Modeling Rewards and Incentive Mechanisms for Social BPM

Ognjen Scekic, Hong-Linh Truong, and Schahram Dustdar

Distributed Systems Group, Vienna University of Technology, Vienna, Austria
{oscekic,truong,dustdar}@infosys.tuwien.ac.at

Abstract. Social computing is actively shaping Internet-based business models. Scalability and effectiveness of collective intelligence are becoming increasingly attractive to investors. However, to fully exploit this potential we still have to develop crowd-management frameworks capable of supporting rich collaboration models, smart task division and virtual careers. An important step in this direction is the development of models of rewarding/incentivizing processes. In this paper, we conceptualize and represent rewarding and incentive mechanisms for social business processes. Our techniques enable definition, composition, execution and monitoring of rewarding mechanisms in a generic way.

Keywords: rewards, incentives, social computing, socially-enhanced BPM

1 Introduction

Incentives and rewarding are inseparable parts of business processes today. Their main purpose is to align the interests of workers and employers. By stimulating workers with various monetary, material and psychological rewards the employer can enhance productivity, quality, knowledge, collaboration, leadership, and other positive qualities in the company. Even more beneficial are the selective effects of the incentives [4]. Each particular incentive usually targets to enhance a single aspect of worker's performance. This can lead to workers starting to exhibit various dysfunctional types of behavior, meant to increase productivity only in segments targeted by the incentive while neglecting the others. This is why in practice often a number of simple incentives are combined together targeting each other's unwanted consequences.

Motivation: Social Computing for business is expected to grow substantially in the coming years[1]. Crowdsourcing is already a well-established business model, but it is characterized by exploitation of unstructured crowds of independent workers performing small, simple tasks. Collaboration models on the web are becoming richer, evolving from traditional company to crowdsourcing to social business. In the future, we expect that the development and adoption of novel business models involving social business processes will match or extend the contemporary processes in traditional companies. These business processes (so-called *Social BPs)* will (partially) rely on dynamic, distributed workforces, structured in problem-related, ad-hoc assembled teams of professionals. The new

business reality will require advanced organizational and management structures for the workers, intelligent task division and distribution, with the advent of long-lasting "virtual careers".

These trends inevitably require advanced crowd-management capabilities in future social computing platforms, including novel rewarding/incentive frameworks exploiting the advantages of vast amounts of digital productivity records, cheap peer evaluation, psychological techniques, etc. However, to the best of our knowledge, existing social computing platforms lack techniques for formulating, composing and automatically deploying incentive mechanisms.

Contribution: We identify the basic composing parts and conceptualize a model for representing most real-world incentive mechanisms using rewarding rules and events. Our model supports reasoning and acting along quantitative, structural and temporal data associated with teams of workers, allowing composition of incentive mechanisms into complex schemes.

Related work: Most related work in the area of rewarding and incentives originates from economics, organizational science, psychology and applied research. It can be used to classify and substantiate the basic rewarding approaches and expected outcomes, and to simulate the responses to incentive strategies. The principal economic theory treating incentives today is the *Agency Theory*[3]. We use many of the basic findings from this theory implicitly in the foundation of our model. The paper [6] presents a comprehensive review and comparison of different incentive strategies. In computer science, the topic has been treated only within application-specific contexts so far, e.g., social networks[8], human microtask platforms[5,7], peer-to-peer networks, etc. However, to the best of our knowledge, the topic has not been previously addressed elsewhere in a comprehensive, general manner.

2 Modeling Rewards and Incentive Mechanisms

Incentive is any activity employed by the system to stimulate or discourage certain worker activities before the actual execution of those activities. **Reward** is any kind of recompense for worthy services rendered or retribution for wrongdoing exerted upon workers after the completion of the activity.

Based on a thorough review of classical economic literature on the topic we identified the three components that every *incentive mechanism* consists of:

(1) **Evaluation methods** serve to assess the quality of worker's performance from different aspects. They provide inputs for making a decision whether to apply a reward/sanction. (2) **Incentive conditions** represent the business logic behind the incentive mechanism. They contain the rules for application of rewarding actions and take the evaluation results as inputs. (3) **Rewarding actions** are concrete measures taken against individuals or teams to influence their future behavior.

Any concrete incentive mechanism can be expressed as *incentive rules* containing these three components, in a system-independent way. It is is then possible to translate automatically such rules into queries and actions upon a **re-**

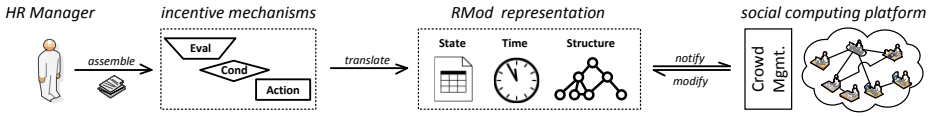

Fig. 1. Supporting incentives in social computing environment

warding model (RMod), representing the following aspects of a real-world system:

State represents quantitative state of the system. It includes global attributes and individual worker attributes, representing different performance metrics (QoS). These metrics are part of the business logic of a company, and, as such, represent an input to our model. **Time** is expressed as a collection of time-annotated records of past and future worker interactions, supporting various time conditions and constraints. **Structure** allows representation and manipulation of various types of relationships among workers.

The RMod represents an abstraction layer between an actual real-world platform that manages worker teams and client's system-independent representation of an incentive mechanism (Figure 1). At any time, the RMod must mirror the current state of the external system. RMod must be versatile and general enough to model many different real-world platforms and support application of any incentive mechanism. Therefore, it must stay decoupled of both. This allows for seamless switching between different incentive strategies and application of same strategies on different systems. The aforementioned implies a highly abstract and minimalistic RMod that fits to various underlying systems, and is able to support expressing a range of specific incentive mechanisms by the end-users. We believe that incentive mechanisms should be expressed *declaratively*, i.e. without explicit control flow and data manipulation. As in real life, a client then needs only to specify what incentive actions should be applied and upon which conditions. The condition evaluation and actual scheduling and execution of incentive actions should be encoded *imperatively* at runtime in RMod, transparent to the client. The benefit for the client is the ability to specify human-friendly, portable, scalable, composable and modifiable incentive strategies. In the remainder of this paper we focus only on RMod.

2.1 The Rewarding Model (RMod)

To develop a general-purpose model we adopt simple and abstract representations. An organization, referred to as *principal*, employs a group of *workers* to perform a complex process, consisting of multiple tasks. The principal uses a system that splits, assigns, and in every other aspect manages task lifecycles. A worker is assigned a (sub)task to perform in a given time and agrees to be subject of incentive evaluations. Workers can work individually on assigned tasks, or have a formalized organization or relationship with the principal (be employed, be part of teams, have managers, etc.) Workers can be paid and/or otherwise

rewarded for their contribution. Principal's knowledge of the task progress is obtained by periodic *messages* (updates) that he receives from workers and subsequent reasoning over that data. Similarly, his influence over a worker (penalty, promotion, bonus, etc.) is performed via legally-binding messages to the worker. With this assumption, a worker can be represented by a real human, e.g., as a member of a Social Compute Unit (SCU) [2] and via a Web Service interface. Without loss of generality, we can assume that the principal employs a group of humans that perform their work via Web Services, by contracting a third-party human-labor platform that fully takes care of task and worker management. That way, we can focus solely on providing the services of management of incentives and rewards (**RI Management**).

Task is the basic working unit. Workers are rewarded for working on a particular task within the task's timeframe, although the outcome of the evaluation can also depend on the history of previous contributions. Therefore, the lifetime of a worker is not related to the duration of the task. The principal maintains his own view of the workers and the relations between them in a *community graph*. The nodes in the graph represent the workers, while the edges represent different real-world relationships among the workers (e.g., records of past collaborations, trust, dependencies, managerial relations, etc.). In addition, each node is described by a set of attributes. The attributes may represent task-specific (short-lived) or permanent records of worker's performance. This is the most general representation possible. However, in practice we expect this model to be coupled with a real-world system, so the nodes and relations can be mapped to entities in a system that uses, e.g., BPEL4People, SCU or a custom platform for managing tasks and workers.

Each task is performed in iterations. *Iteration* length is measured in clock ticks. *Clock tick* is the basic unit of time measurement. Worker's progress is submitted upon iteration expiry so the system can update the QoS metrics. Iteration is the basic unit for splitting, monitoring and evaluating task execution in runtime. Iteration cycle length is tunable to allow better runtime adaptability, as the iteration length can be a significant factor when evaluating results and can affect the performance of the team. In order to represent history of past behavior, as well as scheduling of future performance evaluations and rewarding actions, we include in the model the notions of *timeline* and *event*. The timeline is a time-stamped collection of past and future event records. An **event** object encapsulates an executable action and a timestamp. Events are interpreted by the system as orders or suggestions to the system itself or particular workers, e.g, to notify a worker to increase QoS level in future iterations, to dissolve a team, invite new workers, terminate contracts, etc. Events can be generated by the system itself or originate from an incentive mechanism. They can target individual workers, groups of workers or global system properties, depending on the query that forms part of the action contained in the event object.

An event can be in two states: *scheduled* and *past*. **Scheduled events** are used to enforce/influence future behavior. They contain information to execute performance measurements, evaluations or concrete rewarding actions in a spec-

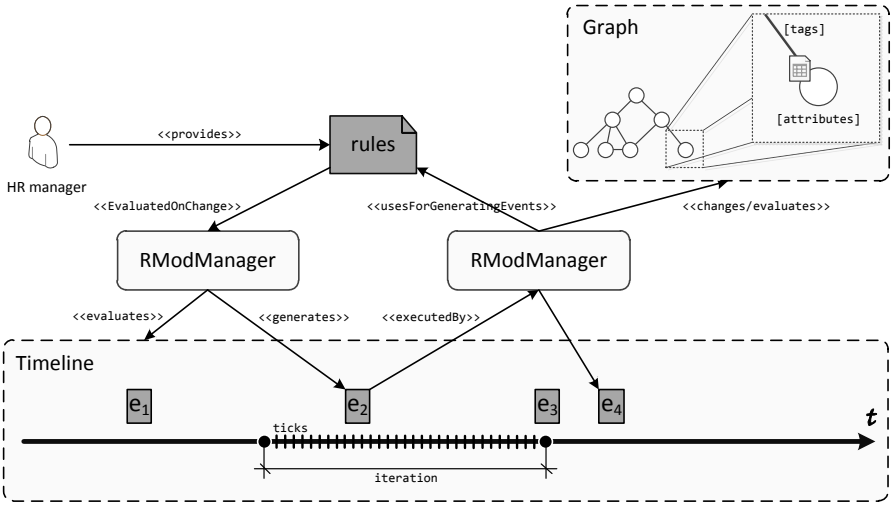

Fig. 2. Components and interactions in RMod

ified moment in the future. Scheduled events can be canceled or re-scheduled when needed. The timestamp can be expressed either in iterations or clock ticks. Time expressed in clock ticks is fixed, whereas time expressed in iterations is automatically recalculated to an appropriate clock tick if the iteration duration is altered. This can be useful in many real-world situations. For example, we want Christmas bonuses to be paid out on a fixed date, while if a process stage is prolonged due to some unexpected events, we want to reschedule the current iteration and perform the rewarding only at its end. When the time to execute an event is reached, the contained action is executed and the results stored back in the event, which is then archived and put into past state. After that point, the purpose of the **past event** is to serve as a historical reference for future evaluations of workers. An event execution can generate new events, or perform modifications of the team structure and worker attributes. Events are initially generated by executing *rewarding rules*. The rules encode an actual rewarding mechanism provided by the principal. Those rules that fulfill the execution condition generate new event objects to be stored in the timeline. Rules also contain the various bits of logic that get embedded into event objects.

Figure 2 describes a typical working cycle of our Rewarding Model (RMod). Rules provide the necessary logic for performing evaluations and rewarding actions. At every clock tick rules get evaluated. Only the rules that fulfill a logical condition will be triggered to execute. The rules examine the current state of the model and, if an action needs to be performed, produce one or more events. The action contained in the event will include the logic contained in the rule. The events then get stored into the timeline. When the appropriate time comes, the events get executed, modifying the attributes and the graph structure, and

possibly spawning new events. The *RModManager* boxes in Figure 2 represent the system that implements the functionalities and manipulates RMod.

The RMod allows us to express and compose different incentive mechanisms. For example:

- *"At the end of iteration, award each contributor who scored better than the average score of his neighbors in that iteration."*
- *"Reward every worker (contributor) who within the last* n *iterations scored a score* t *or greater in at least* k *iterations (*$k \leq n$*)."*
- *"Assign the person with most check-ins at a place a 'Mayor' badge."*
- *"Unless the productivity increases to a level* p *within* n *next iterations, replace team's current manager with the most-trusted of his subordinate workers."*

3 Conclusions and Future Work

Considering the lack of general methods for defining and composing incentive mechanisms for social business process, in this paper we analyze common components of incentive mechanisms and devise novel techniques for modeling and representing incentive schemes suitable for emerging, social-based processes. Our Rewarding Model supports expressing, composing and executing customized, complex incentive schemes. We are currently developing a prototype and intend to illustrate our techniques with real-world scenarios that cover the most important aspects of rewarding. Furthermore, we are integrating our model with the Social Compute Unit[2], a framework for on-demand virtual team provisioning for managing distributed large-scale software systems in clouds.

References

1. Austin, T., Drakos, N., Rozwell, C., Landry, S.: Business Gets Social, `http://www.gartner.com/DisplayDocument?doc_cd=207424&amp;ref=g_noreg`
2. Dustdar, S., Bhattacharya, K.: The Social Compute Unit. Internet Computing, IEEE 15(3), 64–69 (2011)
3. Laffont, J.J., Martimort, D.: The Theory of Incentives: The Principal-Agent Model. Princeton University Press, New Jersey (2002)
4. Lazear, E.P., Shaw, K.L.: Personnel economics: The economist's view of human resources. Journal of Economic Perspectives 21(4), 91–114 (2007)
5. Mason, W., Watts, D.J.: Financial incentives and the "performance of crowds". ACM SIGKDD Explor. Newsl. 11(2), 100–108 (2010)
6. Prendergast, C.: The provision of incentives in firms. Journal of economic literature 37(1), 7–63 (1999)
7. Shaw, A.D., Horton, J.J., Chen, D.L.: Designing incentives for inexpert human raters. In: Proceedings of the ACM 2011 conference on Computer supported cooperative work. pp. 275–284. CSCW '11, ACM (2011)
8. Yogo, K., Shinkuma, R., Takahashi, T., Konishi, T., Itaya, S., Doi, S., Yamada, K.: Differentiated incentive rewarding for social networking services. In: Applications and the Internet (SAINT), 10th IEEE/IPSJ International Symposium on. pp. 169–172. IEEE Computer Society (2010)