Simulation-Based Modeling and Evaluation of Incentive Schemes in Crowdsourcing Environments

Ognjen Scekic, Christoph Dorn, and Schahram Dustdar

Distributed Systems Group, Vienna University of Technology
{oscekic,dorn,dustdar}@dsg.tuwien.ac.at
http://www.dsg.tuwien.ac.at

Abstract. Conventional incentive mechanisms were designed for business environments involving static business processes and a limited number of actors. They are not easily applicable to crowdsourcing and other social computing platforms, characterized by dynamic collaboration patterns and high numbers of actors, because the effects of incentives in these environments are often unforeseen and more costly than in a well-controlled environment of a traditional company.
In this paper we investigate how to design and calibrate incentive schemes for crowdsourcing processes by simulating joint effects of a combination of different participation and incentive mechanisms applied to a working crowd. More specifically, we present a simulation model of incentive schemes and evaluate it on a relevant real-world scenario. We show how the model is used to simulate different compositions of incentive mechanisms and model parameters, and how these choices influence the costs on the system provider side and the number of malicious workers.

Keywords: rewards, incentives, crowdsourcing, social computing, collective adaptive systems

1 Introduction

Research on incentives in crowdsourcing systems has been increasingly attracting interest recently (e.g., [19,11,15,9]). Today's commercial crowdsourcing systems mostly deal with simple tasks and lack worker interactions and dependencies. Such collaborative patterns in many ways resemble traditional piece-work, enabling use of conventional pay-for-performance incentive mechanisms [16]. These existing incentive mechanisms are based upon statistical models (e.g., *agency theory*) that take into consideration workers engaging in contractual, long-term relationships with a traditional company and seeking to maximize their utility metrics (see Section 2). However, as social computing systems grow more complex (e.g., Collective Adaptive Systems[1]) the web scale and unstable nature of crowd worker interactions with the system makes the use of traditional incentives unpredictable and inappropriate.

Conventional incentive models completely disregard social characteristics of the crowd, such as coordinated group actions, social/regional/ethnic peculiaritites, voluntary work [6], importance of reputation/flaunting [18], or web-scale malicious behavior

[1] http://focas.eu/

[21]. An additional complication is that these phenomena change often and characterize different subsets of the crowd differently in different moments. This makes development of appropriate mathematical incentive models difficult. Specifically, the root cause lies in insufficient understanding of the implications arising from a particular combination of worker participation patterns and applied incentive schemes.

The system designer needs to consider additional factors, such as: emerging, unexpected and malicious worker behavior, incentive applicability, range of stability, reward fairness, expected costs, reward values and timing. Failing to do so leads to exploding costs and work overload, as the system cannot scale with the extent of user participation. Unbalanced rewards keep new members from joining or cause established members to feel unappreciated and leave. Ill-conceived incentives allow users to game the system, prove ineffective against vandalism, or assign too many privileges to particular members tempting them to abuse their power [13].

This calls for a systematic approach in designing and evaluating incentive schemes before deployment on real-world social computing systems. In [16] we surveyed existing incentive practices in traditional companies and different crowdsourcing platforms and illustrated the shortcomings of applying conventional incentive mechanisms in crowdsourcing environments. We then proposed how to combine proven atomic incentive mechanisms into scalable and portable incentive schemes suitable for social computing systems. Based on these conclusions, in [15,17] we presented a model and a system capable of deploying and executing such mechanisms. Continuing on this line of reseach in this paper we now investigate how to select, customize and evaluate appropriate atomic incentive mechanisms and how to compose them for a given crowdsourcing scenario. Specifically, we propose modeling and simulating various participation options available to workers, activities and costs on the system provider side, how user actions are transformed into rewards, and how these rewards in turn influence user behavior. Further justification for this approach is presented in Section 2.

The contributions of this paper are:

1. Abstract simulation model of incentive mechanisms for crowdsourcing (Section 3).
2. Concrete incentive model for a real-world crowdsourcing scenario based on 1.
3. Complete modeling and simulation methodology, detailing the implementation and evaluation processes to design a concrete incentive model such as 2. (Section 5)

The validity and capabilities of the model and the methodology are evaluated through a relevant simulation scenario.

The remainder of this paper is structured as follows. Section 2 provides a discussion on related work and our previous work. Section 3 provides an overview of our approach, the abstract incentive model and the simulation rationale. Section 4 presents two relevant scenarios that we use as the environment to demonstrate and evaluate the methodology. We discuss the methodology and concrete design decisions in Section 5. A scenario case-study in Section 6 demonstrates the simulation's usefulness to provide insights into behavior and effectiveness of selected incentive mechanisms and other design decisions. Section 7 gives an outlook on future work and concludes this paper.

2 Background and Related Work

Previous research on incentives can be roughly categorized in two groups. One group seeks to find optimal incentives in formally defined environments through precise mathematical models (e.g., principal-agent theory [8,2], game theory [4,7]). Both the agent (worker) and the authority (employer) are seen as entities deciding on their actions with the exclusive goal of maximizing gain or achieving a predefined goal. Although successfully used in microeconomic models, these incentive models do not fully capture the diversity and unpredictability of human behavior that becomes accentuated in a crowdsourcing environment. These disadvantages (see [11]) prompted the advent of another direction in research of incentives in crowdsourcing.

The other group examines the effects of incentives by running experiments on existing crowdsourcing platforms and rewarding real human subjects with actual monetary rewards (e.g., [9,11]). The major disadvantages of this approach are its high cost and duration. Furthermore, although seemingly yielding realistic findings, there is evidence that the low amounts of monetary rewards used in these experiments make the findings applicable only for a very limited range of simple activities, such as image tagging and text translation. These and other shortcomings that this type of research suffers from are listed in [1].

In contrast to these two approaches, our intention is not to devise novel nor optimal incentive mechanisms for crowdsourcing, but rather to offer system designers a methodology for quickly selecting, composing and customizing existing, real-world atomic incentive mechanisms [16,19], and roughly predicting the effects of their composition in dynamic crowdsourcing environments. The model and simulation parameters can be changed dynamically, allowing quick testing of different incentive scheme setups and behavioral responses at low cost. The schemes can then be deployed on systems such as [15,17] and provided as a service to the third parties.

Our simulation approach allows modeling of incentives and responses of workers of arbitrary complexity. Specifically, we employ principles of agent-based social simulation [10,5], an effective and inexpensive scientific method for investigating behavioral responses of large sets of human subjects. As we are primarily interested in investigating how reputation affects (malicious) behavior, we characterize each agent by reputation metric, as laboratory experiments confirmed that reputation promotes desirable behavior in a variety of different experimental settings [20,12,14,18].

However, unlike the usual approach where agents interact directly (and thus benefit from cooperative behavior or suffer from defective behavior), we introduce a provider that facilitates interactions and determines the benefits or costs of those interactions. Reputation allows the provider to assess an agents reliability. Consequently, pure reputation sharing alone is insufficient. We require additional incentive mechanisms to obtain cooperative behavior beyond the users intrinsic level. Further differences to the conventional agent-based simulation include the explicit, detailed modeling of the underlying collaboration patterns, thereby building upon our previous work [3].

3 Incentive Mechanisms for Crowdsourcing Processes

Any incentive mechanism in general involves two interested parties - an *authority* and a *worker* (actor, agent). The authority is interested in stimulating, promoting or discouraging certain behavioral responses in workers.The incentive exhibits its psychological effect by promising the worker a reward or a punishment based on the actions the worker will perform. The wish to get the reward or escape the punishment drives the worker's decisions on future actions. The reward (punishment) can be material or psychological (e.g., a change of status in a community – ranking, promotion). The type, timings and amounts of reward need to be carefully considered to achieve the wanted effect of influencing a specific behavior in a planned direction. In addition, introduction of incentives introduces additional costs for the authority who hopes to compensate for them through the newly arisen worker behavior (e.g., increased productivity).

However, as soon as an incentive mechanism is introduced, it produces dysfunctional behavioral responses in the worker population. The workers adapt to the new rules and change their working patterns, often in unpredictable or even malicious ways, trying to misuse the new incentive to profit more than the rest of the population [13]. The authority compensates for this by introducing other incentive mechanisms targeting the dysfunctional behavior, further increasing the authority-side costs, and causing new types of dysfunctional behavior. However, once the proper combination of incentive mechanisms is put in place and calibrated, the system enters a stable state. The problem with the crowdsourcing processes is that the system may not stay long in a stable state due to an unforeseen change in worker participation or collaboration pattern. Therefore, the incentive setup needs to be reconfigured and re-calibrated as quickly as possible, in order to avoid incurring high costs to the authority. This feedback control-loop involving the authority and the worker represents the actual incentive mechanism that we model and simulate in this paper.

Modeling an incentive mechanism, therefore, always involves modeling both the authority and the worker side, as well as the possible interactions between them. In Figure 1 we show an abstract representation of the model of incentive mechanism that we implement in the following sections.

Workers differentiate from each other by having different sets of personal characteristics (e.g., accuracy, speed, experience). The characteristics are determined by a private set of variables stored in the *internal state* S. Usually, the variables are normally distributed across the worker population, although particular variables can be intentionally given predefined values to provoke a certain type of behavior. The internal state also contains records of worker's past actions. The internal state is private to the worker, and is used as one of the inputs for the *decision-making function* f_a that chooses the next action to perform.

Apart from the internal state, each worker is characterized by the publicly exposed set of *performance metrics* M that are defined and constantly updated by the authority for each worker. The performance metrics reflect the authority's perception of the worker's past interactions with the system (e.g., trust, rank, expertise, responsiveness). Knowing this allows the worker to decide better on his future actions. For example, knowing that a poor reputation will disqualify him from getting a reward in future may drive the worker to work better or to quit the system altogether. It also allows him to

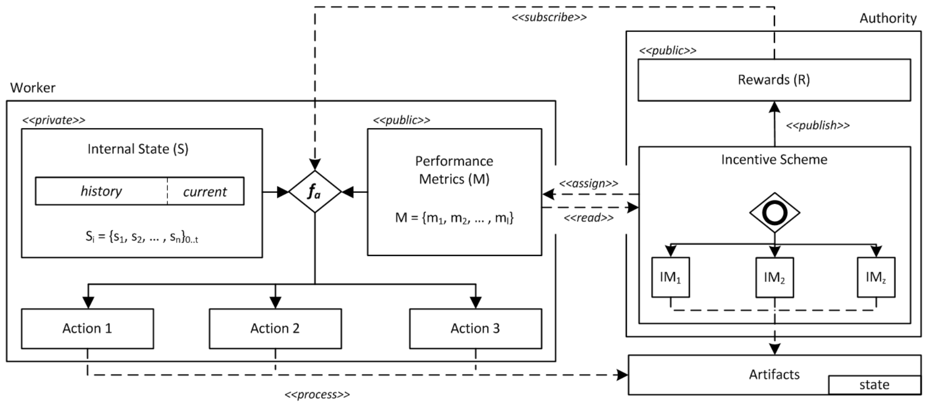


Fig. 1: Incentive mechanisms need to capture the interaction between worker and authority.

compare with other workers. Therefore, the set of performance metrics is another input for the decision-making function f_a.

The third input for the decision-making function f_a is the set of promised *rewards (punishments) R*. Rewards are expressed as publicly advertised amounts/increments in certain parameters that serve as the recognized means of payment/prestige within the system (e.g., money, points, stakes/shares, badges). They are specified per action per artifact and per performance metrics, thus making them also dependent per user. For example, a reward may promise an increase of at least 100 points to any/first user who performs the action of rating an artifact. The amount of points can then be further increased or decreased depending on the user's reputation.

Worker interacts with the authority solely by performing actions over *artifacts* (K) offered to the worker population by the authority. Worker's behavior can thus be described as a sequence of actions in time, interleaved with periods of idling (idling being a special-case of action). The set of possible actions is the same for every worker. However, the effects of the execution of an action may be different, depending on the worker's personal characteristics from the internal state S. For example, a worker with innate precision and bigger experience can improve an artifact better than the worker not possessing those qualities.

As previously stated, worker's next action is selected through the use of a decision-making function $f_a = f(S, M, R)$ potentially considering all of the following factors: a) the statistically or intentionally determined personality of the worker; b) historical record of past actions; c) authority's view of one's own performance; d) performance of other workers; and e) promised rewards, with respect to the current state of one's performance metrics. The decision-making function is arbitrarily defined by the system designer. For example, we can use a utility-maximization function, as described in the papers cited in Section 2.

The authority's motivation for offering artifacts for processing to the crowd is to exploit the crowd's numerosity to either achieve higher quality of the artifacts (e.g., in terms of accuracy, relevance, creativity), or lower the cost (e.g., in terms of time or money). This motivation guides the authority's choice of incentive mechanisms. Authority has at its disposal a number of *incentive mechanisms* IM_i. Each one of them should be designed to target/modify only a small number of very specific parameters (see later). Thus, it is the proper addition or composition of incentive mechanisms that allows the overall effect of an incentive scheme, as well as fine-tuning and runtime modifications.

An incentive mechanism IM takes as inputs: 1) the current state of an artifact K_i; 2) the current performance metrics of a worker M_j; and optionally 3) the output from another incentive mechanism returning the same type of reward – R'_{a_k}. The output of an incentive mechanism is the amount/increment of the reward R_{a_k} to offer to the worker M_j for the action a_k over artifact K_i.

$$IM : (K_i, M_j, R'_{a_k}) \rightarrow R_{a_k} \tag{1}$$

The true power of incentive mechanisms lies in the possibility of their combination. The reward (f_R) can be calculated through a number of additions (+) and/or functional compositions ($\circ$) of different incentive mechanisms. For example, a worker may be given an increment in points for each time he worked on an artifact in the past. Each of those increments can then be modified, depending on how many other workers worked on that same artifact. In addition, the total increment in points can be further modified according to the worker's current reputation. The finally calculated increment value represents the promised reward. The set of finally calculated rewards per worker $R_w = \{f_{R_1}, ..., f_{R_z}\}$ is then advertised to the workers, influencing their future behavior, and closing the feedback loop.

The major difficulty in designing a successful incentive scheme lies in properly choosing the set of *incentive parameters* (performance metrics, incentive mechanisms, and their compositions). Often, the possible effects when using one set of parameters are unclear at design time, and an experimental or a simulation evaluation is needed to determine them. A proven set of incentive parameters is usually called an *incentive scheme*.

4 Motivating Scenarios

Here we present two relevant scenarios for which our simulation model and methodology can be used to design and evaluate appropriate incentive schemes.

Citizen-driven traffic reporting. Local governments have a responsibility to provide timely information on road travel conditions. This involves spending considerable resources on managing information sources as well as maintaining communication channels with the public. Encouraging citizens to share information on road damages, accidents, rockfalls, or flooding reduces these costs while providing better geographical coverage and more up to date information[2]. Such crowdsourcing process, however,

[2] For a real world example visit the Aberdeen City Council's SmartJourney initiative at `http://smartjourney.co.uk/`

poses data quality related challenges in terms of assessing data correctness, completeness, relevance, and duplication.

Crowdsourced software testing. Traditional software testing is a lengthy and expensive process involving teams of dedicated engineers. Software companies[3] may decide to partially crowdsource this process to cut time and costs and increase the number and accuracy of detected defects. This involves letting the remote testers detect bugs in different software modules and usage environments and submitting bug reports. Testers with different reputations provide reports of varying quality and change the assigned bug severity. As single bugs can be reported multiple times in separate reports, testers can also declare two reports as duplicates.

The two scenarios exhibit great similarities. The expected savings in time and money can in both cases be outweighed by an incorrect setup and application of incentive mechanisms. Furthermore, the system could suffer from high numbers of purposely incorrect or inaccurate bug report submissions, driving the processing costs high. For the purpose of this paper, we join and generalize the two scenarios into a single, abstract one that we will use in our simulation setup:

The *Authority* seeks to lower the time and cost of processing a large number of *Reports* on various *Situations* occurring in the interest domain of the Authority. The *Workers* are independent agents, occasionally and irregularly engaging with the system managed by the Authority to perform one of the following *Actions*: *Submit* a new Report on a Situation, *Improve* an existing Report, *Rate* the accuracy and importance of an existing Report, inform the Authority of his belief that two existing Reports should be considered *Duplicates*. The Worker actions are driven by the combination of the following factors: a) possibility to earn *Points* (translating to increased chances of exchanging them for money); b) possibility to earn *Reputation* (translating to a higher status in the community); and c) the intrinsic property of people to contribute and help or to behave maliciously. In order to influence and (de-)motivate workers, the Authority employs a number of *Incentive Mechanisms*, collectively referred to as *Incentive Scheme*.

This scenario also needs to address the following challenges:

- *Crowdsourced report assessment.* The effort required for manual validation of worker-provided reports may easily outweigh the gained effort and cost reduction from crowdsourced reporting in the first place. Hence, workers need to be properly stimulated to supplement and enrich existing reports as well as vote on their importance, thereby lifting the verification burden off the authority. The system also needs to strike a balance not to collect too much information.
- *Worker reputation (trust).* A worker's reputation serves as one potential indicator for data reliability, assuming that reputable workers are likely to provide mostly accurate information. Subsequently, reports from workers with unknown or low reputation need to undergo more thorough peer assessment. The system must support continuous adjustment of workers' reputation.
- *Adjustable and composable incentive scheme.* An effective incentive scheme needs to consider all past citizen actions, the current state of a report and the predicted costs of processing a report manually in order to decide whether and how to stimulate workers to provide additional information. It also needs to correctly identify

[3] For example, www.utest.com

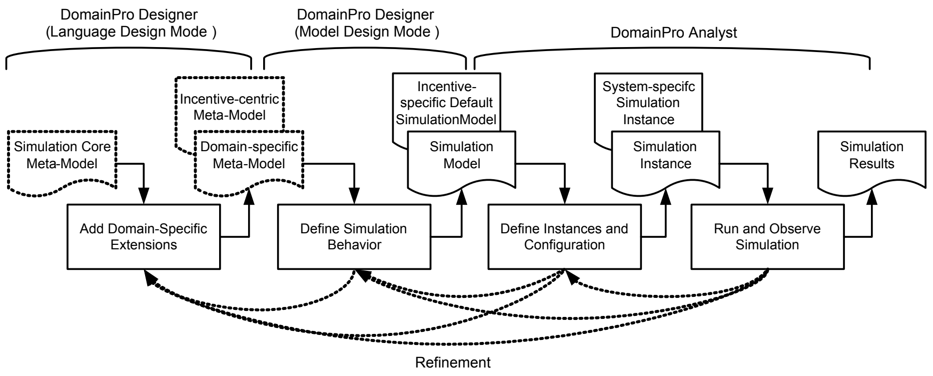
and punish undesirable and selfish behavior (e.g., false information, deliberate duplication of reports, intentional up/downgrading of reports).

The resulting complexity arising from the possible combination and configuration of worker behavior, incentive schemes, and processing costs requires a detailed analysis to identify a stable and predictable system configuration and its boundaries.

5 Modeling and Simulation Methodology

Our methodology for simulating worker participation and incentive mechanisms in crowdsourcing processes is depicted in Figure 2. It consists of four basic steps, usually performed in multiple iterations: *i)* defining a domain-specific meta-model by extending a core meta-model; *ii)* capturing worker's behavioral/participation patterns and reward calculation into an executable model; *iii)* defining scenarios, assumptions, and configurations for individual simulation runs; and *iv)* evaluating and interpreting simulation results. These steps are described in more detail below.

We use the DomainPro[4] modeling and simulation tool suite in each of the outlined methodology steps to design and instantiate executable models of incentive mechanisms and run simulations of those models. The tool allows creating custom simulation languages through metamodeling and supports agent-based and discrete event simulation semantics (see [3]). However, our overall approach is generic and can be easily applied using a different modeling and simulation environment.



DomainPro Designer (Language Design Mode)
DomainPro Designer (Model Design Mode)
DomainPro Analyst
Simulation Core Meta-Model
Incentive-centric Meta-Model
Domain-specific Meta-Model
Incentive-specific Default SimulationModel
Simulation Model
System-specifc Simulation Instance
Simulation Instance
Simulation Results
Add Domain-Specific Extensions
Define Simulation Behavior
Define Instances and Configuration
Run and Observe Simulation
Refinement

Fig. 2: The methodology of simulation design and development.

The simulation core meta-model is implemented in the DomainPro Modeling Language. Optional extensions result in a domain-specific meta-model that defines which component types, connector types, configuration parameters, and links a simulation model may exhibit. In our case, we extend the core meta-model to obtain what we refer to as *incentive-centric meta-model* (Section 5.1). The obtained incentive-centric

[4] `www.quandarypeak.com`. Open-source version forthcoming.

meta-model serves as the basis for defining the simulation behavior, i.e., the *executable simulation model* (Section 5.2). Obtaining the executable simulation model requires defining workers' behavioral parameters, authority's business logic (including incentive mechanisms and cost metrics), the environment and the control flow conditions between them. Finally, prior to each execution, the executable simulation model requires a quick runtime configuration in terms of the number of worker instances and monitored performance metrics (Section 6.1). During the execution, we do near real-time monitoring of metrics, and if necessary, perform simulation stepping and premature termination of the simulation run to execute model refinements.

The tool we use enables refinement at any modeling phase. A designer will typically start with simple meta- and simulation models to explore the basic system behavior. She will subsequently refine the meta-model to add, for example, configuration parameters and extend the functionality at the modeling level. This enables testing simple incentive mechanisms first, and then extending and composing them once their idiosyncrasies are well understood.

5.1 Incentive-Centric Meta-Model

The derived meta-model (Fig.3) reflects the conceptual view of incentive mechanisms as presented in Section 3. A *ParticipationPattern* consists of *Actors* (*User*s or *Providers*) and the *InteractionObjects*. Actors exhibit *Behavior* that encapsulates different *UserActivities*. InteractionObjects contain *ObjActivities* that define allowed and reward-yielding activities on an InteractionObject. *InternalSequences*, *ExternalSequence*, and *ObjectSequence* determine the control flow among activities by specifying trigger conditions. The *EnvGenerator* drives the simulation by controlling the generation of interaction objects (artifacts) for the *Workers* to act upon, and for the *Authority* to check and further process. The *AtomicData* within a *SimulationElementType* defines which data may be passed between UserActivities and/or ObjActivities when an InternalSequence, ExternalSequence, or ObjectSequence fires. While arbitrary data types can be passed along, only AtomicData of type int, double, long, or boolean may be used as observable metrics during simulation execution. The exact applicable metrics are defined later on, on the simulation instance level.

As previously outlined, the iterative nature of the modeling process usually requires extending the core meta-model with domain-specific elements, as the need for them is identified. The domain-specific extensions we introduce are highlighted in bold/blue in Figure 3.

5.2 Simulation Model of the Real-World Scenario

In this section we derive an executable simulation model for evaluating the impact of various design decisions taken during the modeling of the case-study scenario from Section 4. Specifically, the goals of the simulation are:

i) prototyping and evaluating various incentive schemes;
ii) determining the impact of malicious user behavior;

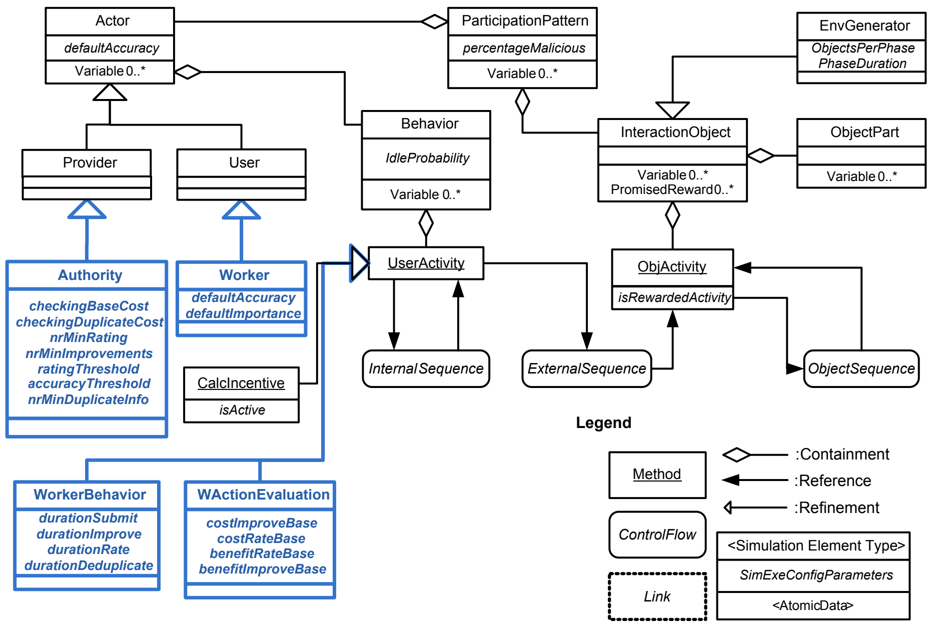


Fig. 3: Simulation meta-model including domain specific extensions in bold/blue.

iii) observing trends in processing costs, reward payments, report accuracy, and user activities.

Figure 4 provides a partial screenshot of the case-study simulation model.

The simulation model comprises over 40 simulation parameters, determining various factors, such as: distribution of various personality characteristics in the worker population, injected worker roles (e.g., malicious, lazy), base costs for the authority, selection and composition of incentive mechanisms. Due to space limitations, describing them all in detail/formally is not possible. Therefore, the rest of this section is written in a narrative style. [5]

Location and importance characterize a *Situation*. Situations can be generated with user-determined time, location and importance distributions, allowing us to concentrate more problematic (important) situations around a predefined location in selected time intervals, if needed. For the purpose of this paper, we generate situations with uniform probability across all the three dimensions. The *SituationGenerator* contains the activities for creating new situations and calculating phase-specific simulation metrics on cost, reputation, points, actions, and importance across reports, situations and workers.

The *Worker*'s *SetNextStep* activity represents the implementation of the worker's decision-making function f_a, introduced in Section 3. As previously explained, the worker here considers the next action to perform based on: 1) internal state (e.g., *location*), including innate, population-distributed personality characteristics (e.g, *laziness*,

[5] The interested reader can consult the annotated source code and visualize the model and the metamodel here: `http://tinyurl.com/sdd-coopis13-src`

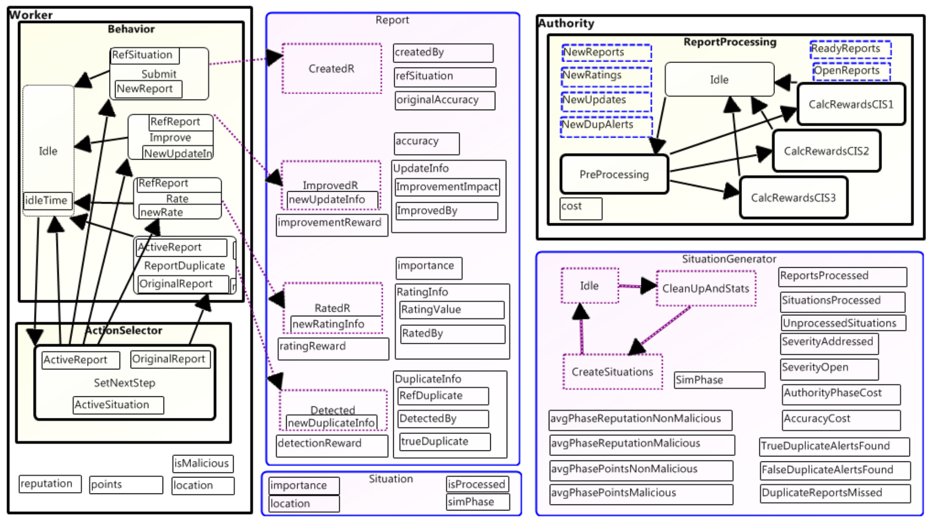


Fig. 4: Partial screenshot of the implemented case-study simulation model in Domain-Pro Designer.

isMalicious); 2) current performance metrics (e.g., *reputation*, *points*); 3) advertized rewards (*detectionReward*, *ratingReward*, *improvementReward*).

Worker's location determines his/her proximity to a situation, and, thus, the likelihood to detect or act upon that situation (the smaller the distance, the higher the probability). However, two workers at the same distance from a situation will not equally likely act upon it. This depends on their personality, past behavior, and the number of points they currently have.

Points and *reputation* are the principal two metrics by which the authority assesses Workers in our scenario. In principle, points are used by the Authority as the main factor to stimulate activity of a Worker. The more points, the less likely will a worker idle. On the other hand, a higher reputation implies that the Worker will more likely produce artifacts of higher quality. Each new worker joining the system starts with the same default point and reputation values. Precisely how the two metrics are interpreted and changed thereafter depends on the incentive mechanisms used (see below).

The four *Behavior* activities produce the respective artifacts – *Reports*, *UpdateInfos*, *RatingInfos* and *DuplicateInfos*. Worker's internal state determines the deviations of accuracy, importance, improvement effect, and rating value of the newly created artifacts. The subsequently triggered *Report*-located activities (*CreatedR*, *ImprovedR*, *RatedR* and *Detected*) determine the worker action's effect on the two metrics that represent the artifact's state and data quality metrics at the same time – report *accuracy* and *importance*. We use Bayes estimation to tackle the cold-start assessment of report accuracy and importance, taking into account average values of existing reports and the reputation of the worker itself.

The produced artifacts are queued at the *Authority* side for batch processing. In *PreProcessing* activity we determine whether a Report is ready for being processed.

This depends on the report's quality metrics, which in turn depend on the amount and value of worker-provided inputs.

Processing reports causes costs for the Authority. The primary cost factors are low quality reports and undetected duplicate reports. Secondary costs arise when workers focus their actions on unimportant reports while ignoring more important ones. Therefore, the Authority incentivizes the workers to submit required amounts of quality artifacts. As noted in Section 4, gathering as much inexpensive data from the crowd as possible was the original reason for the introduction of a crowdsourced process in the first place.

Our proof-of-concept simulation model for the given scenario defines three basic incentive mechanisms:

- **IM_1:** Users are assigned fixed amounts of points per action, independent of the artifact. Submitting yields most points.
- **IM_2:** The amount of points is increased before assignment, depending on the current quality metrics of the report. E.g., the fewer ratings or improvements the higher the increment in points.
- **IM_3:** Users are assigned a reputation. The reputation rises with accurately submitted reports, useful report improvements, correctly rated importance and correctly flagged duplicates.

As we shall see in Section 6.1, we can compose these three mechanisms in different ways to produce different incentive schemes which we can run and compare.

Workers that behave differently than usual can be easily injected into the system for simulation purposes by defining new *roles* and generating new workers or converting existing workers to take up those roles. For demonstration purposes we define only a single additional role - that of a malicious worker.

Malicious worker behavior is designed to cause maximum cost for the Authority. To this end, we assume malicious workers to have a good perception of the actual situation characteristics. Hence, upon submission they will set initial report importance low and provide very inaccurate information subsequently. For important existing reports they submit negative improvements (i.e., conflicting or irrelevant information) and rate them low and while doing the opposite for unimportant reports.

6 Evaluation

For evaluating our approach, we keep using the case-study scenario from the previous sections and perform a set of experiments on it. All provided experimental data is averaged from multiple, identically configured simulation runs. Details on the experiment setup are followed by experiment result[6] presentation and gained insights.

6.1 Experiment Setup

Timing Aspects. We control the pace of the simulation by determining the amount of situations created per phase. Taking a reading of all relevant (i.e., experiment-specific)

[6] Data available here: `http://tinyurl.com/scekic-dorn-dustdar-coopis13`

metrics at the end of each phase provides an insight on how these metrics change over time. All our simulations last for 250 time units (t), consisting of 10 phases of $25t$ each. Batch creation of situations is representative for real world environments such as bugs that typically emerge upon a major software release or spikes in traffic impediments coinciding with sudden weather changes. Report submission takes $5t$, while improving, rating, and duplication flagging require only $1t$. The exact values are irrelevant as we only need to express the fact that reporting requires considerably more time than the other actions. Processing of worker-provided data on the provider side occurs every 1t. Note here, that for the purpose of the case study, we are only interested in the generic processing costs rather than the time it takes to process that data. Each report is assumed to cause 10 cost units for zero-quality, and almost no cost when quality (through worker-provided improvements) approaches 1.

Scenario-specific thresholds. As we aim for high-quality data and significant crowd-base confirmation, the following thresholds need to be met before a report is considered for processing: at least three updates and high accuracy (> 0.75); or five ratings and medium importance (> 0.5); or four duplication alerts; or being reported by a worker of high reputation (> 0.8) and having high importance (> 0.7). Workers obtain various amounts of points for (correct) actions, the amount depending on the value of the action to the provider and the incentive scheme used.

Worker Behavior Configuration. A worker's base behavior is defined as 70% probability idling for $1t$, 20% submitting or duplication reporting, and 10% rating or improving. Obtained points and reputation increase the likelihood to engage in an action rather than idle. The base behavior represents rather active workers. We deliberately simulate only the top-k most involved workers in a community as these have most impact on benefits as well as on costs. Unless noted otherwise, $k = 100$ for all experiments.

Composite Incentive Schemes. The experiments utilize one or more of the following three *Composite Incentive Schemes – CIS*, introduced in Section 5.2:

- $CIS1 = IM_1$
- $CIS2 = IM_2 \circ IM_1 = IM_2(IM_1)$
- $CIS3 = CIS2 + IM_3 = IM_2 \circ IM_1 + IM_3$

CIS1 promises and pays a stable amount of points for all actions. CIS2 dynamically adjusts assigned points based on the currently available worker-provided data, but at least as high rewards as CIS1. CIS3 additionally introduces reputation calculation.

6.2 Experiments

Experiment 1: Comparing Composite Incentive Schemes. Here we compare the impact of CIS1, CIS2, and CIS3 on costs, assigned rewards, report accuracy, and timely processing. Figure 5 displays incurred costs across the simulation duration. All three schemes prove suitable as they allow 100 workers to provide sufficient data to have 20 situations processed at equally high accuracy. They differ, however, significantly in cost development (Fig.5 inset), primarily caused by undetected duplicate reports (on average 0.2, 0.25, and 0.4 duplicates per report per phase for CIS1, CIS2, and CIS3, respectively). CIS1 yields stable and overall lowest costs as the points paid induce just the right level of activity to avoid workers getting too active and thus causing duplicates. This is exactly the shortcoming of CIS2 which overpays workers that subsequently become overly active. CIS3 pays even more, and additionally encourages worker activity through reputation. The cost fluctuations are caused by the unpredictable number of duplicates (however remaining within bounds). Although more costly and less stable, CIS3 is able to identify and subsequently mitigate malicious workers (see Experiment 3 below).

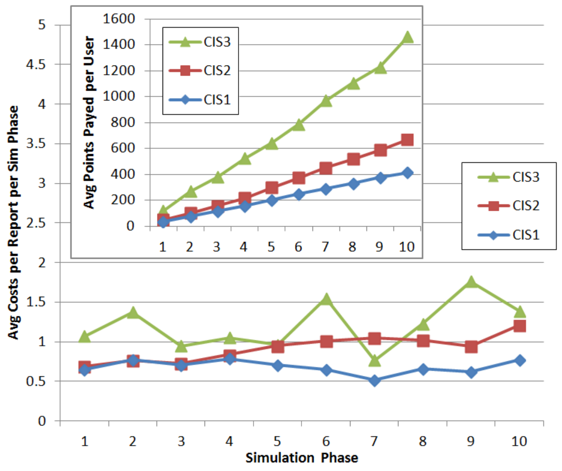


Fig. 5: Incurred report processing costs for CIS1, CIS2, and CIS3. Inset: average paid points per worker.

Experiment 2: the Effect of Worker/Situation mismatch. Here we analyze the effects of having too few or too many workers per situation. In particular, we observe per phase: the cost, points assigned, report importance (as reflecting situation importance), and reputation when: *i)* the active core community shrinks to 20 workers while encountering 50 situations (20u/50s); *ii)* a balance of workers and situations (100u/25s); *iii)* many active workers but only a few situations (100u/5s).

A surplus in situations (20u/50s) causes workers to become highly engaged, resulting in rapid reputation rise (Fig 7 bottom) coupled with extremely high values of accumulated rewarding points (Fig 6 inset). Costs per report remain low as duplicates become less likely with many situations to select from (0.18 duplicates per report). Here, CIS3 promises more reward for already highly-rated reports to counteract the expected inability to obtain sufficient worker input for all situation (on average 22 reports per phase out of 50). Subsequently, the authority receives correct ratings for reports and can focus on processing the most important ones. Compare the importance of addressed situations in Figure 7 top. A surplus in active workers (100u/5s) suffers from the inverse effect. As there is little to do, reputation and rewards grow very slowly. Perceiving little benefit, workers may potentially leave while the authority has a difficult time distinguishing between malicious and non malicious workers. Configurations (100u/5s) and (100u/25s) manage to provide reports for all situations, therefore having average report importance remaining near 0.5, the average importance assigned across situations.

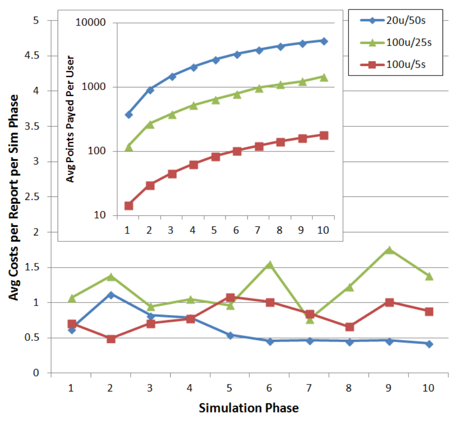


Fig. 6: Costs per report incurred at various combinations of worker and situation count.

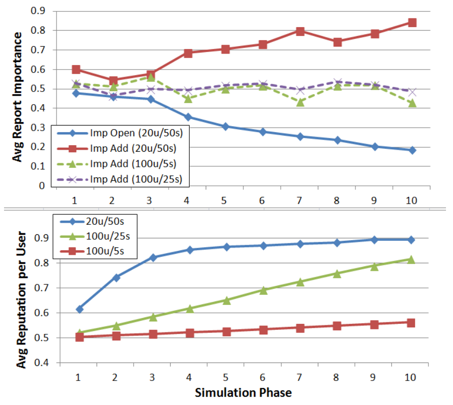


Fig. 7: Reputation acquired by workers (bottom), and report **imp**ortance **add**ressed, respectively remaining **open** (top).

Experiment 3: Effect of Malicious Workers. Here we evaluate the effects of an increasing amount of malicious workers on cost when applying CIS3. Figures 8 and 9 detail cost and reputation for 0%, 20%, 30%, 40%, and 50% malicious workers. All workers are considered of equal, medium reputation 0.5 upon simulation start. The drop in costs across time (observed for all configurations) highlights that the mechanism indeed learns to distinguish between regular, trustworthy workers and malicious workers. The irregular occurrence of undetected duplicates cause the fluctuations in cost apparent for 0% and 20% malicious workers. Beyond that, however, costs are primarily determined by low accuracy induced by malicious workers. CIS3 appears to work acceptably well up to 20% malicious workers. Beyond this threshold harsher reputation penalties and worker blocking (when dropping below a certain reputation value) need to be put in place. In severe cases lowering the default reputation assessment might be applicable but requires consideration of side effects (i.e., thereby increasing the entry barrier for new workers).

6.3 Limitations and Discussion

Simulations of complex socio-technical processes such as the use-case presented here can only cover particular aspects of interest, never all details. Thus any results in terms

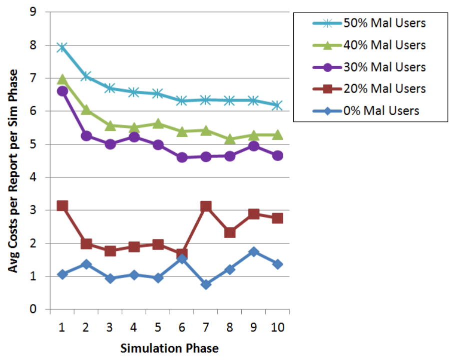


Fig. 8: Costs per report incurred due to various level of malicious workers.

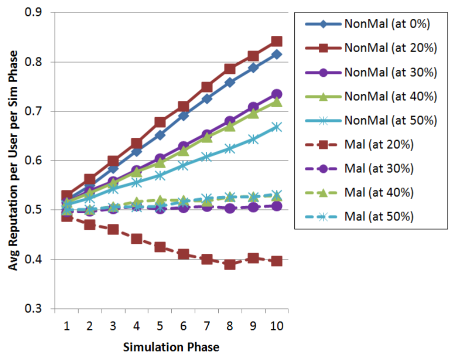


Fig. 9: Average reputation acquired by malicious and non-malicious workers.

of absolute numbers are unsuitable to be applied directly in a real-world systems. Instead, the simulation enables incentive scheme engineers to compare the impact of different design decisions and decide what trade-offs need to be made. The simulation outcome provides an understanding what mechanisms might fail earlier, which strategies behave more predictably, and which configurations result in a more robust system design.

In particular, the presented comparison of CISs in Experiment 1 gives insight into the impact of overpaying as well as indicating that the CIS3 would do well to additionally include a mechanism to limit submissions and better reward the action of flagging the duplicates. Experiment 2 provides insights on the effect of having too few or too many workers for a given number of situations. It highlights the need to adjust rewards and reputation in reaction to shifts in the environment and/or worker community structure. Experiment 3 provides insight into the cost development in the presence of malicious worker and highlights the potential for mechanism extension.

True advantages of our simulation approach can be appreciated when used together with an automated system for incentive management (e.g., [17]). The incentive management system can then be used to provide a number of simulation parameters, so that the system architect can quickly set up a simulation environment resembling the real system. The new incentive mechanisms and their combinations can then be tried out and the feedback sent to the incentive management system which can then re-adjust its incentive scheme setup.

7 Conclusion and Outlook

In this paper we presented a methodology for modeling and simulating incentives in crowdsourcing environments, highlighting the challenges with which the system architects are faced, and pointing to a possible way of alleviating them. We intend to continue our work on modeling incentives, trying to devise suitable models for different and more complex processes and environments. In the long run, we intend to develop a uniform and generally applicable language/notation for the description and ad-hoc instantiation of various incentive processes on real-world socio-technical systems.

Acknowledgments

This work has been partially supported by the EU FP7 SmartSociety project under Grant No. 600854 and the Austrian Science Fund (FWF) under Grant No. J3068-N23. The first author of this paper is a student of the Vienna PhD School of Informatics.

References

1. Adar, E.: Why I Hate Mechanical Turk Research (and Workshops). Control (2011), http://www.cond.org/eadar-crowdsourcing-workshop.pdf
2. Bloom, M., Milkovich, G.: The relationship between risk, incentive pay, and organizational performance. The Academy of Management Journal 41(3), 283–297 (1998), http://www.jstor.org/pss/256908
3. Dorn, C., Edwards, G., Medvidovic, N.: Analyzing design tradeoffs in large-scale socio-technical systems through simulation of dynamic collaboration patterns. In: Meersman, R., Panetto, H., Dillon, T.S., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I.F. (eds.) OTM Conferences (1). Lecture Notes in Computer Science, vol. 7565, pp. 362–379. Springer (2012)
4. Gal, Y., Grosz, B., Kraus, S., Pfeffer, A., Shieber, S.: Agent decision-making in open mixed networks. Artif. Intell. 174(18), 1460–1480 (Dec 2010), http://dx.doi.org/10.1016/j.artint.2010.09.002
5. Gilbert, N., Troitzsch, K.: Simulation for the social scientist. Open University Press, McGraw-Hill Education (2005)
6. Heckman, J., Smith, J.: What do bureaucrats do? The effects of performance standards and bureaucratic preferences on acceptance into the JTPA program. Advances in the Study of Entrepreneurship Innovation and Economic Growth 7, 191–217 (1996)
7. Kraus, S., Hoz-Weiss, P., Wilkenfeld, J., Andersen, D.R., Pate, A.: Resolving crises through automated bilateral negotiations. Artificial Intelligence 172(1), 1 – 18 (2008), http://www.sciencedirect.com/science/article/pii/S0004370207001051
8. Laffont, J.J., Martimort, D.: The Theory of Incentives. Princeton University Press, New Jersey (2002)
9. Little, G., Chilton, L.B., Goldman, M., Miller, R.: Exploring iterative and parallel human computation processes. Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA '10 p. 4309 (2010), http://portal.acm.org/citation.cfm?doid=1753846.1754145
10. Macal, C.M., North, M.J.: Agent-based modeling and simulation. Proceedings of the 2009 Winter Simulation Conference (WSC) pp. 86–98 (Dec 2009), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5429318
11. Mason, W., Watts, D.J.: Financial incentives and the performance of crowds. In: Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP '09). vol. 11, pp. 77–85. ACM, Paris, France (May 2009), http://portal.acm.org/citation.cfm?doid=1809400.1809422
12. Milinski, M., Semmann, D., Krambeck, H.J.: Reputation helps solve the 'tragedy of the commons'. Nature 415(6870), 424–426 (Jan 2002), http://dx.doi.org/10.1038/415424a
13. Prendergast, C.: The provision of incentives in firms. Journal of economic literature 37(1), 7–63 (1999), http://www.jstor.org/stable/2564725

14. Rockenbach, B., Milinski, M.: The efficient interaction of indirect reciprocity and costly punishment. Nature 444(7120), 718–723, `http://dx.doi.org/10.1038/nature05229`
15. Scekic, O., Truong, H.L., Dustdar, S.: Modeling rewards and incentive mechanisms for social bpm. In: Barros, A., Gal, A., Kindler, E. (eds.) Business Process Management, Lecture Notes in Computer Science, vol. 7481, pp. 150–155. Springer Berlin Heidelberg (2012), `http://dx.doi.org/10.1007/978-3-642-32885-5_11`
16. Scekic, O., Truong, H.L., Dustdar, S.: Incentives and rewarding in social computing. Commun. ACM 56(6), 72–82 (Jun 2013), `http://doi.acm.org/10.1145/2461256.2461275`
17. Scekic, O., Truong, H.L., Dustdar, S.: Programming incentives in information systems. In: In Proceedings of the International Conference on Advanced Information Systems Engineering - CAiSE 2013. Forthcoming. (2013), `http://tinyurl.com/std-incentives-caise13-pdf`
18. Semmann, D., Krambeck, H.J., Milinski, M.: Strategic investment in reputation. Behavioral Ecology and Sociobiology 56(3), 248–252 (Jul 2004), `http://dx.doi.org/10.1007/s00265-004-0782-9`
19. Tokarchuk, O., Cuel, R., Zamarian, M.: Analyzing crowd labor and designing incentives for humans in the loop. IEEE Internet Computing 16, 45–51 (2012)
20. Wedekind, C., Milinski, M.: Cooperation Through Image Scoring in Humans. Science 288(5467), 850–852 (May 2000), `http://dx.doi.org/10.1126/science.288.5467.850`
21. Wegner, R.: Multi-agent malicious behaviour detection. Phd thesis, University of Manitoba (2012), `http://hdl.handle.net/1993/9673`