, and neither the author nor the round of the design was identified. A score for each design was computed by averaging the scores across judges and summing over the three criteria.

Several analyses of the data were performed. To examine predictors of high quality designs, we computed correlations between design scores, designers' years of expertise, and several characteristics of the design using Pearson's correlation coefficient. We performed t-tests to test if designs improved and for an effect of performing ranking in the first round. To evaluate the accuracy of peer and self evaluations, we used Pearson's correlation coefficient to compare rankings between participants and the panel.

To identify aspects of the final designs which were borrowed from others, two authors independently compared participants' revised designs to their initial designs and identified instances in which changes may have been borrowed from others. The findings of each author were then combined into a

[7] http://sdcl.ics.uci.edu/study-materials-and-data/

single matrix identifying potential instances of copying, which were then confirmed and augmented as needed through the interviews with participants.

To systematically identify common ideas and themes in the interviews, we used a qualitative data analysis process. The 22 interviews were first transcribed. Four authors then independently coded the transcripts (2 authors per transcript). Each author identified sections expressing an insight, pasted the section onto an index card, and labeled the index card with the insight, participant, and researcher. The four authors then created an affinity diagram, iteratively grouping similar cards into hierarchic categories (e.g., "Reasons for not incorporating [ideas]", "Time constraints"). Figure 3 depicts a section of the final affinity diagram.

IV. Results

In the following sections, we examine the designs participants created, the relationship between peer and expert evaluations, participants' revisions to their designs and recombination of ideas from others, and participants' perceptions of the competition. Throughout, we use a mixed-methods analysis, incorporating quantitative analysis of attributes of the designs and qualitative analysis of the designs and interview data.

A. Designs

In the UX competition, most of the designs consisted of mockup screenshots of the final, envisioned user interface (Fig. 4). These screenshots were often accompanied by brief explanatory text, describing possible user interactions with user interface elements and the resulting behavior of the interface. All of the designs described mechanisms for enabling the user to lay out roads, adjust traffic density, create light sequences, and simulate traffic flows. Most designs depicted screenshots in temporal order, for example, depicting how to build a road map, how to add street lanes, and then how to adjust light behavior. Designs varied greatly in length of textual descriptions, ranging from extensive explanations to a few words. Top designs were often more visually polished and contained clear, concise, and detailed explanations of interface elements (e.g., Fig. 4a). Weak designs were often less visually refined and less detailed and precise in their consideration of user interactions (e.g., Fig. 4b). Surprisingly, none of the designs explicitly discussed user needs that had led to their design, simply focusing on the final product - the user interactions. Only one designer (UC1) explicitly listed design decisions and assumptions about the domain such as "cars drive the speed limit".

In contrast to the UX designs, the AD designs were text-centric, using diagrams as supporting materials (Fig. 5). Many had a high-level structure that included requirements, assumptions, discussion of the domain, and implementation details, although designs varied widely in the ways that each was presented and discussed. Unlike the UX designs, many designs walked through the derivation of the design, discussing in detail assumptions about the requirements and the domain model before presenting a design itself. Top designs often focused more on presenting a detailed and precise characterization at the level of a domain model (e.g., Fig. 5a, 7c, 7d), while bottom designs often focused more on a characterization emphasizing the implementation through class diagrams and detailed listings of algorithms (e.g., Fig. 5b). As in the UX designs, top

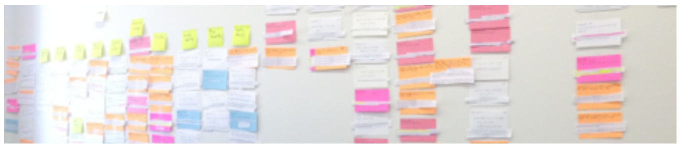


Fig 3. A section of the affinity diagram constructed during data analysis.

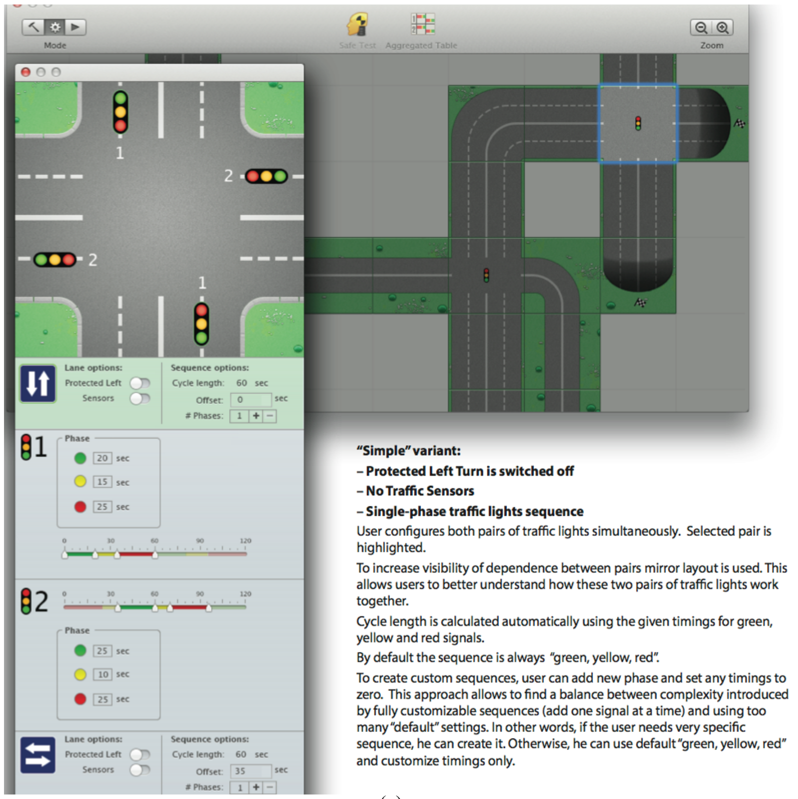
AD designs were often also more visually polished, detailed, and precise. But, unlike the UX designs, there a wide range of sections presented, including requirements, scope, quality attributes, technology choices, design rationale, constraints, use cases, and algorithms. Designs used a variety of diagrams, including both diagrams of the domain and of the design itself. Some designs used diagrams with formal notations, such as class diagrams (e.g., Fig. 5b), sequence diagrams, and use case diagrams; but stronger designs often focused instead on diagrams of the domain model with ad-hoc notations (e.g., Fig. 5a). Most, but not all, followed the instructions of the prompt in not considering the design of the user interactions.

Designs varied greatly in length (Table 1). UX designs varied from 1 page to 18 pages, with a first round mean length of 6.5 (±4.4) pages. AD designs were, on average, longer, ranging from 3 to 19 pages with a mean first round length of 10.3 (±4.9) pages. In UX designs, the first round page length was significantly correlated with scores (r = .59, p = .04), while the second round correlation was not significant (r = .46, p = .13). In AD designs, scores in both rounds were strongly correlated with page count (R1: r = .80, p = .005; R2: r = .76, p = .01).

Surprisingly, there was no significant relationship between the amount of time spent creating initial designs and first round scores (UX: r = .11, p = .74; AD: r = -.18, p = .62). This suggests that there is a strong expertise effect that enables strong designers to produce top designs in similar amounts of time. In a few cases, top designs were produced in considerably less time. For example, the winning initial design in the AD competition (AC1) was produced in 5.5 hours of time, while one of the lowest scoring designs (AC5) was produced in 20 hours. Expertise effects are partially visible in the relationship between industry experience and scores; in the AD, but not UX designs, there was a significant relationship (UX round 1: r = .32, p = .31; UX round 2: r = .35, p = .26; AD round 1: r = .66, p = .04; AD round 2: r = .61, p = .03). The differences between AD and UX may partially result from the small sample size and the greater variability in industry experience amongst AD designers. The effects of the time participants invested in their designs was more visible in their revisions. The time spent on design revisions and the improvement in score was moderately and significantly correlated in the UX competition (r = .61, p = .04) but not in the AD competition (r = .55, p = .10).

B. Peer Evaluations

Overall, peer evaluations by the UX competition were moderately correlated with expert ranks (r = .37, p < .0007) and strongly correlated with expert ranks in the AD competition (r = .65, p < .00001). 66% of UX peer ranks were within one rank of the expert rank, while 85% of AD peer ranks were within one rank of the expert rankings (Fig. 6). Self evaluations were less accurate, especially in the UX groups. Self evaluations in the UX groups were not significantly correlated with expert ranks (r = .18, p = .39) but moderately correlated with expert rankings in the AD competition (r = .55, p = .012). 54% of UX



Lane options: Protected Left Sensors
Sequence options: Cycle length: 60 sec Offset: 0 sec # Phases: 1

1 Phase 20 sec 15 sec 25 sec

2 Phase 25 sec 10 sec 25 sec

Lane options: Protected Left Sensors
Sequence options: Cycle length: 60 sec Offset: 35 sec # Phases: 1

"Simple" variant:

– Protected Left Turn is switched off

– No Traffic Sensors

– Single-phase traffic lights sequence

User configures both pairs of traffic lights simultaneously. Selected pair is highlighted.

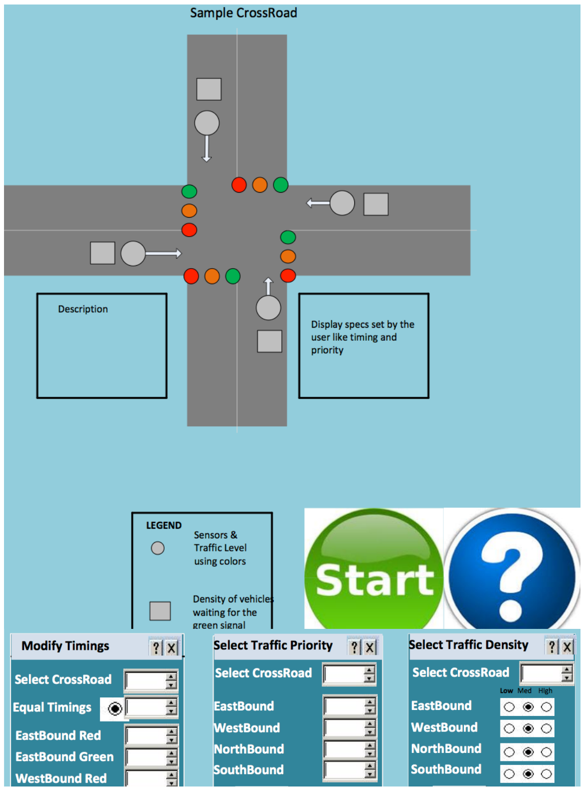
To increase visibility of dependence between pairs mirror layout is used. This allows users to better understand how these two pairs of traffic lights work together.

Cycle length is calculated automatically using the given timings for green, yellow and red signals.

By default the sequence is always "green, yellow, red".

To create custom sequences, user can add new phase and set any timings to zero. This approach allows to find a balance between complexity introduced by fully customizable sequences (add one signal at a time) and using too many "default" settings. In other words, if the user needs very specific sequence, he can create it. Otherwise, he can use default "green, yellow, red" and customize timings only.

(a)


Sample CrossRoad
Description
Display specs set by the user like timing and priority
LEGEND
Sensors & Traffic Level using colors
Density of vehicles waiting for the green signal
Start
?

Modify Timings
Select CrossRoad
Equal Timings ●
EastBound Red
EastBound Green
WestBound Red

Select Traffic Priority
Select CrossRoad
EastBound
WestBound
NorthBound
SouthBound

Select Traffic Density
Select CrossRoad

	Low	Med	High
EastBound	○	●	○
WestBound	○	●	○
NorthBound	○	●	○
SouthBound	○	●	○

(b)

Fig. 4. Sections of the revised UX designs from the winning design (a) - UE2 - and lowest ranked design (b) - UC7.

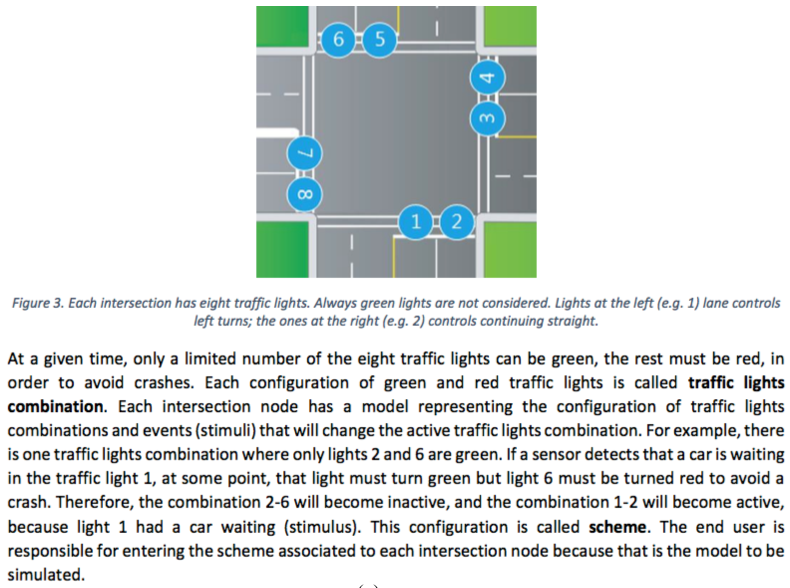


Figure 3. Each intersection has eight traffic lights. Always green lights are not considered. Lights at the left (e.g. 1) lane controls left turns; the ones at the right (e.g. 2) controls continuing straight.

At a given time, only a limited number of the eight traffic lights can be green, the rest must be red, in order to avoid crashes. Each configuration of green and red traffic lights is called **traffic lights combination**. Each intersection node has a model representing the configuration of traffic lights combinations and events (stimuli) that will change the active traffic lights combination. For example, there is one traffic lights combination where only lights 2 and 6 are green. If a sensor detects that a car is waiting in the traffic light 1, at some point, that light must turn green but light 6 must be turned red to avoid a crash. Therefore, the combination 2-6 will become inactive, and the combination 1-2 will become active, because light 1 had a car waiting (stimulus). This configuration is called **scheme**. The end user is responsible for entering the scheme associated to each intersection node because that is the model to be simulated.

(a)

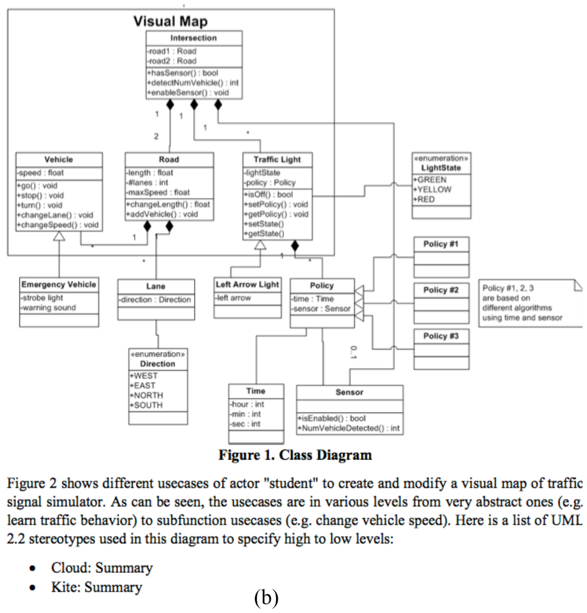


Figure 1. Class Diagram

Figure 2 shows different usecases of actor "student" to create and modify a visual map of traffic signal simulator. As can be seen, the usecases are in various levels from very abstract ones (e.g. learn traffic behavior) to subfunction usecases (e.g. change vehicle speed). Here is a list of UML 2.2 stereotypes used in this diagram to specify high to low levels:

- Cloud: Summary
- Kite: Summary

(b)

Fig. 5. Sections of the revised AD designs from the winning design (a) - AC1 - and lowest ranked design (b) - AE5.

self evaluations were within one rank of the expert rank, while 45% of AD self evaluations were within one rank of the expert rank (Fig. 6). In making self evaluations, participants were much more likely to rate themselves higher than the experts, although some did rate themselves lower.

To evaluate designs, participants reported using several strategies. Most examined each design individually, assessing its fitness according to one of several criteria. The most popular criteria designers reported using were those suggested in the design prompt: completeness, elegance, and clarity. Beyond these, designers also reported considering designs' usability and visual design (UX) and level of detail and flexibility (AD).

Several participants reported evaluating designs by making explicit comparisons between designs. Two AD designers (AC2, AE2) reporting using an insertion sort, where they compared each design with the previous, inserting it in the appropriate place. Several UX designers (UE2, UE5, UC7) reported examining designs in pairs, while UC6 compared all designs to their own. AC3 used a grouping strategy, first separating designs into "good" and "bad" groups before reading each in

Table 1. Summary of Experience, Design Scores, and Design Effort

User Experience (UX) Competition								Architecture Design (AD) Competition							
Designer	In-dustry exp. (yrs)	Round 1			Round 2			Designer	In-dustry exp. (yrs)	Round 1			Round 2		
		Score	Pages	Time (hrs)	Change in Score	Page incr-ease	Time (hrs)			Score	Pages	Time (hrs)	Change in Score	Page incr-ease	Time (hrs)
Control															
UC1	5	15.25	10	5.5	-1.25	1	2	AC1	8	18.25	14	5.5	0.25	1	2
UC6	2	6	5	2	7.5	1	5	AC2	4	15.75	11	5	0.5	1	2.5
UC5	4	8	4	10	5	1	15	AC3	5	8.75	12	6	2	1	5
UC2	2	12.5	8	6	0.25	-1	3.5	AC4	4	7.75	5	9	2.75	2	24
UC3	4	10.5	7	10	2	1	6	AC5	3	8.0	3	20	2	5	37.5
UC4	2	8.75	1	4.5	1.5	4	2.25								
UC7	3	4.5	2	6	1.5	1	2.5								
Mean	3.1	9.4	5.3	6.3	2.4	1.1	5.2	*Mean*	4.8	11.7	9	9.1	1.5	2	14.2
Experimental															
UE2	5	14.5	6	13.5	4	2	13.5	AE1	8	17.5	19	11	0	1	4
UE1	4	16	5	5	0	0	3	AE2	3.5	15.75	11	20	1.5	0	6
UE3	2	14.25	18	5	0.75	2	5	AE3	4	13.75	14	8	1.75	3	8
UE4	2	14	8	7.5	0.25	4	3	AE4	4	9.5	9	12	4.75	3	15
UE5	3	10	4	15	-0.25	1	4	AE5	4	8.0	5	10	0	0	2
Mean	3.2	13.8	8.2	9.2	1.0	1.8	5.7	*Mean*	4.7	12.9	11.6	12.2	1.6	1.4	7
Overall mean	3.2	11.2	6.5	7.5	1.8	1.4	5.4	*Overall mean*	4.8	12.3	10.3	10.7	1.6	1.7	10.6

Designers are identified by a code indicating their group and rank of their initial design (e.g., UC1 indicates the top ranked control UX participant). Designers are listed in order of the score for their revised design.

more detail. Several (UC3, UC7, AC2, AE1) reported that they found ranking similar designs hard. One designer (UE5) also found ranking difficult for designs that were too dissimilar, "*It's hard to put designs side by side because there are so many different variables in terms of design style and clarity. There was no baseline.*" Participants reported finding evaluating designs neither particularly easy nor difficult, with a mean difficulty of 3.8 (UX) and 4.1 (AD) on a 7 point scale (1 easiest).

Participants reported spending an average of 1.1 (±.7)(UX) and 2.2 (±1.8)(AD) hours evaluating designs in the first round and 1.2 (±.6)(UX) and 1.8 (±1.4)(AD) hours in the second round. AD designs may have been more time consuming to rank due to their greater length and extensive use of text.

C. Design Revisions

1) Effects of design revisions

Overall, participants' revised designs were significantly better than their first round designs (UX: p = .03; AD: p = .009). On average, UX designs improved by 1.8 points and AD designs improved by 1.6 points (Table 1). 75% of UX designs and 80% of AD designs improved. Only two UX designs (17%) decreased in score, while no AD designs decreased in score. There was no effect of first round design evaluations on improvements; experimental participants who evaluated designs in the first round did not improve more than the control participants that did not (UX, p = .28; AD, p = .92).

2) Borrowing ideas

Overall, all participants, both control and experimental, reported to have carefully reviewed each initial design and borrowed at least one idea from another design. Table 2 lists the borrowed ideas we identified in each design (as corroborated

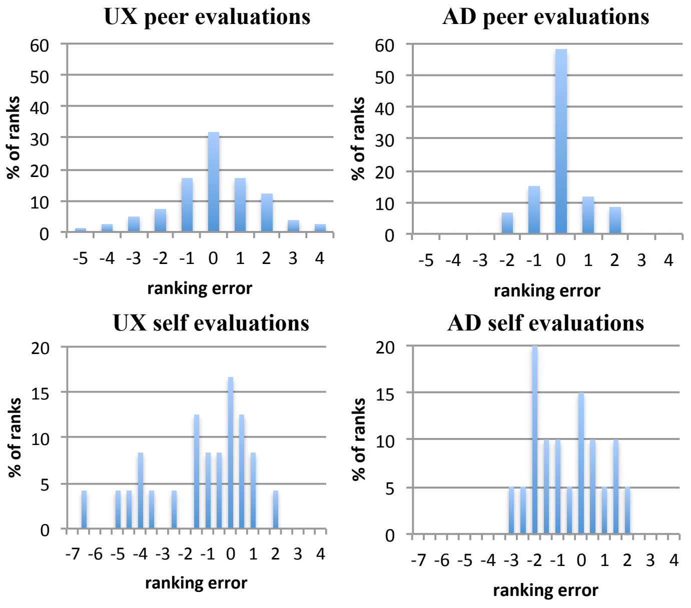
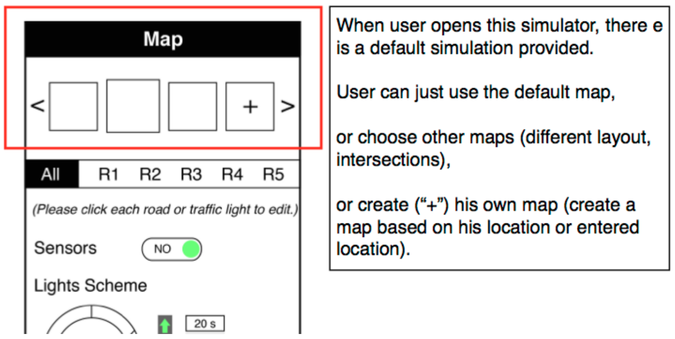


Fig. 6. Accuracy of peer and self evaluations. Positive ranking errors occur when rankings were lower than expert rankings, and negative errors occur when rankings were higher. Participants were allowed to give non-integer rankings for themselves (e.g., "1 or 2") but not peers.



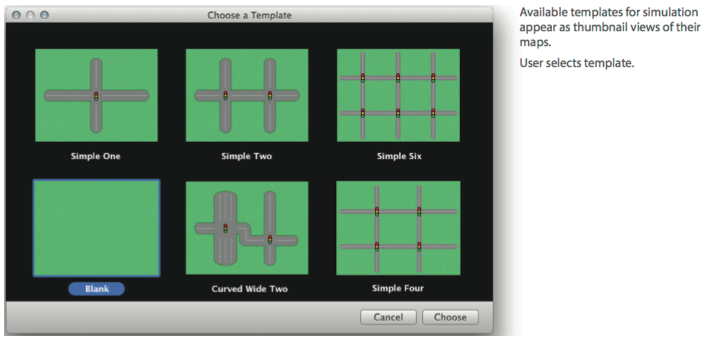
When user opens this simulator, there e is a default simulation provided.

User can just use the default map,

or choose other maps (different layout, intersections),

or create ("+") his own map (create a map based on his location or entered location).

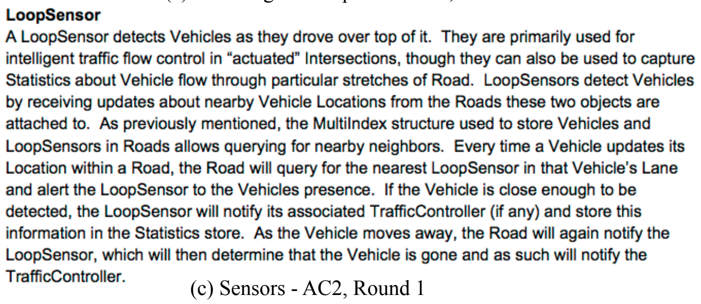
(a) Pre-designed templates - UE4, Round 1



Available templates for simulation appear as thumbnail views of their maps.

User selects template.

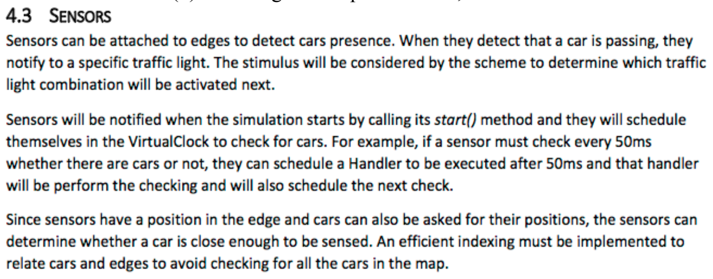
(b) Pre-designed templates - UE2, Round 2



LoopSensor

A LoopSensor detects Vehicles as they drove over top of it. They are primarily used for intelligent traffic flow control in "actuated" Intersections, though they can also be used to capture Statistics about Vehicle flow through particular stretches of Road. LoopSensors detect Vehicles by receiving updates about nearby Vehicle Locations from the Roads these two objects are attached to. As previously mentioned, the MultiIndex structure used to store Vehicles and LoopSensors in Roads allows querying for nearby neighbors. Every time a Vehicle updates its Location within a Road, the Road will query for the nearest LoopSensor in that Vehicle's Lane and alert the LoopSensor to the Vehicles presence. If the Vehicle is close enough to be detected, the LoopSensor will notify its associated TrafficController (if any) and store this information in the Statistics store. As the Vehicle moves away, the Road will again notify the LoopSensor, which will then determine that the Vehicle is gone and as such will notify the TrafficController.

(c) Sensors - AC2, Round 1



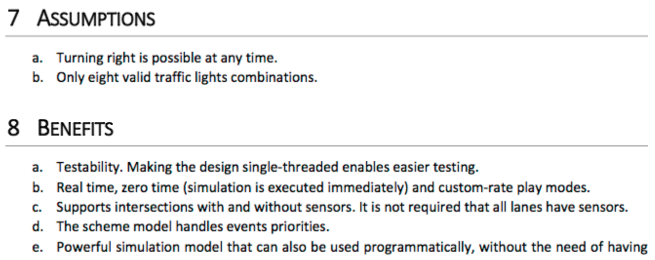
4.3 SENSORS

Sensors can be attached to edges to detect cars presence. When they detect that a car is passing, they notify to a specific traffic light. The stimulus will be considered by the scheme to determine which traffic light combination will be activated next.

Sensors will be notified when the simulation starts by calling its *start()* method and they will schedule themselves in the VirtualClock to check for cars. For example, if a sensor must check every 50ms whether there are cars or not, they can schedule a Handler to be executed after 50ms and that handler will be perform the checking and will also schedule the next check.

Since sensors have a position in the edge and cars can also be asked for their positions, the sensors can determine whether a car is close enough to be sensed. An efficient indexing must be implemented to relate cars and edges to avoid checking for all the cars in the map.

(d) Sensors - AC1, Round 2



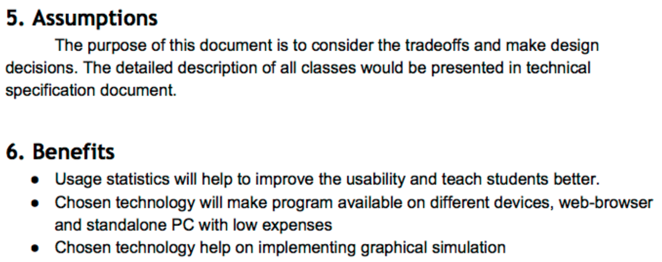
7 ASSUMPTIONS

a. Turning right is possible at any time.
b. Only eight valid traffic lights combinations.

8 BENEFITS

a. Testability. Making the design single-threaded enables easier testing.
b. Real time, zero time (simulation is executed immediately) and custom-rate play modes.
c. Supports intersections with and without sensors. It is not required that all lanes have sensors.
d. The scheme model handles events priorities.
e. Powerful simulation model that can also be used programmatically, without the need of having

(e) Assumptions and benefits - AC1, Round 1



5. Assumptions

The purpose of this document is to consider the tradeoffs and make design decisions. The detailed description of all classes would be presented in technical specification document.

6. Benefits

- Usage statistics will help to improve the usability and teach students better.
- Chosen technology will make program available on different devices, web-browser and standalone PC with low expenses
- Chosen technology help on implementing graphical simulation

(f) Assumptions and benefits - AC3, Round 2

Fig. 7. Examples of borrowing by UX (a; b) and AD designers (c; d)(e; f).

through the interviews), listing designs in order of their first round score. Both participants with top initial designs and participants with low-ranked initial designs found ideas to borrow. Participants varied greatly in the ideas they chose to borrow; few ideas were consistently borrowed by multiple participants. Participants found ideas in many different designs, borrowing ideas from both top ranked designs and bottom ranked designs. Of the 16 initial UX designs (including participants that subsequently dropped), 13 provided at least one idea that another participant subsequently adopted. Of the 12 initial AD designs, 10 provided at least one idea. In some cases, designers reported adapting a single idea from multiple source designs.

Both UX and AD designs often borrowed ideas in the form of features. Participants borrowed ideas from both higher-ranked designs and from lower-ranked designs. For example, UE3 identified and borrowed from the higher-ranked UE1 the idea of a 24 hour traffic graph enabling users to see, at a glance, how traffic flow varied over a day (Fig. 1). UE2 borrowed from the lower-ranked UE4 the idea of enabling users to start a new map with a pre-defined template containing a specific layout of roads (Fig. 7a, b). As a final example, AC1 borrowed from the lower-ranked AC2 the idea of using sensors to detect the presence of cars (Fig. 7c, d). In borrowing ideas, participants often took only the essence of the idea, adapting and reinterpreting its meaning in the context of their design to make it their own. For example, while AC1 borrowed the idea of sensors from AC2 (Fig. 7c, d), the discussion of how sensors are implemented is specific to their own design. Thus, while participants borrowed ideas from higher and lower-ranked designs, the polish and precision of designs often reflected their own design style rather than their source design (e.g., in Fig. 7b, the higher-ranked UE2 has a higher degree of polish in their version of pre-designed templates).

In contrast to UX participants, AD participants also borrowed presentation elements of the design, including sections and types of diagrams. For example, AC3 borrowed sections describing assumptions and benefits (Fig. 7e, f); AC4 borrowed sections describing quality attributes and a static architectural view (Table 2). This may reflect the greater diversity of presentation styles available for AD participants to borrow.

Overall, however, participants' revised designs were very similar to their initial designs. While most participants borrowed several ideas and, for AD designs, sections and diagrams, participants, with one exception, did not reenvision their design wholesale or make large, global changes (UC6 revised their initial low-fidelity mockups with high-fidelity mockups).

While participants reported extensively reviewing other designs and identifying ideas, not all ideas could be incorporated into their own designs. When participants felt that their design was sufficiently general or that the source design was similar, designers felt that there was a good fit between designs:

TABLE 2. IDEAS BORROWED BY PARTICIPANTS

User Experience (UX) Competition

Designer	**Ideas borrowed**	**Source of idea**							
Control participants	***UC***	***1***	***2***	***3***	***4***	***5***	***6***	***(A)***	***7***
UC1	1. Pause, finish buttons, 2. Accelerate button, 3. Car density measured as CPM, 4. CPM		1, 2		3	4	2		
UC2	1. Added more roads, 2. Manually changing traffic			1			2		
UC3	1. Vector, 2. Compass guide	1				2			
UC4	1. Change words to icons, 2. Help and instructions, 3. Colorblindness assistance	1		1, 2, 3		2	1, 2		
UC5	1. Notifications panel, 2. Error handling, 3. Traffic specified per road, 4. Toolbar to add signals and roads, 5. Play button behavior, 6. Road congestion highlighting	1, 2, 3, 4		5			2, 6		
UC6	1. Randomize traffic density, 2. Roads layout		1	2					
UC7	1. Start and help buttons			1		1	1		
Experimental participants	***UE***	***1***	***2***	***3***	***4***	***(B)***	***(C)***	***(D)***	***5***
UE1	1. Popup window light sequence, 2. Light timing visualization			1					2
UE2	1. Intersection creation, 2. Color labels, 3. Saving light sequences, 4. Pre-designed templates, 5. Simulation controls, 6. Check boxes to toggle buttons	1, 2, 3		2	2, 4, 5		5		6
UE3	1. Saving light sequences, 2. Conflict warnings, 3. 24 hour traffic graph, 4. Use existing light sequence, 5. Naming streets and intersections	1, 2, 3, 4				5			
UE4	1. Saving light sequences, 2. Popup window for light scheme, 3. Sequence and time	1		2, 3					
UE5	1. Path tracing							1	

Architecture Design (AD) Competition

Designer	**Ideas borrowed**	**Source of idea**					
Control participants	***AC***	***1***	***2***	***(E)***	***3***	***4***	***5***
AC1	1. Sensors, 2. Concept of events, 3. Framework to design optimal path		1, 2, 3				
AC2	1. Light scheme, 2. Bi-directional traffic	1					2
AC3	1. Assumptions, 2. Benefits, 3. Limitations	1, 2, 3					
AC4	1. Quality attributes, 2. Static view of architecture				1, 2		
AC5	1. Extended domain model, 2. Compass directions	1		2			
Experimental participants	***AE***	***1***	***2***	***3***	***4***	***(F)***	***5***
AE1	1. Off-the-shelf distribution generator, 2. UI mockup, 3. Functional requirements, 4. Map controller logic		1	2, 3		2, 4	
AE2	1. System diagram, 2. Road graph and subcomponents diagram	1, 2				1, 2	1, 2
AE3	1. Model elements, 2. Simulation sequence	1, 2					
AE4	1. Graphical dependency graph, 2. Multiple turns at streets, 3. Classes cheatsheet		1, 2			3	
AE5	1. Compass directions, 2. System reqmts and components, 3. Traffic simulator, 4. Traffic flow controller	1	2, 3, 4	2		1	

Designers are listed in order of their initial design scores (highest first). First round designs from participants that subsequently dropped out of the competition are listed with a letter in parenthesis (e.g., (A)).

I looked through all of them individually again and then went through them one by one. I made a list of things that I liked from each of them. But I kept my own design in mind. If the things I liked would also fit in my design, I chose to incorporate it. (UC1)

But designers also felt that there were ideas that they could not incorporate due to a lack of fit or lack of time. For example, AE2 reported that *"They had a state machine for the whole thing, and that was just a really interesting approach… [But] I would pretty much have had to redesign my whole design, so I decided to stick with what I had so far"*.

3) *Self-critique*

Beyond borrowing feature or elements of the presentation from other designs, both UX and AD participants reported that reading other designs led them to reflect on and critique their own designs. This lead some participants to better understand the expectations of the contest and consider alternative approaches. Several (AC5, UE2, AC1, AE5, AE1, AE2, AC2) reported that seeing other designs led them to improve the presentation of their designs to be more clear and explicit. For example, AC2 reported that "*there were like a few cases where I felt like I didn't explain particular features enough*." Several participants also reported that seeing alternative approaches inspired them to consider additional aspects of the problem and to identify missing pieces of their design. For example, AE2 *"felt like the other designs talked a lot more about roads than I did. So I tried to improve [that] in the next round.*" And AE5 reported that *"when I saw the designs and different thoughts behind them, I was able to see the crowd better and what kind of approaches there are"*. One (UC5) reported that other designs led them to feel that theirs was too technical and detailed.

Many participants (AC3, AC4, AE1, AE3, AE4, UC1, UC2, UC4, UC7) were encouraged by seeing good designs by others, as it gave them the opportunity to improve. For example, AE3 reported "*When I see that something can be made better, I actually see how I can implement it in my design, and that motivates me*". Similarly, others (UE2, UE3) were disappointed by designs that they felt reflected a lack of effort. However, the lowest ranked AD participant (AE5) reported that seeing good designs was discouraging as it made her think that her own design was really bad. Three designers (UE4, AC2, AE1) also reported that seeing worse designs was encouraging. For example, AC2 reported, "*I feel that that made me feel like I put more effort than other people did into the design process, and that I stood a pretty good chance*".

D. *Perceptions of the Design Competitions*

Overall, participants found the contest to be effective. First, participants thought the setup of the competition encouraged good design, as seeing the efforts of others motivated them to do good work and to put forth more effort. Second, participants felt that sharing ideas helped design both in providing ideas and in providing a deeper understanding of the design space that encouraged self-critique. For example, both UC5 and UC7 reported that seeing how other designs approached the problem made them realize how different UIs can be. All participants reported that they did not try to game the competition in any way; none reported withholding ideas from the first round to prevent them from being borrowed. "*To be honest, I haven't even really thought of it.*" (AC2)

A number of participants reported difficulties in the design competitions. Some (UC5, UE2, UE3, UE4) reported unfamiliarity with the domain, "*it was very hard, especially for [a] non-US citizen, to create the system because I had to search for the rules*." Many (UC3, UC4, UC5, UC7, UE5, AE2, AE5, AC1, AC4, AC5) reported wishing that they had more time. A few (UC4, UE3, AE5) reported being blocked by limitations of their design tools.

Participants reported three main ideas for improving the effectiveness of design competitions:

1) Standardized requirements, expectations, and tools

Participants suggested clearer requirements and expectations would help them to borrow more. For instance,

What do you mean by design? The term design. Because for me it's more about architecture stuff, maybe for others it's more about classes, about detailed descriptions like specifications for programmers.(AC3)

Similarly, UE2 felt "*it was too general. Everybody designed differently. Maybe give more guidance.*" AC2 suggested common tooling would reduce the effort required to borrow. "*If I could have accessed their diagrams in order to ... pick some components of theirs that I thought were better explained ... and dropped them into mine.*"

2) More iteration through smaller steps and early feedback

Participants reported that making large changes in their design revisions was hard, as they had already created a complete design and that that completeness served as a barrier to incorporating large or incompatible design ideas.

It's more helpful to have more stages in smaller chunks. Maybe one stage would just be [to] explore wireframes or sketch ideas. Because I think a lot of the good ideas, they take time and several iterations to perfect... In the second phase, I felt it was hard to change. (UE5)

I think it would've helped me a lot if even before writing out the whole design document I could've seen the thought process. So kind of like a pre-design step. It would've been really interesting to see how other people framed the core of the system. So like state machine versus concurrent queues, which is what I used. Little things like that, half page things. I thought that once my design was written, it was a lot harder to incorporate because you kind of have to find a way for it all to fit in. (AE2)

3) Provide two-way communication

While the contest enabled participants to see other designs and create indirect feedback through comparisons to others, several UE participants (UE1, UE4, UC3, UC5) felt that direct critiques of their designs by others would be very helpful. AE3 (AD) further suggested that participants be "*allowed to discuss ideas with each other, and after the first round the best designs should cooperate on a single design.*"

V. Limitations and Threats to Validity

As all studies, our study has several limitations. While all our participants reported having industry experience, none were senior or highly experienced designers with decades of experience. Highly experienced designers might not similarly benefit from recombination. However, in practice, software competitions are usually open and often specifically target those with less expertise, who benefit most from the experience visibility to recruiters that participation affords. Moreover, our results suggest that both stronger and weaker designers may similarly benefit from a recombination process.

While the task was carefully designed to be representative of real world design tasks and used in previous studies, it was, by necessity, limited in scope, and is not inclusive of all activities and aspects that may be involved in larger design tasks.

Our results are based on a mix of quantitative and qualitative analysis. While data analysis involving coding may introduce bias, we used several mechanisms to reduce and mitigate potential sources of bias. In scoring designs, four panelists independently assessed each design, and members of the panel were blind to the identity and round of each design. In identifying borrowed ideas, two authors independently coded instances, which were then cross-checked with the designers themselves during the interviews. In analyzing interview data, two authors independently coded each interview to identify insights and all of the coded insights were organized into themes. Finally, while analysis of participants' changes and interviews together with the expert scores provides evidence that designers borrowed and that designs improved, our study provides no direct causal linkage between the two. As we were unsure how much, if any, borrowing might occur, we chose to use an experimental manipulation to explore the conditions in which borrowing occurs. Future experiments comparing recombination to design improvements are necessary.

VI. Discussion

Our study revealed the potential of recombination within software design competitions to enable designers to share ideas and improve their designs. Other designs furnished designers with a rich source of ideas. The motive to borrow was so strong that an experimental manipulation intended to increase borrowing had no discernible effect. Borrowing was surprisingly egalitarian - while designs varied substantially in quality, borrowed ideas came from nearly all designs. Even strong designers found ideas with which to improve from weak designs. Designers often took only the essence of an idea, adapting, reinterpreting, and extending it to fit their own design.

Designers found that seeing the designs of others provided a new perspective with which to examine their own. Designers were inspired by viewing the designs of others, returning to their own designs eager to address features they felt to be weak, improve their presentation, and fill in missing pieces. Rather than simply borrow, designers often used what they learned to reflect and think more deeply about their own ideas. Valuing the perspectives of others, designers wished to see more explicit feedback to provide more opportunities to improve.

Both architectural and user experience design can benefit from outside ideas. While obtaining outside ideas using techniques such as design critiques has long been a central emphasis in user interaction design, our results suggest that new ideas can also benefit architectural design. All designers in both competitions borrowed from the crowd. Throughout, UX and AD designers' activities were often more alike than different, as both critiqued their own designs and improved their designs. One difference between the types of design was in presentation. While UX designers all chose to present designs very similarly, AD designers benefited more from simply observing the presentation styles of others, leading them to add new sections and types of diagrams to explain additional dimensions of their design. This suggests that the nature and scope of architectural design may be less well-defined than interaction design.

Incorporating new ideas into a complete design enables design refinement rather than radical redesign. Participants used the designs of others to add features and enhance their designs, not to rethink their central approach. While participants wished to borrow more, several barriers held them back. Participants spoke extensively of the "fit" between their designs and others, explaining that they saw ideas that they liked but whose poor fit made them difficult or prohibitively time consuming to adapt. Moreover, by encouraging participants to produce a polished initial design, participants may have already felt committed to its precepts and been less willing to imagine reenvisioning or restarting from scratch. This suggest that, to encourage borrowing of bigger ideas, it is crucially important for designers to first submit earlier stage ideas. Just as traditional design processes emphasize ideation, low-fidelity sketches, and iterative improvement [2], software design competitions may be able to encourage larger design improvements by supporting iteration beginning with early stage ideas.

Designers engaged in competition still value collaboration. Despite competing against each other for substantial $1000 prizes, designers wished to see more opportunities for collaboration and direct, explicit feedback on their designs. This seems somewhat counterintuitive: why would designers provide helpful feedback to others that might serve only to reduce their subsequent chances of winning? Yet, despite the competitive nature inherent, designers already felt they were receiving value from their peers, finding useful ideas and inspiration from the strong designs others produced. Indeed, more participants felt motivated by seeing the strong designs of others than by weak designs. This suggests there may be promise in competition models that further combine competition and collaboration.

Peer-evaluations can be used to approximate evaluations of designs by experts. Participants demonstrated modest success in evaluating peer designs. Individual peer evaluations were moderately to strongly correlated with expert evaluations and often differed only slightly from experts. This suggests that, in commercial crowdsourcing context where work is commissioned by a client, it may be possible to let the crowd themselves perform some of the evaluation work, reducing the significant burden evaluations can impose on clients [33]. One important aspect to further investigate is in averaging individual evaluations into aggregate evaluations, as this may enable even higher quality evaluations.

The effects of expertise may be as important in software design as in programming. In an early study of human aspects of software development, Sackman et al. found a ratio between the best and worst developers of over 10 to 1 for tasks such as initial coding and debugging [30]. Many studies have since found similar expertise effects across a range of programming tasks (e.g., [23, 12, 28, 17]). Our results provide evidence that substantial expertise effects extend to architectural and user experience design. Some designers were able to produce top designs in substantially less time than that in which others produced low ranked designs. While recombination enabled most to improve, it did not enable weak designers to produce strong designs. Even in adapting the ideas of stronger designers, the level of polish and precision in their version often reflected more their own level of design expertise than that of the original designer. For software competitions, this is an important limitation, as they may contain a range of expertise levels and substantial populations of students and less experienced developers looking to gain expertise and knowledge. Finding ways to help weak designers improve more through interactions with stronger designers is a an important area of future research.

Our results suggest that adopting a multi-round structure in software design competitions that enables revision and recombination can increase the quality of designs produced and better utilize the diversity inherent to competitions. Yet, our results also suggest there is much more to explore in making full use of the crowd in software design through alternative competition structures. A competition with many more rounds, standardized tooling, and lower fidelity design ideas might enable true collective design, in which the core ideas are exchanged and adapted and alternatives are developed and explored over time within the crowd. Designers might initially produce ideas for sections of the design, which are then adopted and extended. Shorter rounds might also reduce contribution barriers, enabling designers to come and go through the competition, contributing in shorter periods when they are able.

Our results may also have implications for more traditional design processes within companies. In illustrating the value of outside ideas, they suggest that adopting even simple brainstorming processes might help to increase design quality. Rather than have a single designer bring a complete design to a meeting for review by his colleagues, it might be possible to instead have several designers independently sketch several early ideas, increasing the diversity of ideas available and reducing the commitment to a single, fully specified design. Designers might even continue a collaborative design process, posting designs to a shared wiki to enable idea exchange and having multiple designers iterate the designs over time. Whereas pair programming enables developers to collaboratively work together to solve hard programming tasks, our results suggest a new team organization in which pairs or small groups explore alternatives in parallel and exchange ideas.

ACKNOWLEDGMENTS

We thank the participants of our study and Ben Koehne for assisting in judging. This work was supported in part by the NSF under NSF grants IIS-1111446 and CCF-1414197.

References

[1] A. Begel, J. Bosch, M. A. Storey, “Social networking meets software development: perspectives from github, msdn, stack exchange, and topcoder.” IEEE Software, vol. 30 (1), 2013, pp. 52-66.

[2] W. Buxton, Sketching User Experiences: Getting the Design Right and the Right Design. Morgan Kaufmann, 2007.

[3] D. T. Campbell, “Blind variation and selective retention in creative thought as in other knowledge processes.” Psychological Review, vol. 67, 1960, pp. 380-400.

[4] S. Chawla, J. D. Hartline, B. Sivan, “Optimal crowdsourcing contests.” Symposium on Discrete Algorithms, pp. 856-868.

[5] K. Crowston, K. Wei, J. Howison, and A. Wiggins, “Free/libre open source software development: what we know and what we do not know.” ACM Computing Surveys, vol. 44 (2), 2012, 35 pages.

[6] W. Dietl, S. Dietzel, M. D. Ernst, et. al., “Verification games: making verification fun.” FTfJP 2012, pp. 42-49.

[7] S. Dow, J. Fortuna, D. Schwartz, B. Altringer, D. Schwartz, and S. Klemmer, “Prototyping dynamics: sharing multiple designs improves exploration, group rapport, and results.” CHI, 2011, pp. 2807-2816.

[8] S. Dow, A. Glassco, J. Kass, M. Schwarz, D. L. Schwartz, and S. R. Klemmer, “Parallel prototyping leads to better design results, more divergence, and increased self-efficacy.” ACM Transactions on Computer-Human Interaction, vol. 17 (4), 2010, 24 pages.

[9] F. Geyer, J. Budzinski, H. Reiterer, “IdeaVis: a hybrid workspace and interactive visualization for paper-based collaborative sketching sessions.” In Nordic Conference on Human-Computer Interaction, 2012, pp. 331-340.

[10] M. Goldman, “Software development with real-time collaborative coding in a Web IDE.” Symposium on User Interface Systems and Technologies (UIST), 2011, pp. 155-164.

[11] M. Goldman, G. Little, and R. C. Miller, “Collabode: collaborative coding in the browser.” Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2011, pp. 155-164.

[12] L. Gugerty and G. M. Olson. Comprehension differences in debugging by skilled and novice programmers. In Empirical Studies of Programmers, 13-27, 1986.

[13] J. Hailpern, E. Hinterbichler, C. Leppert, D. Cook, and B. P. Bailey, “TEAM STORM: demonstrating an interaction model for working with multiple ideas during creative group work.” Conference on Creativity and Cognition, 2007, pp. 193-202.

[14] B. Hartmann, D. MacDougall, J. Brandt, and S. R. Klemmer, “What would other programmers do: suggesting solutions to error messages.” CHI, 2010, pp. 1019-1028.

[15] B. M. Hill and A. Monroy-Hernandez, “The cost of collaboration for code and art: evidence from a remixing community.” CSCW, 2013, pp. 1035-1046.

[16] J. Howe, Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business. Crown Business, 2008.

[17] T. D. LaToza, D. Garlan, J. D. Herbsleb, B. A. Myers, “Program comprehension as fact finding.” European Software Engineering Conference and Foundations of Software Engineering (ESEC/FSE), 2007, pp. 361-370.

[18] T. D. LaToza, W. B. Towne, C. M. Adriano, and A. van der Hoek, “Microtask programming: building software with a crowd.” Symposium on User Interface Systems and Technology (UIST), 2014, pp. 43-54.

[19] T. D. LaToza, W. B. Towne, A. van der Hoek, and J. D. Herbsleb, “Crowd development.” Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2013.

[20] W. Li, S. A. Seshia, and S. Jha, “Towards crowdsourced human-assisted verification.” Design Automation Conference (DAC), 2012, pp. 1254-1255.

[21] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, “Design lessons from the fastest Q&A site in the west.” CHI, 2011, pp. 2857-2866.

[22] N. Mangano, T. D. LaToza, M. Petre, A. van der Hoek, “Supporting informal design with interactive whiteboards.” CHI, 2014, pp. 331-340.

[23] S. McConnell, “What does 10x mean? Measuring variations in programmer productivity” In A. Oram and G. Wilson, eds., Making Software, O’Reilly, 2011.

[24] S. L. Minneman and S. A. Bly, “Managing a trois: a study of a multi-user drawing tool in distributed design work.” CHI, pp. 217-224.

[25] D. Mujumdar, M. Kallenbach, B. Liu, and B. Hartmann, “Crowdsourcing suggestions to programming problems for dynamic web development languages”. CHI 2011 Extended Abstracts, pp. 1525-1530.

[26] F. Pastore, L. Mariani, and G. Fraser, “CrowdOracles: can the crowd solve the oracle problem?” ICST 2013, pp. 342-351.

[27] M. Petre and A. van der Hoek, Software Designers in Action: A Human-Centric Look at Design Work. CRC Press, 2013.

[28] M. P. Robillard, W. Coelho, G. C. Murphy. How Effective Developers Investigate Source Code: An Exploratory Study. In IEEE Transactions on Software Engineering (TSE), vol. 30, no. 12, 889-903, Dec. 2004.

[29] B. Rohrbach, “Creative nach regeln: Methode 635, eine neue technik zum losen von problemen. Absatzwirtschaft, vol. 12 (19), 1969, pp. 73–75.

[30] H. Sackman, W. J. Erikson, E. E. Grant, “Exploratory experimental studies comparing online and offline programming performance.” Communications of the ACM (CACM), vol. 11 (1), 1968, pp. 3-11.

[31] J. J. Shah, N. Vargas-Hernandez, J. D. Summers, and S. Kulkarni, “Collaborative sketching (c-sketch)–an idea generation technique for engineering design.” Creative Behavior, vol. 35 (3), 2001, pp. 168–198.

[32] D. K. Simonton, “Scientific creativity as constrained stochastic behavior: The integration of product, person, and process perspectives.” Psychological Bulletin, vol. 129, 2003, pp. 475-494.

[33] K. Stol and B. Fitzgerald. Two's company, three's a crowd: a case study of crowdsourcing software development. International Conference on Software Engineering (ICSE), 2014, pp. 187-198.

[34] J. Surowiecki, The Wisdom of Crowds. Random House, Inc., 2005.

[35] R. N. Taylor and A. van der Hoek, “Software design and architecture: the once and future focus of software engineering.” Future of Software Engineering, 2007, pp. 226-243.

[36] P. Thagard, Conceptual revolutions. Princeton University Press, USA, 1992.

[37] C. Watson, F. W. B. Li, J. L. Godwin, “Bluefix: using crowd-sourced feedback to support programming students in error di¯agnosis and repair.” In E. Popescu, Q. Li, R. Klamma, H. Leung, and M. Specht, eds., Advances in Web-Based Learning, ICWL 2012, Springer Berlin Heidelberg, pp. 228-239.

[38] A. Xu and B. P. Bailey, “A crowdsourcing model for receiving design critique.” CHI Extended Abstracts, 2011, pp. 1183-1188.

[39] L. Yu and J. V. Nickerson, “Cooks or cobblers? Crowd creativity through combination.” CHI, 2011, pp. 1393-1402.

[40] Z. Zhao, S. K. Badam, S. Chandrasegaran, D. G. Park, N. L. E. Elmqvist, L. Kisselburgh, and K. Ramani, “skWiki: a multimedia sketching system for collaborative creativity” CHI, 2014, pp. 1235-1244.