Modelling the Provenance of Data in Autonomous Systems

Simon Miles, Steve Munroe, Michael Luck, Luc Moreau
School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
{sm, sjm, mml, L.Moreau}@ecs.soton.ac.uk

ABSTRACT

Determining the *provenance* of data, i.e. the process that led to that data, is vital in many disciplines. For example, in science, the process that produced a given result must be demonstrably rigorous for the result to be deemed reliable. A *provenance system* supports applications in recording adequate documentation about process executions to answer queries regarding provenance, and provides functionality to perform those queries. Several provenance systems are being developed, but all focus on systems in which the components are *reactive*, for example Web Services that act on the basis of a request, job submission system, etc. This limitation means that questions regarding the motives of autonomous actors, or *agents*, in such systems remain unanswerable in the general case. Such questions include: who was ultimately responsible for a given effect, what was their reason for initiating the process and does the effect of a process match what was intended to occur by those initiating the process? In this paper, we address this limitation by integrating two solutions: a generic, re-usable framework for representing the provenance of data in service-oriented architectures and a model for describing the goal-oriented delegation and engagement of agents in multi-agent systems. Using these solutions, we present algorithms to answer common questions regarding responsibility and success of a process and evaluate the approach with a simulated healthcare example.

Categories and Subject Descriptors

Computing Methodologies [**ARTIFICIAL INTELLIGENCE**]: Problem Solving, Control Methods, and SearchBacktracking; Computing Methodologies [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial IntelligenceMultiagent systems

General Terms

Algorithms, Standardization

Keywords

provenance, autonomy, process, agent-oriented design

1. INTRODUCTION

In many fields, including both business and science, there is a need among users, regulators and others to be able to determine the *provenance* of application results, i.e. the process that led to those results. The provenance of a data item can be used to gain a better understanding of the way that item was produced and to check its reliability from the qualities of the process that produced it. Distributed, electronic applications, such as those in agent-based e-Science, which involve multiple collaborating organisations, have an even stronger requirement for provenance, because results are produced in processes not wholly controlled by any one party. However, in these applications, determining the provenance of results depends on the components of the application collectively recording adequate *documentation* from which the provenance can be determined, and significant efforts have been made to develop systems and data models to support this [2, 19]. While these efforts are generating valuable results, they only address *what* has happened rather than *why* it has happened. Indeed, determining and understanding why a particular result has come about is especially important when autonomous entities are involved in an application because, in such systems, the intent behind any process that takes place is not fixed in the original design of the application and so cannot be understood solely by examining that design.

We argue that, to be able to answer questions that arise in autonomous systems, we need to consider not just the actions performed by agents in those systems but also the *goals* they are attempting to achieve. This allows us to address questions such as the following.

- Who was responsible for effect E?
- Does the effect E match what was intended to happen?
- What is the reason (both causal and intentional) of effect E?

Being able to answer such questions allows developers to take account of the intentions of participating agents when auditing an application, assessing the reliability of agents or checking whether the application is being used for conflicting purposes by multiple users. While absent from current Grid computing approaches, the modelling of autonomy has been addressed in the context of agent-based systems. Therefore, to address the above limitations, we adapt an existing provenance model [8] and augment it with notions of autonomy and goals in multi-agent engagement and cooperation [11]. In this paper, we present the integrated data representation of provenance and autonomous action, specify algorithms for determining the answers to questions such as those above and evaluate the approach in a healthcare example.

1.1 Provenance

It is essential to know the provenance of entities when answering questions in many domains, whether it is in judging the authenticity of a piece of art, assessing the validity of scientific results, determining where a fault originated in a manufacturing process, and so on (see [14] for a survey of requirements for provenance). To determine the provenance of an item in a system, adequate documentation must have been recorded regarding the processes in which the item was involved, which we call *process documentation*. In fact, if we were to answer *all* questions about an entity's provenance we would need access to records of all events and entities that had *caused* the entity to be as it is.

With complete knowledge, all information about a process in a system can be represented as data exchanged between agents in that process and causal relationships between those exchanges, i.e. one message was sent because another message was received. However, when it comes to representing the internal *volition* for an autonomous agent performing an action, this model in itself does not provide the most *useful* account of what occurred. Moreover, where people are involved in processes, it is not feasible to determine and record the source of information that ultimately caused them to act in a certain way. For example, to describe the cause of a person eating an orange as a reaction to the trigger of hunger is correct but misses crucial information that is not easily encoded as receipt of information: that the person likes oranges, that they are not eating meat because they are on a diet, and so on. While we can document the *cause* of a person's action, we cannot determine the *reason* for it (a distinction made in [20]).

The concept of causality has been used in a range of ways, in particular in distributed systems literature [13, 18]. For our purposes, we distinguish between causes and reasons as follows. If *E was caused by C*, then E would not have occurred if C had not occurred, all else being equal. If *E had reason R*, then R was the intention behind E occurring. To take a simple software example, if running a program displays the time, then the *causes* of a time being displayed include the current local clock time of the machine on which the program is run and the fact that the program was run. On the other hand, the *reasons* for a time being displayed include the program author's intention that the program display the current time and the program executor's intention to determine the current time. In many cases, the reason behind something is also a cause of it: if the intention had not existed, there would be no trigger for the effect.

The reasons *why* a process is initiated are critical in understanding *why* a particular result has come about, and are especially important in situations in which autonomous agents are involved. While humans are clearly autonomous, it is also increasingly relevant to consider autonomy of machines, especially in emerging computational environments like the Grid [6, 7], in which large numbers of different and sometimes unknown entities co-operate across multiple organisations to deliver services, or achieve results, that would not otherwise be possible. Indeed, the notion of autonomy has been identified as a characterising feature of intelligent agents, which are increasingly being deployed across the range of computational systems. Existing provenance models, such as Buneman et al.'s model for determining provenance of database query results [4], do not consider autonomy, they merely model the actions involved once a process has been started.

1.2 Autonomy

To address this problem, we propose allowing the *goals* of autonomous agents, and the connections of such goals to the processes they trigger, to be made explicit in *process documentation*. To do this, we require a model of autonomy and of the relationships between autonomous and non-autonomous actors, and to integrate it with our model of process documentation. Notions of autonomy have been considered elsewhere [10, 11], and the agent relationships that underlie the kinds of interactions we envisage have been addressed in the *engagement chains* approach [11]. Here, an engagement chain refers to a sequence of interactions between agents, starting with the autonomous agent that initiated a process, going through to those agents undertaking specific activities to achieve the result, and all those agents and interactions between them.

To clarify the explanation and make the benefits more concrete, we introduce a motivating running example from the medical domain. This example is a simplified version of a process presented in work by Vazquez and Willmott [9, 15], and is detailed further in later sections as we use it.

On it being recorded that a patient is near death, a donor data collector, which may be a doctor or a software agent acting on a doctor's behalf, can decide to start the process of assessing the patient for donation of organs. This process involves testing blood for diseases, retrieving the electronic medical history of the patient and, on death, asking the patient's family for consent. All these actions may be carried out by different and disparate organisations. The output of the process is a decision on the suitability of the patient for donation.

In considering autonomy in relation to provenance, this paper makes three key contributions. First, it integrates the complementary models for process documentation and engagement chains, involving a mapping of concepts between approaches. Second, it defines how the goals of autonomous agents can be represented in XML, and how this then can be integrated into the schema defined for the process documentation model. Third, it provides an algorithm over data in the integrated model, allowing the responsibility and reason for effects in an application to be determined. In particular, we discuss how this can be used to determine whether the application is successfully fulfilling the intentions of those using it, and whether the way in which it is done is desirable.

In Section 2, we introduce conceptual and data models for documenting application processes. Then, in Section 3, we show how this can be integrated with the engagement chain model for representing autonomous agents, and define a common, generic data format to support this in Section 4. In Section 5, we present algorithms which apply to the integrated model and determine the responsibility, reason, success and desirability of process results. These are applied to a simulation of the example application above, for which we describe the implementation and results in Section 6. We discuss the wider applicability and practical issues in using the approach and conclude in Section 7.

2. PROCESS DOCUMENTATION

The *process documentation model* is used for representing past processes. By documenting processes in a generic, well-specified way, questions about the provenance of data, decisions and other outcomes of processes can be answered. The primary concept in the process documentation model is the *interaction* of *actors*, which are merely entities that perform actions (an autonomous agent is, therefore, a kind of actor). An interaction is the exchange of data and/or control between actors, corresponding to a single message in a distributed system. In the hospital example, we can identify several actors described by their roles in the process below.

- The *donor data collector* is the actor that initiates the process of obtaining a decision on a donor's suitability for organ donation.

- The *blood tester* is the actor that tests the donor's blood for diseases.

- The *consent obtainer* is the actor that obtains consent for donation from the patient's family.

- The *decison maker* is the actor that makes the final decision on donation.

Between these actors is a series of interactions. The donor data collector contacts the blood tester and the consent obtainer to request they perform their respective duties. On obtaining blood test results or a family consent decision respectively, the latter actors send these to the decision maker. The final decision is sent by the decision maker to the doctor responsible for the potential donor. This view of the process that takes place is simplified for brevity, and many other actors and interactions could be modelled (the methodology by which the most suitable model can be obtained is not covered in this paper but is available elsewhere [17]).

Process documentation comprises of *p-assertions*, which are assertions by actors about the processes they were involved in. As the process takes place, the actors involved record p-assertions regarding their activities in a *provenance store*. While the assertions that may be made by an actor depend on how that actor functions, we have specified a *process documentation data model* that acts as a common framework for p-assertions, allowing questions to be answered regarding whole processes spanning many actors and their supporting organisations. The data model is expressed in XML Schema, and we give snippets for the most significant parts later.

The data model is targeted at open or semi-open systems, where it is essential for independent actors to record documentation using the same model in order for that documentation to be collectively traversed afterwards. In such an environment, there is an increased possibility of incorrect documentation being accidentally or maliciously recorded. While there is no way to force independently deployed software to behave in a particular way, the data model allows those recording documentation to be held to account by requiring every p-assertion to be digitally signed by its asserter.

There are three types of p-assertion: interaction, actor state and relationship. We describe each in turn.

2.1 Interaction P-Assertions

The data contents of an interaction are apparent in process documentation as an *interaction p-assertion*. Here, either actor in the interaction, sender or receiver, can assert the data sent between them. For example, the donor data collector may assert that they sent a blood test request for a particular patient to the blood tester. The blood tester may assert that they received that request. The *content* of an interaction p-assertion is the data sent between actors and can be any XML element, as such data is application-specific and all formats need to be accommodated.

2.2 Actor State P-Assertions

The state of an actor at the time that it sent or received a message is documented in an *actor state p-assertion*. For example, the blood tester may assert the identities of those people allocated to perform the blood tests when a request is received, or the donor data collector may document the time (on their local clock) that they request a blood test, so that the duration taken for the process to complete can later be determined. Again, the content of an actor state p-assertion is application dependent and so can be any XML fragment.

2.3 Relationship P-Assertions

Finally, the causal relationships between events and data in the system, such as the sending and receiving of messages, is documented in *relationship p-assertions*. If documenting the causal relationship between two interactions, this will be an assertion by an actor that "I sent this message because I received that message". If between an interaction and an actor state, it will be an assertion by an actor that "I sent this message because this was my state."

In the example, there are three causal relationships between interactions to document (and, as with actors and interactions, causal relationships may be modelled at different levels of detail to best suit an application).

- The blood test results are produced (*effect*) because there was a request from the donor data collector to perform them (*cause*).

- The family consent decision was obtained (*effect*) because there was a request from the donor data collector to obtain it (*cause*).

- The decision on the suitability of the patient for donation (*effect*) was because of the blood test results and the family consent decision (*causes*).

2.4 P-Assertions in the Example

The three types of p-assertion can be seen in Figure 1, where we show the interaction of the four example actors. For every interaction between actors, there is an interaction p-assertion recording the message sent. For every causal relationship between one interaction and another, there is a relationship p-assertion connecting the *effect* with its *cause*. Finally, each of the four actors may assert something about its state during the process.

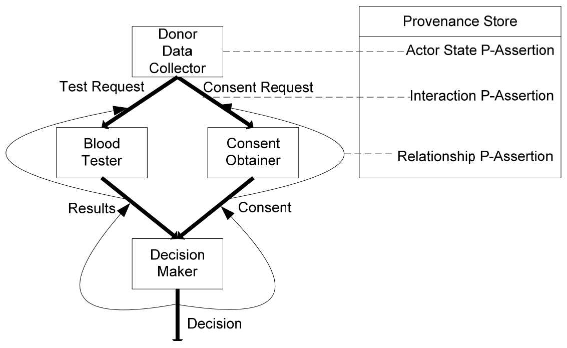

Figure 1: Process and p-assertions

Using process documentation, it is possible to determine the *provenance* of a result, i.e. the process that led to it. In the example, we can determine, from relationship p-assertions, that the donation decision was caused by the blood test results and the family consent decision, each of which were caused by a request from the donor data collector. The details of the data in the process, such as the actual donation decision and blood test results, are documented in interaction and p-assertions.

However, there are limitations with this model. In particular, it assumes that everything an actor does is a reaction to an explicit prior cause and so the *autonomy* of actors is not taken into account. Moreover, while it documents the *cause* of any event in the system, there is no specification of the *intention* behind an action.

3. INTENTIONS IN PROVENANCE

Given the limitations of the model described above, and our desire to account for *why* certain courses of action have been followed, we require a model in which intentionality can be explicitly represented and combined with the descriptions of *what* has occurred provided by the provenance model. It would then become possible to account for two different, yet complementary descriptions of past events:

- Reactive: what happened, how it came to happen.
- Intentional: why it happened, what was the reason for it happening.

As discussed, the reactive description can already be captured by the provenance model. However, the intentional description requires that we can represent the *reasons* why a particular course of action was taken. This kind of representation is naturally captured by computational notions of *goals*: descriptions of desired states that guide action. In turn, the ability to generate goals and use them as instigators of action captures our view of *autonomy* [12], which has been defined in computational systems as the ability to *"operate without the direct intervention of others, and have some kind of control over one's actions and internal state"* [21]. While some have argued for adjustable autonomy that can be increased or decreased depending on circumstances [1], or for different dimensions of autonomy [3], we see goal generation as absolute and defined in terms of *motivation*, or the ability to generate goals as result of internal drives.

Adopting such a model allows for a greater range of queries about systems. For example, if we restrict ourselves to entirely reactive descriptions of events in the hospital scenario, we can only answer queries that relate to what happened, e.g. that a decision was made on a patient's donor suitability. We cannot, however, answer queries relating to *why* a particular patient was submitted for screening in the first place. That is, we cannot ask for the *reasons* why a particular screening process was begun. To do this, we need *intentional* descriptions of events, which allow us to answer such a query by discovering the goal of the doctor or system in the hospital that initiated the screening process, of which there may be several. For example, a doctor may initiate the process in order to satisfy her goal to comply with hospital procedures, or to satisfy a goal to source an organ for a transplant patient, or to satisfy a combination of such goals.

In order that autonomy can be used within the model of provenance and not just remain theoretical abstractions, we require an explicit model of autonomy and goals. We consider this next.

3.1 Engagement Chains

The engagement chain model [11] presents a view of agent interaction as *goal driven* and *intentional*. Agents interact to satisfy goals that originate from *autonomous agents*, agents that possess the capability to generate goals. Such agents generate goals when an *intention* is formed for their satisfaction, where such intentions are controlled by higher level, non-derivative *motivations* [16]. For example, an agent may have a motivation for hunger which, when active, may cause the agent to generate a goal to eat. Once generated, the goal is met by performing *actions* dependent upon the agent's *capabilities*, e.g. seeking out a source of food. In many cases, however, an autonomous agent may generate a goal that it cannot meet alone, when the actions required are not part of the agent's capabilities. Here, the agent must obtain assistance from other agents, *server agents*, possessing the capabilities necessary.

In this model, a *direct engagement* is formed when a server agent adopts a goal from an agent possessing some goals, a *client agent*. While server agents are defined as non-autonomous, there is no restriction on client agents. [1] An *engagement chain* is a sequence of server agents each of which has been directly engaged by the preceding agent. At the head of the chain there must exist an autonomous agent, since only these agents can begin the formation of a chain by generating and passing on a goal.

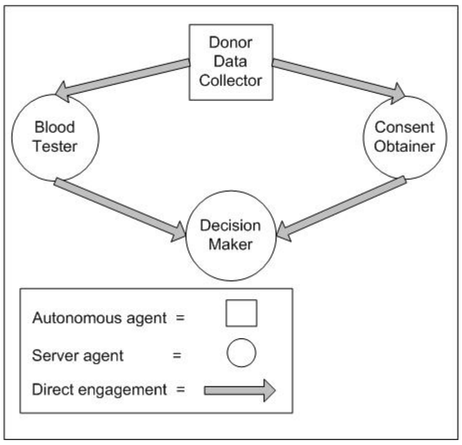

Figure 2: An example engagement chain

An engagement chain is shown in Figure 2, which shows a *donor data collector* initiating a donor screening process in order to satisfy one of its operational goals, to enter a dying patient into the organ donation process. To satisfy this goal, the donor data collector directly engages the *blood tester* sending it a message requesting a blood test for the patient as well as directly engaging the *consent obtainer*, directing it to ask for the patient's family's consent for organ harvesting. Both blood tester and consent obtainer then directly engage the *decision maker*, passing along the information each has collected. The decision maker makes the final decision about the patient's suitability for donation based on this information. In this example, the donor data collector is the autonomous agent generating and issuing a goal into the engagement chain. The blood tester, the consent obtainer and the decision maker are the server agents acting within their capabilities to satisfy the donor data collector's goal.

The provenance of an item in a system in which processes are triggered by an autonomous agent engagement is includes the direct engagements of server agents, where each satisfies a part of the autonomous agent's goal. For each of these agents, the goals they are attempting to achieve in performing their actions can be explicitly asserted, and so can be included in the answers to provenance related questions.

3.2 Limitations

While each of the models, process documentation and engagement chains, addresses aspects of the problem we are concerned with, they are distinct, come from very different perspectives, and

[1] In an extension [11], server agents are allowed to be modelled as autonomous so that, instead of direct engagements, a case of *cooperation* arises, since the autonomous server agent must be persuaded to adopt the goal. In this paper we restrict our analyses to situations in which server agents are strictly modelled as non-autonomous and leave the more complex analyses of cooperation between autonomous agents for future work.

have several limitations. By combining them, we can overcome these limitations and allow richer queries to be answered.

For process documentation, the primary limitation is that no notion of intention of the actors involved is present in the model. This means that it is not possible, in a generic, re-usable way, to answer questions of responsibility for effects in the world, nor whether the effects of processes match their intent.

For engagement chains, there are two limitations. First, there is no explicit notion of causality. This is apparent in that if two agents, A and B, have both engaged a third agent, C, at a particular point in time, then a querier cannot determine for which client C's engagements are made. That is, multiple engagement chains become indistinguishable in a system to anyone trying to determine the causes of engagements.

Second, server agents in a real system may not be able to adopt goals exactly as their client would express them. For example, a Web Service interface only allows requests of particular forms to be made to the service, and these may not exactly articulate the goals of the client. Therefore, an engagement chain may involve agents further down the chain receiving modified versions of goals, and it may not be possible for a querier to determine from the agent itself which goals it has been engaged to achieve.

3.3 Integration of Models

To combine approaches, and overcome the above limitations, we first perform a mapping between concepts in the two models, as shown in Table 1. An actor in process documentation is something that has acted on its own or another's behalf, so corresponds to an entity in an engagement chain. An interaction between actors in process documentation is an event in which control, and usually data, is passed between actors, so corresponds to the creation of a direct engagement, or the conclusion of an engagement, where the server agent ceases to be engaged. For an engagement to be created, or for an agent to be freed from an engagement, we assume that some information must pass between agents. Documentation of this information maps to an interaction p-assertion and, on creation of an engagement, includes some representation of the goal for which the agent is engaged. The state of an actor at the time it engages another includes its goals, motivations, capabilities and other attributes. Process documentation documents the past, while engagement chains are models of the current state of a system, but records of past engagement chains can be seen in process documentation. In fact, the set of process documentation about a system can be seen as documenting all the engagement chains that were present in the system. An individual engagement chain is thus apparent in a sequence of connected relationship p-assertions, connecting the actions that fulfil a goal back to the autonomous agent that initiated the engagement.

Relationship p-assertions are not apparent in the engagement chain model, but there are two kinds of causal relationship that can be recorded about engagement chains:

- The relationship of a direct engagement to the goal that the engaging (client) agent was trying to fulfil.

- The relationship of a direct engagement to a preceding engagement that caused it to be made, i.e. the delegation of a goal or sub-goal.

The first is a relationship between an interaction (the engagement of one agent by another) and the state of an agent (the goal of an agent that the engagement aims to fulfil). Goals are represented in the model by actor state p-assertions. The second relationship is between two interactions, where one led to the other.

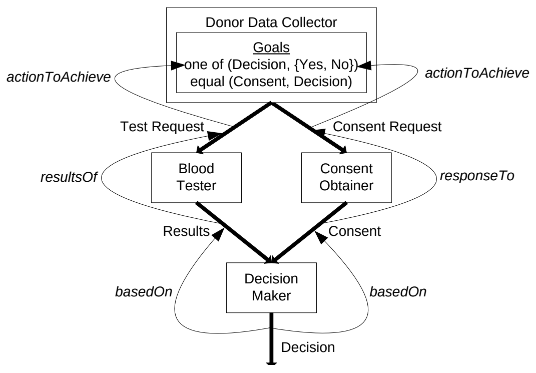

Figure 3: Goals and reason-specifying relationships in the example scenario process documentation.

By making explicit the relationships between actions of agents and the goals that were being fulfilled by performing them, we complete the graph of relationships, allowing us to connect an effect with the reason for it. This is depicted in Figure 3 where, in addition to the p-assertions shown in Figure 1, the donor data collector's goals have been made explicit as actor state p-assertions, and the test and consent request messages have been related to the goals that were a reason for them being sent. Every relationship p-assertion is marked with its specific type. Two relationships are of the first kind listed above, relating an engagement to the goal it is intended to fulfil, and are of type 'actionToAchieve'. The other relationships are of the second kind above, relating interactions, and are of varying, application-specific types: the test Results are the 'resultsOf' the Test Request, the family Consent is the 'responseTo' the Consent Request, and the final Decision on organ donation is 'basedOn' the Results and Consent.

Because they are all directly or indirectly connected via relationhips to an initiating goal, the interactions can be seen as *engagements* of one agent by another to achieve that goal. Where the engagement of an agent to achieve one goal causes the engagement of another agent to achieve another goal, then the second can be said to be a *sub-goal* of the first (it is being attempted to partially achieve the first). Therefore, causal relationships in the integrated model can be seen as relating sub-goals to their parents.

Ideally, whenever an autonomous agent performs an action, it should assert the goal that caused the action and the 'actionToAchieve' relationship between goal and action. There may be constraints on this ideal in reality, due to the volume of data recorded or security issues, but these are application specific and so addressed in our methodology [17].

4. REPRESENTING GOALS IN PROCESS DOCUMENTATION

Given the mapping between provenance and engagement chains above, we now proceed to show how process documentation can be modified to include goals. The process documentation model has an XML format, and to query programmatically over the contents of a provenance store, the additions to the model should also have a well-defined XML format.

A goal is represented in XML by a `goal` element. This element contains a set of `statement` elements, each describing one property of the world that must be true for the goal to have been achieved. The goal is a conjunction of the statements contained. A `statement` element contains a single `predicate`, which is

Process Documentation Concept	Engagement Chain Concept
Actor	Entity
Interaction	Creation of direct engagement
	Conclusion of direct engagement
Interaction p-assertion	Content of above with representation of exchanged goal
Actor state p-assertion	Goals, capabilities, attributes, motivations
Process documentation	Record of world engagement chain
Provenance query results	Record of engagement chain

Table 1: Mapping of Process Documentation and Engagement Chain Concepts

```
<goal>
  <statement>
    <predicate> oneOf </predicate>
    <parameter> <name> variable </name> <value> Decision </value> </parameter>
    <parameter> <name> choices  </name> <value> Yes, No  </value> </parameter>
  </statement>
</goal>
```

Figure 4: An XML representation of a goal

```
<actorStatePAssertion>
    <localPAssertionID>...</localPAssertionID>
    <content>
            <goal>...</goal>
    </content>
</actorStatePAssertion
```

Figure 5: An XML snippet of an actor state p-assertion containing a goal

the type of check on world state expressed by the statement, and 0 or more `parameter` elements which make the statement specific. Every parameter has a `name`, denoting its role in the state described by the statement, and a `value`.

In the example in Figure 4, the goal contains a single statement with two parameters. It describes a state of the world where a decision, $Decision$, has been made about the suitability for donation of a patient's organs, and states that the acceptable values of that decision are Yes or No.

A sequence of goals can be recorded in actor state p-assertions, according to the model described in the previous section. An actor state p-assertion has a well-defined schema [8], in which the content of the actor's state, such as one or more goals, is embedded. A snippet in which a goal is inserted in the actor state p-assertion model is in Figure 5. This is the format apparent in a provenance store when using our approach to document the goals of agents.

By defining a generic, well-specified representation of goals in actor state p-assertions, we allow independent agents in an application to all assert their goals in this same format, and then allow queriers of the process documentation to determine the intentions behind agents' actions without having prior knowledge of those agents. This representation of goals is designed explicitly for goals as they occurred in the past, i.e. for use in determining provenance. This allows them to be a lot simpler and more general than would be required to represent arbitrary goal-based plans, which could include details such as priorities of goal achievement, conflicts between goals etc.

5. ALGORITHMS

In this section, we give algorithms to answer the questions posed in the introduction regarding the responsibility for, the reason for, the success of and desirability of an effect in an application.

5.1 Reason and Responsibility

```
• To find those agents ultimately responsible for result $X$ and their reasons for causing $X$:
• For each relationship p-assertion $R$ of which $X$ is the effect:
    – For each cause $C$ in $R$:
        ∗ If $C$ is the goal of an autonomous agent, $A$, then $A$ is one of those ultimately responsible for $X$ and $C$ is $A$'s reason (or one of them) for causing $X$.
        ∗ Otherwise, all those agents found ultimately responsible for $C$ by this algorithm are also ultimately responsible for $X$ and their reasons for $C$ are also reasons for $X$.
```

Figure 6: An algorithm to determine those agents ultimately responsible for causing an outcome and their reasons

In particular, we can ask, with regard to the result of a process, who was *ultimately responsible* for a result and what was the *reason* they initiated the process that led to the result. Here, we consider responsibility to be held only by agents that had a choice not to perform the actions they did, i.e. those that are autonomous. By following relationship p-assertions backwards in time, we can establish both of these properties.

In Figure 6 we show the algorithm for determining those agents ultimately responsible for a result, and the reasons that they initiated the process that led to it, i.e. the goals that caused the agents to engage other agents to produce the result.

5.2 Success and Desirability

Once the reasons for the result of process are found, additional questions can be asked about whether the result truly reflected what those initiating the process wanted. If not, this could suggest that there is a problem in the application processes that should be rectified.

We consider a result, X, to be *successful* with regard to one of the reasons for it (goals that ultimately caused it), G, if X implies G. That is, if X at least fulfils the goal G, then X is a successful outcome of the process. We consider a result, X, to be *desirable* for an agent, A, ultimately responsible for X, if none of the goals held by A at the time they initiated the process that led to X is contradicted by or inconsistent with X.

6. EXAMPLE

We simulated the healthcare process in a Java application, using the tools provided by the PASOA project. Each agent recorded p-assertions into a Web Service provenance store (available from `www.pasoa.org`). The set of p-assertions for a simulation run is

```
 #1 Goal:
      oneOf (
        variable = Decision,
        choices  = {Yes, No}
      )
 #2 Goal:
      equal (
        first  = Consent,
        second = Decision
      )
 #3 Interaction:
      actors:  donorDataCollector -> bloodTester
      content: testRequest
 #4 Relationship:
      #3 actionToAchieve #1
 #5 Interaction:
      actors:  donorDataCollector -> consentObtainer
      content: consentRequest
 #6 Relationship:
      #5 actionToAchieve #1
 #7 Interaction:
      actors:  bloodTester -> decisionMaker
      content: testResults
 #8 Relationship:
      #7 resultsOf #3
 #9 Interaction:
      actors:  consentObtainer -> decisionMaker
      content: consent
#10 Relationship:
       #9 responseTo #5
#11 Interaction:
     actors:  decisionMaker -> doctor
     content: decision
#12 Relationship:
     #11 basedOn #7, #9
```

Figure 7: P-assertions produced by simulation

shown in Figure 7. For brevity, we use a more succinct and human-readable format than the original XML, defined as follows.

- For an interaction p-assertion, such as #3, this format appears on three lines. The first gives a number (local identifier) to the p-assertion and identifies it as an interaction p-assertion, the second states which actors sent and received the message in the interaction, and the third states the (abbreviated) content of the message.
- For relationship p-assertions, such as #4, this format appears on two lines. The first gives the numerical identifier of the p-assertion and identifies it as a relationship p-assertion. The second states the effect, type of relationship and (one or more) causes of the relationship. The effect and causes are references to other p-assertions.
- For a goal actor state p-assertion, such as #1, this format appears on multiple lines. The first gives the numeric identifier of the p-assertion and identifies it as a goal actor state p-assertion. The second gives the predicate. Remaining lines specify parameters.

Applying the algorithm in Figure 6 to the hospital example, we follow the relationship p-assertions (as depicted in Figures 1 and 7) back from the final decision on donation to all the data, events and goals causing that decision. This is summarised in Figure 8, in which each line represents an effect in the application and is followed by the cause or causes of that effect. We find that the actor ultimately responsible for the decision on donation was the donor data collector, which initiated the process, and the reason for doing so was the collector's goal to get a yes/no answer on the patient's suitability for donation. The intermediate steps of the process, testing blood and obtaining consent, were merely reactions to

```
#11 decision basedOn
 - #7 testResults resultsOf
 - - #3 testRequest actionToAchieve
 - - - #1 oneOf (variable=Decision, choices={Yes,No})
 - #9 consent responseTo
 - - #5 consentRequest actionToAchieve
 - - - #1 oneOf (variable=Decision, choices={Yes,No})
```

Figure 8: Provenance of decision produced by querying process documentation

the collector's initial engagement of actors to perform those tasks. If modelling were extended to include the patient's family as an actor, we might also determine that they had responsibility for the decision and their desires would be additional reasons for the final decision.

To determine the success of the outcome, we can see that the donor data collector initiates the process that leads to a donation decision because of the goal to get a yes/no answer on the patient's suitability. If the final decision is 'yes' or 'no', then the goal is clearly met. If, however, the decision is undecided or inconclusive then the goal has not been met. The latter would suggest that either the decision making process is inadequate or that the goal is unrealistic.

Finally, with regard to the desirability of the outcome, the donor data collector holds the goal that the family's wishes are always respected (the $Decision$ and the $Consent$ are equal as seen in the process documentation). If the decision on donation contradicts the family's decision on consent, the outcome is not desirable. Again, this suggests that the decision making process needs to be repaired.

Given that different agents may initiate organ donation assessments then, without process documentation, there would be no way of connecting the result of a process to the agents that originated it: the decision maker may be unaware of the donor data collector that initiated the process. In addition, without including goals in our model to represent the intentions of agents, the success and desirability of a process in terms of the agents initiating it could not be judged. With the approach presented in this paper, we have been able to ask questions about an application process which can help identify inconsistencies between the intended and actual effects of those processes.

7. CONCLUSIONS

As systems become ever more complex in support of grand visions of computing, with large-scale distributed systems being applied to solve challenging and important problems, the notion of provenance, to track and trace data and processes is becoming increasingly more important. At one level, significant results have already been achieved, and good progress is being made. However, capturing intentional notions in such frameworks has, until now, not been considered. Yet if provenance is to be useful in tracing more than just *what* was done, and consider *why* it was done, then this is vital. In this paper we have developed the first model for addressing just this concern, by mapping existing models of intentional interactive behaviour on the one hand, and provenance on the other, to derive a new model for representing autonomy in provenance.

While the example presented in this paper is small, most real applications include complex processes that span multiple components and, in an increasing number of cases, multiple sites and organisations. The provenance architecture, its extension to include representation of autonomy and the algorithm presented in this paper all apply equally well to small or large processes. This facility comes from the *localisation* of the recording of p-assertions, whereby actors record what is occurring locally to themselves only, and the iterative following of relationships, which means that only one p-assertion needs to be evaluated at once and the execution of the algorithm can be distributed.

The value of this initial model is clear, but there is still more work that remains to be done. For instance, in this paper we have only considered processes initiated by one agent, rather than by multiple co-operating agents. Furthermore, we envisage that, if a system includes multiple autonomous agents that document their cooperation and the engagement chains they initiate, and if that documentation includes the goals that they are attempting to fulfil by these processes, then another agent, A, can later come along and use the described processes as a *plan* to fulfil its own goals. Because the goals of the agents involved in past processes are made explicit, A can determine whether the plan will exactly suit its requirements or needs to be adapted. From a software-engineering perspective, the methodology by which developers determine what process documentation should be recorded by their application could be integrated with existing agent-oriented methodologies to ease the development of provenance-aware multi-agent system applications. Finally, in systems in which the autonomous agents of interest are people, rather than software, we cannot expect p-assertions regarding their goals to be recorded directly. Instead, we can, in some circumstances, *infer* their goals from their interaction with software that does record p-assertions.

In this paper, we have extended a concrete system, which is being deployed in multiple domains, for capturing the provenance of data, so that the intentions behind the actions of application actors can be generically documented in a well-specified format. This allows agents to record their actions, interactions and intentions without needing to know of each other's existence, an important characteristic in large, dynamic systems. Because of the common data structure and semantics of that recorded, a querier can then combine the disparate documentation to answer important questions regarding whether the application is fulfilling the goals of those using it, the algorithm presented in this paper being an example of such a query. Future work will expand on this basis to consider asking questions regarding the provenance of data in systems in which agents cooperate and compete.

8. REFERENCES

[1] S. Barber and C. Martin. Agent autonomy: Specification, measurement, and dynamic adjustment. In *Proceedings of the Autonomy Control Software Workshop, Autonomous Agents 1999 (Agents99)*, pages 8–15, Seattle WA, 1999.

[2] R. Bose and J. Frew. Lineage retrieval for scientific data processing: A survey. *ACM Computing Surveys*, 37(1):1–28, March 2005.

[3] S. Brainov and H. Hexmoor. Quantifying relative autonomy in multi-agent interaction. In *Proceedings of the IJCAI01 Workshop Autonomy, Delegation, and Control: Interacting with Autonomous Agents*, pages 27–35, Seattle, 2001.

[4] P. Buneman, S. Khanna, and W.C. Tan. Why and where: A characterization of data provenance. In *Int. Conf. on Databases Theory (ICDT)*, pages 316–330, 2001.

[5] A. Cooke, A. Gray, W. Nutt, J. Magowan, M. Oevers, P. Taylor, R. Cordenonsi, R. Byrom, L. Cornwall, A. Djaoui, L. Field, S. Fisher, S. Hicks, J. Leake, R. Middleton, A. Wilson, X. Zhu, N. Podhorszki, B. Coghlan, S. Kenny, D. O'Callaghan, and J. Ryan. The relational grid monitoring architecture: Mediating information about the grid. *Journal of Grid Computing*, 2(4):323–339, December 2004.

[6] I. Foster, N. R. Jennings, and C. Kesselman. Brain meets brawn: Why grid and agents need each other. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multi-Agent Systems*, pages 8–15, New York, USA, 2004.

[7] I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.

[8] P. Groth, S. Jiang, S. Miles, S. Munroe, V. Tan, S. Tsasakou, and L. Moreau. An architecture for provenance systems. Technical report, Electronics and Computer Science, University of Southampton, October 2006. Available at http://eprints.ecs.soton.ac.uk/12023/.

[9] T. Kifor, L. Z. Varga, J. Vazquez-Salceda, S. Alvarez, S. Willmott, S. Miles, and L. Moreau. Provenance in agent-mediated healthcare systems. *IEEE Intelligent Systems*, November/December:To appear, 2006.

[10] M. Luck and M. d'Inverno. A formal framework for agency and autonomy. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 254–260. AAAI Press / MIT Press, 1995.

[11] M. Luck and M. d'Inverno. Engagement and cooperation in motivated agent modelling. In C. Zhang and D.Lukose, editors, *Distributed Artificial Intelligence Architecture and Modelling: Proceedings of the First Australian Workshop on Distributed Artificial Intelligence*, volume 1087 of *Lecture Notes in Computer Science*, pages 70–84. Springer-Verlag, 1996.

[12] M. Luck, S. Munroe, and M. d'Inverno. Autonomy: Variable and generative. In H. Hexmoor, C. Castelfranchi, and R. Falcone, editors, *Agent Autonomy*, pages 9–22. Kluwer, 2003.

[13] Nancy Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[14] S. Miles, P. Groth, M. Branco, and L. Moreau. The requirements of using provenance in e-science experiments. *Journal of Grid Computing*, 2006. To appear.

[15] L. Moreau, P. Groth, S. Miles, J. Vazquez, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan, and L. Varga. The provenance of electronic data. *Communications of the ACM*, page 10 pages, 2006.

[16] S. Munroe and M. Luck. Agent autonomy through the 3M motivational taxonomy. In *Agents and Computational Autonomy: Potential, Risks, and Solutions*. Lecture Notes in Artificial Intelligence 2969, Springer, 2005.

[17] S. Munroe, S. Miles, L. Moreau, and J. Valquez-Salceda. Prime: A software engineering methodology for developing provenance-aware applications. In *Proceedings of the Software Engineering and Middleware Workshop (SEM 2006)*. ACM Digital, 2006. To appear.

[18] M. Raynal and M. Singhal. Logical time: Capturing causality in distributed systems. *Computer*, 29(2):49–56, February 1996.

[19] Y. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36, 2005.

[20] D. M. Taylor. *Explanation and Meaning: An Introduction to Philosophy*. Cambridge University Press, 1970.

[21] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.