SC-PROV: A Provenance Vocabulary for Social Computation[*]

Milan Markovic, Peter Edwards, and David Corsar

Computing Science & dot.rural Digital Economy Hub, University of Aberdeen, Aberdeen, AB24 5UA
{m.markovic,p.edwards,dcorsar}@abdn.ac.uk

Abstract. In this paper we present SC-PROV - an extension to PROV-O and P-PLAN that is designed to capture the provenance of social computations.

1 Introduction

The Web has enabled the rapid growth of various forms of social computation [RG13] - hybrid workflows that consist of tasks executed by both computational agents and humans. Such workflows are typically used to solve problems that are difficult for machines (e.g. image classification). We have previously argued [MEC13] that existing social computation systems suffer from a lack of transparency, that makes decisions about the reliability of participants and the quality of generated solutions difficult. We believe that such transparency issues could be addressed by recording the provenance of social computation executions. While PROV-O[1] can be used to document retrospective provenance (such as execution traces of workflows) this would not include details of why or how a workflow was expected to execute [MDB+13,GG12]. For this purpose, P-PLAN[2] extends PROV-O with the ability to document workflow plans in terms of steps and variables. However, in order to improve the transparency of social computations and support enhanced reasoning about human participants (and their contributions), we believe that plans should also include additional elements describing important characteristics of the social computation [MEC13]. These include pre and post conditions associated with social computation tasks, e.g. a participant must be an English speaker and the outcome has to be validated by two additional participants. Also needed is a means to describe incentives that are associated with successful completion of a task (e.g. receive 10 points). We have developed SC-PROV as an extension of PROV-O and P-PLAN to enable descriptions of such conditions and incentives as part of the social computation plan. In addition, SC-PROV enables such concepts to be mapped to a provenance record describing the execution trace.

[*] The research described here is supported by the award made by the RCUK Digital Economy programme to the dot.rural Digital Economy Hub; award reference: EP/G066051/1

[1] http://www.w3.org/TR/prov-o/

[2] http://vocab.linkeddata.es/p-plan/

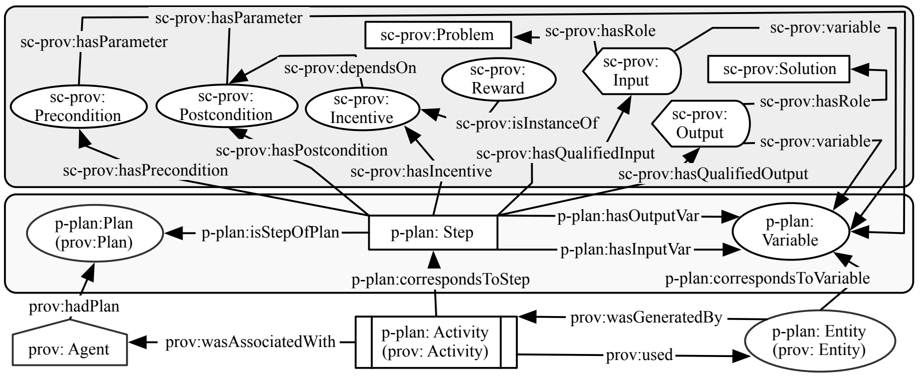

Fig. 1. PROV-O and corresponding extensions including P-PLAN and SC-PROV.

2 Model Description

SC-PROV reuses the concepts defined by P-PLAN to enable the basic structure of a social computation workflow to be captured. P-PLAN describes plans in terms of *p-plan:Step* and *p-plan:Variable*. Steps represent activities that should be executed as part of the workflow. A variable can be related to a particular step as either an input or output variable. The *p-plan:correspondsToVariable* property maps a variable to a *p-plan:Entity* that was used or generated by a *p-plan:Activity* during execution. Similarly, steps can be mapped to a *p-plan:Activity* via the *p-plan:correspondsToStep* property. To augment such descriptions of tasks in the context of social computations, SC-PROV defines *sc-prov:Precondition*, *sc-prov:Postcondition*, *sc-prov:Incentive* and *sc-prov:Reward*. A precondition defines a subclass of *prov:Entity* representing constraints that should be satisfied before a plan step can be fulfilled (e.g. the required location of a human worker). A precondition is associated with a step via the *sc-prov:hasPrecondition* property and can be linked to parameters of type *p-plan:Variable* (e.g. GPS coordinates) using the *sc-prov:hasParameter* property. As every precondition is a subclass of *prov:Entity*, it can be linked to the retrospective provenance described by PROV-O. For example, a provenance record might include a *prov:Activity* that evaluated whether the precondition was satisfied and therefore used the entity representing the precondition as well as entities corresponding to its parameters. A postcondition defines a subclass of *prov:Entity* representing constraints that should be satisfied after the completion of a particular step for it to be considered successful. For example, a postcondition of a human task might require an agent to produce a solution that should be validated at least by another two agents. The properties *sc-prov:hasPostcondition* and *sc-prov:hasParameter* are analogous to those described for *sc-prov:Precondition*. The *sc-prov:Incentive* class is a subclass of *prov:Entity* representing an incentive associated with the successful completion of a task. This concept can be understood as a thing (e.g. £10, increased knowledge, etc.) that would be realised by a worker following successful completion of a task. An incentive is associated with a step via the *sc-prov:hasIncentive* property. A property *sc-prov:dependsOn* is defined to link the concept of the incentive and a postcondition, where the postcondition is used to describe circumstances under which a worker should receive the reward described by the incentive. The concept of *sc-prov:Reward* defines a subclass of *prov:Entity* representing a realisation of the promised incentive (e.g. a voucher worth £10). The *sc-prov:isInstanceOf* property can be used to map a promised task incentive with such a reward. In a social computation context, there are two important types of variables, namely problems and solutions, which describe the purpose and outcomes of the computation. Using P-PLAN concepts, a problem would be described by a set of variables that represent a problem statement and serve as an input to a step (e.g. human task). Similarly, a solution would be described by a set of variables that specify the answer to the problem. SC-PROV defines two *sc-prov:Role*'s (*sc-prov:Problem* and *sc-prov:Solution*) to describe the expected function of a variable in a step. The qualified relation[3] pattern is used to model properties *p-plan:hasInputVar* and *p-plan:hasOutputVar* as resources. For this *sc-prov:Input* describes a variable that will be taken as an input for a step and *sc-prov:Output* describes a variable that will be produced as an output of a step in the planned execution. Properties *sc-prov:hasRole* and *sc-prov:variable* can then be used to associate the role with a variable.

3 Future Work

In our future work we aim to develop a framework utilising the SC-PROV ontology to record the provenance of a number of social computations. The aim is to demonstrate the utility of provenance records documented using SC-PROV by evaluating the potential of such data to support reasoning about participants' trustworthiness and thus aid workforce selection.

References

GG12. D. Garijo and Y. Gil. Augmenting prov with plans in p-plan: scientific processes as linked data. In *Proceedings of the Second International Workshop on Linked Science 2012 - Tackling Big Data*. CEUR, 2012.

MDB$^+$13. P. Missier, S. Dey, K. Belhajjame, V. Cuevas-Vicenttin, and B. Ludaescher. D-prov: extending the prov provenance model with workflow structure. Technical report, School of Computing Science, Newcastle University, 2013.

MEC13. M. Markovic, P. Edwards, and D. Corsar. Utilising provenance to enhance social computation. In *The Semantic Web–ISWC 2013*, pages 440–447. Springer, 2013.

RG13. D. Robertson and F. Giunchiglia. Programming the social computer. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987), 2013.

[3] http://patterns.dataincubator.org/book/qualified-relation.html