HAC-ER: Tracking Provenance in Disaster Response

Trung Dong Huynh, Sarvapali D Ramchurn, and Luc Moreau

Electronics and Computer Science, University of Southampton
Southampton, SO17 1BJ, United Kingdom
{tdh,sdr,l.moreau}@ecs.soton.ac.uk

Abstract. HAC-ER is a prototype disaster management system designed to address some of the situational awareness and coordination challenges faced by emergency responders in real-world disaster environments. Thanks to the PROV standards, provenance of information and decisions made in HAC-ER are fully tracked across its distributed software components. In addition, a Provenance Agent continuously monitors the recorded provenance to keep stakeholders in the system aware of changes during an operation, helping them react to new information in a timely manner.

Keywords: data provenance, disaster recovery, human-agent collectives, provenance tracking

1 Introduction

Human-Agent Collectives for Emergency Response, or HAC-ER [3], is a prototype disaster management system designed to address some of the situational awareness and coordination challenges faced by emergency responders in real-world disaster environments. It interweaves humans and agents, both robotic and software, in social relationships that augment their individual and collective capabilities. HAC-ER thus demonstrates how such Human-Agent Collectives (HACs) can address key challenges in disaster response. The whole system consists of the following inter-dependent, but loosely-coupled, components:

- CrowdScanner: a HAC that integrates crowdsourcing to gather, interpret and fuse information from both trusted agencies and members of the public on the ground, and thereby determine priority areas for responders.
- Mixed-Initiative Multi-UAV Coordination: a HAC that involves agents running a distributed coordination algorithm to coordinate multiple UAVs, as well as human operators, with different roles, that interact with the agents through planning and monitoring interfaces. This system allows them to verify information reported by civilians and prioritise search areas.
- Mixed-Initiative Task Allocation: a HAC composed of a planning agent and responders at the disaster response headquarters and on the ground, who work together to determine a schedule for the completion of tasks as identified by the multi-UAV system.

An overview of the components in HAC-ER is provided in Figure 1, which also depicts the flows of information between them (see [3] for more details).

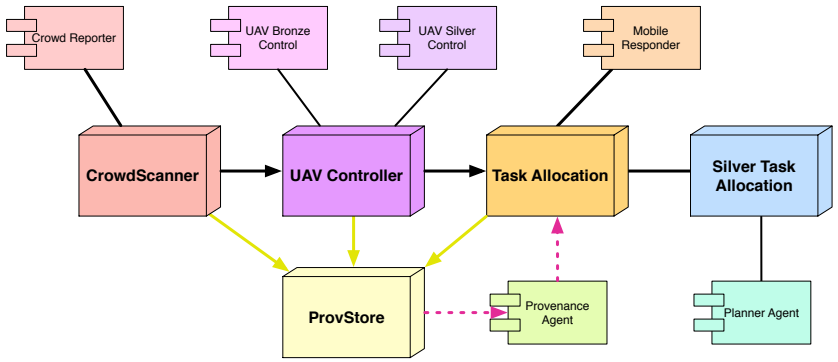

Fig. 1. The information flows between HAC-ER's components.

In order to provide accountability for both human and agent-based decision making, the provenance of information and decisions generated by the above HACs are fully tracked by a provenance infrastructure, which was built on top of the PROV standards by the World Wide Web Consortium [1] (see Section 2). In addition, a Provenance Agent was built to continuously monitor the recorded provenance data to keep stakeholders in the system aware of changes during an operation, helping them react to new information in a timely manner (see Section 3).

2 Tracking Information and Decisions in HAC-ER

In our system, when a piece of information is produced by one of the components, the component records the inputs it used in the production of that piece of information and the agent(s) and/or human(s) that were involved. Fig. 2 shows an example of provenance recorded by the UAV controller component. In the example, the entity `uav/target/9.2` represents a target identified by the UAV Controller component in HAC-ER. It was generated by a "UAV Verification" activity carried out by the UAV Silver commander, who verified the target `cs/target/9.1` reported by the CrowdScanner and marked it as an "Infrastructure Damage". Therefore, the UAV controller recorded that the verified target `uav/target/9.2` wasDerivedFrom `cs/target/9.1` and wasAttributedTo `uav_silver_commander`. Examining this provenance, either when the entity `uav/target/9.2` is used during an operation, or in a much later audit when the operation has finished, allows us and to track back to the origin of the information and to answer questions such as "who was responsible for this information" (via the wasAttributedTo relations) and "on which other information it depended" (via the wasDerivedFrom relations).

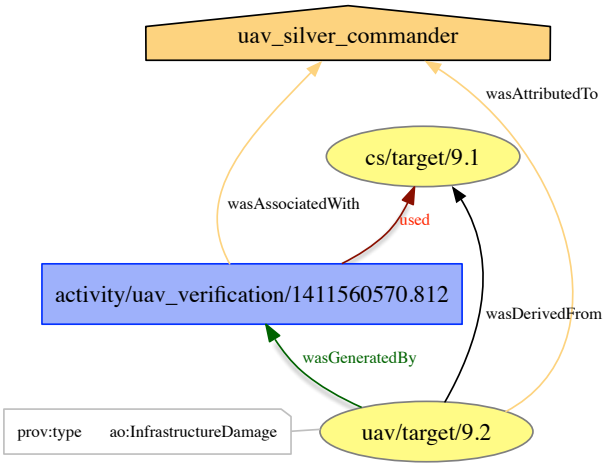

Fig. 2. A provenance graph in HAC-ER: The provenance of a target (`uav/target/9.2`) annotated by the UAV Silver Commander.

The capability to answer the latter is of particular importance during an operation as information our context is often uncertain and changing over time. This is demonstrated later in Section 3.

Provenance information recorded in HAC-ER is stored in a purpose-built repository for provenance, called ProvStore [2]. Individual HAC-ER components (i.e., the CrowdScanner, UAV Controller, and Task Allocation) record the provenance of information and data generated in each of their activities and report it to ProvStore once the activity completes (as shown in Figure 1). The provenance of any entity can then be retrieved from ProvStore whenever it is required. Fig. 3, for instance, shows the Sankey diagram generated from the provenance of `comfimed_plan/166` that was queried from ProvStore. As can be seen, the result goes back all the way to the original crowd reports (and their reporters) aggregated by the CrowdScanner (some entities omitted due to space constraints). It demonstrates the involvement of *all* the components in HAC-ER at various stages, leading to the eventual production of the `comfimed_plan/166`. Should any information in the chain of influences shown in Fig. 3 later be discovered as unreliable or incorrect, it would be possible to assess its potential impact by querying from ProvStore to find all of its *dependents* — information that was directly or indirectly derived from it (via wasDerivedFrom relations). This is indeed what the Provenance Agent monitors in order to ensure that responders on the ground and their commanders are aware of potentially adverse changes in dynamic situations that are typical in disaster areas. In the next section, we describe how this is achieved.

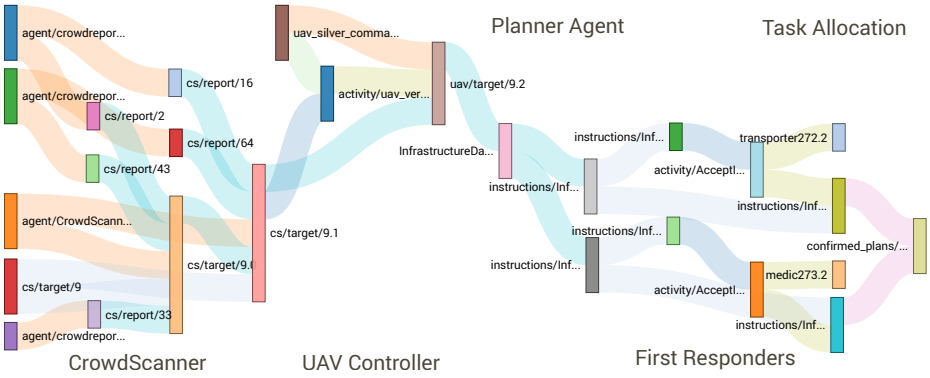

Fig. 3. The Sankey diagram from the provenance of the entity `confirmed_plans/166` showing the influences flowing (from left to right) to it, which was a plan that was originally generated by the Planner Agent and then later accepted by two responders, `transporter272` and `medic273`. The colours of the flows depict the type of influences, e.g. derivation (light blue), attribution (orange), generation (green), etc. The grey text labels were added to highlight the components in which the influences took place.

3 Monitoring Decisions

In an ongoing operation, available information is typically uncertain and/or incomplete. In HAC-ER, for instance, targets identified from aggregating crowd reports are inherently uncertain; new (and more trustworthy) reports from responders, for example, can invalidate targets that are already assigned to UAVs. UAV operators can also make mistakes when creating targets for UAVs and later, modify them. However, during the course of an operation, incorrect information may have already propagated to the Planner agent, resulting in assignments for responders. Those assignments may have subsequently been confirmed by a commander before the incorrect information is discovered. In order to be in control of such dynamic situations and to effectively manage such changes, the Provenance Agent tracks significant information changes at ProvStore that may have impacts on decisions already made. It works as follows:

- **Listening to information invalidation**: ProvStore provides an interface for external services to register to be notified when an event of interest occurs. Whenever prior information is invalidated, decisions and actions must be revised immediately. Therefore, the Provenance Agent asks ProvStore to notify it with any invalidation of existing information, represented as entities, asserted by one of HAC-ER's components.
- **Identifying potential impacts**: Upon receipt of a notification of invalidation, the Provenance Agent identifies all the dependents of the invalidated entity by querying ProvStore for the transitive closure over the wasDerivedFrom relation ending at the entity, i.e. all entities that can reach the invalidated entity via the wasDerivedFrom relation. For example, Fig. 4 shows a

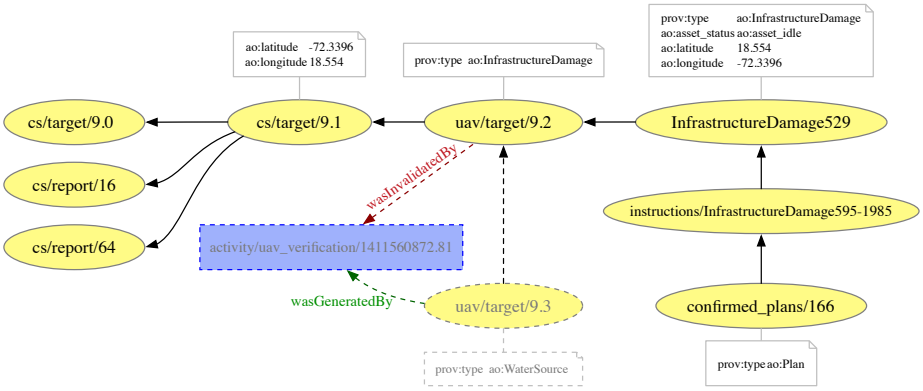

Fig. 4. A chain of derived information tracked across HAC-ER with the wasDerivedFrom relations between the entities depicted by the black edges. At a later point (showing in dotted lines), `uav/target/9.2` was revised and replaced by `uav/target/9.3`. Such events are monitored by the Provenance Agent to ensure that their potential impact on the dependants of the invalidated information is assessed appropriately.

chain of information derivations in HAC-ER. In this figure, the dependents of `uav/target/9.2` are the three entities on its right as it can be reached either directly or indirectly from those. If any of the dependents are confirmed decisions, e.g. entity `confirmed_plan/166`, the Provenance Agent will notify their owners in the next step.

– **Notifying affected stakeholders**: If some information or a decision was arrived based on some input(s) that are now invalidated, its "creator" may need to re-evaluate their decision. n such cases, the Provenance Agent will send a notification to the affected stakeholders, informing them of the fact that one of their assumptions has just become invalid. At the same time, the notification also provides the list of entities that are potentially affected by this, along with the provenance information that has all the details for further investigation. As an example, the entity `uav/target/9.2` (in Fig. 4), may have been mistakenly identified as an "Infrastructure Damage". The error was later discovered and corrected, resulting in a new version of the target (`uav/target/9.3`) and the original version invalidated. The dependents of `uav/target/9.2` were identified by the Provenance Agent as in the previous step, and a notification was sent to the commander who had confirmed the task allocation specified in `confirmed_plan/166`.

With the information and decisions in HAC-ER monitored by the Provenance Agent in tandem with ProvStore, our tests demonstrated that the system maintains situation awareness of the participants in a timely manner. Whenever the information used for task planning is invalidated, commanders were notified of the changes in less than a minute after they occurred. More importantly, the generic provenance tracking and monitoring mechanisms implemented here

are flexible and do not depend on this particular domain. Future changes in HAC-ER's individual components, or even the addition of new components, will not affect the existing operation of the Provenance Agent as long as all the components report the provenance of their operation to ProvStore as previously described in Section 2. This is facilitated by the wide availability of PROV-compatible libraries such as ProvPy[1] or ProvToolbox.[2]

Acknowledgements.

We gratefully acknowledge funding from the UK Research Council for project Orchid (grant EP/I011587/1).

References

1. Groth, P., Moreau, L.: PROV-Overview. An Overview of the PROV Family of Documents. W3c working group note, World Wide Web Consortium (Apr 2013), http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/
2. Huynh, T.D., Moreau, L.: ProvStore: A public provenance repository. In: Ludäscher, B., Plale, B. (eds.) 5th International Provenance and Annotation Workshop, IPAW 2014, Lecture Notes in Computer Science, vol. 8628, pp. 275–277. Springer International Publishing, Cologne, Germany (2015)
3. Ramchurn, S.D., Huynh, T.D., Ikuno, Y., Flann, J., Wu, F., Moreau, L., Jennings, N.R., Fischer, J.E., Jiang, W., Rodden, T., Simpson, E., Reece, S., Roberts, S.: HAC-ER: A disaster response system based on human-agent collectives. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 533–541. International Foundation for Autonomous Agents and Multiagent Systems, Istanbul, Turkey (2015)

[1] Available at https://pypi.python.org/pypi/prov.
[2] http://lucmoreau.github.io/ProvToolbox/.