Determining the Trustworthiness of New Electronic Contracts

Paul Groth[1], Simon Miles[2], Sanjay Modgil[2], Nir Oren[2], Michael Luck[2], Yolanda Gil[1]

[1] Information Sciences Institute, University of Southern California, USA
[2] Department of Computer Science, King's College London, UK

Abstract. Expressing contractual agreements electronically potentially allows agents to automatically perform functions surrounding contract use: establishment, fulfilment, renegotiation etc. For such automation to be used for real business concerns, there needs to be a high level of trust in the agent-based system. While there has been much research on simulating trust between agents, there are areas where such trust is harder to establish. In particular, contract proposals may come from parties that an agent has had no prior interaction with and, in competitive business-to-business environments, little reputation information may be available. In human practice, trust in a proposed contract is determined in part from the *content* of the proposal itself, and the similarity of the content to that of prior contracts, executed to varying degrees of success. In this paper, we argue that such analysis is also appropriate in automated systems, and to provide it we need systems to record salient details of prior contract use and algorithms for assessing proposals on their content. We use *provenance* technology to provide the former and detail algorithms for measuring contract success and similarity for the latter, applying them to an aerospace case study.

1 Introduction

Contracts are a mechanism by which multiple parties codify agreements to undertake tasks, deliver services, or provide products. Before signing a contract, it is generally true that each party expects that the others will be able to fulfil their part of the agreement. This *trust* in a party can be obtained from a variety of sources including prior dealings with the party, recommendations from associates, and the perceived ability to effectively enforce the compliance of the party. For example, when signing a mobile phone contract, the phone provider performs a credit check on the potential customer, to verify that the customer will be able to pay. Likewise, the customer may check independent internet sites to determine whether the service provided is high quality and fairly priced. Thus, the parties trust each other to perform the actions stated in the contract.

However, another trust judgment must be made in addition: one must also trust the contract *itself* [8], to believe that, for example, it correctly specifies terms of service, appropriately deals with exceptional cases, is sufficiently stringent that actions will be taken if obligations are not fulfilled. For a phone contract, the customer might look for a statement on when support will be available, whether there is an indemnification portion of the contract, or even whether the contract is presented on paper with an attractive letterhead (indicating professionalism by the provider).

As open electronic contracting becomes prevalent, we expect issues centred around *content-based trust* (trust of artifacts based on the artifacts themselves) to arise more frequently. Clients must determine whether to place trust in a contract provided by a service even while, in many cases, such services may be unknown, or the service may introduce a new contract specific to the client.

In these cases, reputation-based mechanisms alone do not provide suitable means of measuring trust: if there are little or no grounds on which to make reputation decisions, then assessments of trust based on reputation are of limited value. Mechanism design techniques, attempting to guarantee that agents will behave in a certain way [11], are ill-suited to open systems, where new agents, which may not be analysed by the system designer and may not always act rationally, can enter the system at any time.

Contracts presented by one agent for a service may well have similarities to those proposed by other agents providing similar services, but are unlikely to be of the same form or *template*: each company (or organisation, or agent) drafts contracts in its own way. However, even without exactly matching contracts, an agent may still draw on prior experience, and judge a newly proposed contract on the basis of similarities. For example, if similar contracts to the current one turned out to be valuable or successful in the past, this may indicate future value or success. This approach requires two contract-specific mechanisms to be in place, to determine each of the following.

- The *success* of prior contracts the agent has been party to. Success is meaningful only if we have an unambiguous model of contractual behaviour, and if it is based on the events that occur during a contract's lifetime. For us to draw on salient events, we need to capture data on those events in a queryable form.
- The *similarity* of a prior contract to a newly proposed one. As above, similarity is meaningful only if we understand what a contract contains: syntactic similarity alone would not provide a meaningful measure, e.g. the obligation to do something and the obligation *not* to the same may be expressed using almost the same data are not similar contractually.

In this paper, we begin to address the problem of how a client can make a trust judgment about a contract based on the contract itself. We do so by building on several pieces of existing, though largely unconnected work on *contract modelling*, *content-based trust* and *provenance*. A theoretical framework for modelling contracts, and its use for representation as concrete data, is presented. As with human assessments, we enable clients to access their own prior experiences, as well as records available from others, in order to help make a trust judgement. This prior experience is expressed in the form of a *provenance graph*, which documents how contracts were constructed, where they were used, and the processes that they were related to. While the overall work is preliminary, we have provided a proof of concept implementation and evaluation. The technical contributions provided in this paper are:

- an approach to measuring the success of prior contract executions;
- an algorithm for establishing the similarity between contracts;
- a model for determining the trustworthiness of contract proposals based on the above notions of success and similarity;

- an implementation of all of the above using an existing electronic contract representation scheme and provenance model;
- a case study and evaluation of our approach in the aerospace domain.

1.1 Case Study

In this paper, we use an aerospace application as our running example and case study. We are grateful to Lost Wax for the particulars, which are part of a simulation performed by the Aerogility tool for their (aircraft engine manufacturer) customers [1].

Within the domain under consideration, aircraft operators require the manufacturers of their aircraft engines to maintain the engines during their lifetimes. Engine manufacturers in turn rely on services at particular sites to conduct any repairs. A plane with an engine requiring maintenance will be scheduled to land at a site with which the engine's manufacturer has a contract, so that repairs can take place. A service site makes use of suppliers for given parts, and engine manufacturers often wish to restrict the suppliers they use to those trusted.

As our running example, we consider proposals from new service sites to engine manufacturers to provide, and be paid for, repair of engines. The question asked by the engine manufacturer is: Should I trust this new proposal? Engine manufacturers have more or less prior experience with other sites, and may or may not have access to records about other manufacturers' experience.

2 Trust

Trusting someone or something allows the uncertainty of the world to be handled: even where aspects are unknown and may be unknowable, we do not allow this to preclude us from action. Gambetta [7] defined trust as "a particular level of the subjective probability with which an agent will perform a particular action, both before she can monitor such an action, and in a context in which it affects her own action". We apply this definition in contract-based systems as the likelihood that a contract proposal will lead to other agents (primarily other contract parties) acting in an expected and desirable way.

To understand what we should capture in a measure of trust for a contract proposal, we need to know why an agent may agree to one, including the following reasons.

- The agent believes the outcome of successful execution of the contract will be in their interests.
- The agent believes the other parties will act in accordance with the contract.
- The agent believes another contract could not easily be obtained which better represented their interests.

Castelfranchi [5] gives a more exhaustive list of reasons why an agent may attempt to meet a norm in the general case.

Content-based trust, the trust in artifacts from the artifacts themselves, is a relatively neglected area compared with mechanisms to assess trust in other agents (reviewed recently, for example, by Sabater and Sierra [19], Ruohomaa and Kutvonen [18]). Determining content-based trust was identified as an open issue by Gil and Artz in the context of the Semantic Web [8]. Content-based trust is crucially important, given that the web can be understood as a network of documents whose content people judge trustworthy or otherwise based on various factors, including context, popularity, appearance, provenance, apparent bias, etc. Content-based trust complements assessments on the trust of those providing the content made, for example, through reputation metrics.

Such evaluations also apply where actors in a process are not human but software agents, and it is therefore important to determine such assessments of trust automatically. Aside from general web documents, content-based trust is important in assessing contract proposals in an open system, where evaluations of those issuing proposals may not be sufficient for providing a basis to decide whether to accept. Gil and Artz envisaged users providing feedback on content, which would inform the trust assessments of later users. We take an equivalent approach here, where that feedback data is captured by records of success or otherwise of past contracts.

3 Electronic Contracts

To compare similarity of contracts, so we can judge how much of an influence a past contract's success should have on assessing a new proposal, there needs to be a basis for comparison. Unless contracts are expressed in a relatively uniform way, there is no way to know whether one part of the document, e.g. a clause, is comparable to another. While we cannot rely on contracts created by different agents being instantiated from the same template, we assume that all contracts can be expressed using the same conceptual structure, so can be mapped to the same data structure. We also assume that, within an application domain, contract terms can be mapped to a common ontology, allowing each part of a contract to become comparable.

For example, in our case study, we cannot assume all service sites provide contracts following the same template, as they are independent, but assume that they all rely on the same basic constructs (obligations, permissions etc.) and include some similar content, e.g. obligations to service engines when they require maintenance.

3.1 Basic Concepts

A contract is a document containing *clauses* agreed to by a set of agents, called the *contract parties*. Prior to agreement by the relevant parties, a contract document is simply a *contract proposal*. For an electronic document, agreement may be indicated in the form of digital signatures by the contract parties. Clauses specify *regulative* or *constitutive* norms influencing agent behaviour.

Constitutive norms do not form a mandatory part of a contract, but help with contract reusability and interpretation. According to Boella and van der Torre [4], the main role of a constitutive norm is to specify a *counts-as* relation between facts in the domain and terms in the contract. For example, a contract for a broadband connection may specify what counts-as fair use, in terms of observable facts such as the amount of data downloaded over the course of a month. Another important contractual notion, *role*, where an agent occupying the role takes on the regulative norms imposed on the role, may also be subsumed by the counts-as relationship. The counts-as relationship has been the subject of much analysis [12, 10].

Here, we focus on regulative norms. Regulative norms are associated with a *target* agent, and specify either what the target *should* do, an *obligation*, what the target *should not* do, a *prohibition*, or what the target *may* do, a *permission*. If the target of an obligation or a prohibition does not conform to the behaviour expected by that clause, then this is a *violation* of the clause. If a clause has not been and can no longer be violated (for example, if the action required in an obligation has been performed), then we say that it has been *fulfilled*.

Some obligations or permissions may specify behaviour required or allowed when another clause is violated: *contrary-to-duty* clauses, e.g. one obligation may state "A must deliver the package to B by Thursday" while a contrary-to-duty obligation says "If A has not delivered the package to B by Thursday, it must refund B's payment."

3.2 Electronic Representation of Contracts

Many different computational representations of contracts are possible, but common formats are required for interoperability and comparison. The CONTRACT project [2] addresses this concern (among others), by encoding contracts as XML documents consisting of normative clauses that are essentially declarative specifications of agent behaviours. The model is informed by a need to meet requirements for *practical* execution of contracts and monitoring for violation of contracts by the parties involved (some features of this model are not unique to this approach, but also present in others' models [6, 9]). Importantly, this execution-based motivation and XML-based representation provide us with both the stance and tools needed to develop techniques for evaluating contract similarity. We therefore adopt the CONTRACT model and representation in our work. Although we cannot provide a complete account here, we briefly outline several key qualities of the model, and leave details to other publications [16, 17].

1. A contract clause is often conditional, requiring or allowing an agent to undertake an action only in specific circumstances. For example, an obligation on a service site to repair an engine only matters whenever a request for maintenance has been placed. For this reason, a practical agent should not expend resources aiming to fulfil all clauses when most do not apply at any given time.
2. These 'activating' circumstances can exist in multiple instances, sometimes simultaneously. Continuing the example, multiple requests may be placed at one time, and a single contract clause requires each request be processed.
3. No contract party can predict whether it will, in practice, be possible for them or the other parties to fulfil obligations at the time the contract is agreed. Contracts are based on expected future capabilities of agents not on certainty of achievement. For a recent example, consider passengers left stranded due to the collapse of airline and holiday companies: the obligation to return them home was agreed but the capability did not exist at the relevant time.
4. All information relating to whether a contract clause is violated or not must originate from some source. For example, in a distributed system, information about time comes from *clock* services, and there can be no guarantee in practice that the time received from any two clocks will be the same.

5. That which is obliged or permitted can always be expressed as *maintenance* of some condition (within each instantiating circumstance). Clearly the obligation to "always drive on the left", for example, is already expressed as the maintenance of a condition. However, an obligation to achieve state S in the system before deadline condition D can be expressed as maintaining the condition: *either* achieved S *or* not yet passed D. For example, the obligation to "have repaired an engine within 7 days of the request" can be re-expressed as maintaining the state "have repaired the engine or not yet passed 7 days from the request". Generally, the obligation to perform action A before a deadline D can be re-expressed as maintaining the state of having performed A or D not yet having passed.

Taken together, these points inform the development of a model for contractual clauses using the data structure in Table 1. It should be noted that any logic may be used to specify the conditions in our contract clauses, and that it is possible to map the deontic aspect of a clause to standard conditional logics [21]. Examples of the model's use in the domain of aerospace aftermarkets are provided in Section 6.

Type	Whether this is an obligation or permission. A prohibition is modelled as an obligation not to do something, i.e. with a negative normative condition below.
Target	The contract party obliged, prohibited or permitted by the clause.
Activating Condition	The circumstances under which the clause has force, parametrised by the variables specific to each instance.
Normative Condition	The circumstances under which the obligation is not being violated or the permission is being taken advantage of, parametrised by the variables specific to each instance. Therefore, for an obligation, the target must *maintain* the normative condition so as not to be in violation of the contract.
Expiration Condition	The circumstances under which the clause no longer has force, parametrised by the variables specific to each instance.

Table 1. The model for a contractual clause used in this paper.

The CONTRACT project XML-based electronic format for contracts [17] is used in our implementation. Without providing excessive detail, the key components of a contract in this format can be outlined as follows.

- References to ontologies defining the concepts used in the contract and the pre- and post-conditions of actions obliged, prohibited or permitted.
- A set of contractual roles to which contract clauses apply.
- A set of agents assigned to those roles.
- A set of clauses as modelled above.

A contract, in itself, does not express success or failure in achieving the obligations expressed within it. For that we need to know, through recorded documentation, the outcomes of agents' attempts to fulfil the contract, and the processes by which they did so. This can be called the *provenance* of the contracts' outcomes.

4 Provenance

Judgements of trust must be based on reliable data. Even when the assessed object provides the primary data, its content must be compared against something. In our case, the proposal content is compared against prior contracts, to assess the proposal based on the similarity with, and success of, those contracts. The task of acquiring this historical data is outside the scope of many approaches and so not adequately addressed. However, in a system with multiple *authorities*, where contracts need to be established to guarantee one agent's behaviour towards another, gathering such data is not trivial.

Much work [20] has been conducted on techniques for determining the *provenance* of data: the source or history of that data [14]. There are two critical issues in providing infrastructure for determining provenance. First, applications must be adapted to record documentation of what occurs in a system in such a way that it can be connected to form historical traces of data. This requires the independent agents in the system to document their activities using a suitable common data model. Second, techniques are needed to query potentially large amounts of such documentation, to elicit details relevant to the history of a given item. Additional issues, such as the reliability of documentation for provenance and its secure storage must also been considered [14].

In this paper, we exploit the documentation produced by a provenance infrastructure to find connections between prior contracts and their outcomes. This allows us to measure the success or otherwise of prior contract executions, and the compliance with, or violation of, clauses by the contract parties. The model of provenance used to evaluate our approach is based on *causal graphs*, connecting occurrences in the system as *effects* to their *causes*, whether the occurrences involved one agent or the interaction of many. Here, we are not concerned with *how* the system was adapted to record documentation, but refer readers interested in the software engineering issues elsewhere [13].

We use the Open Provenance Model (OPM) [15], developed by an international collaboration of provenance researchers. OPM documents prior occurrences in terms of *processes* that manipulate *artifacts*. A process can be seen as an individual action by an agent, while an artifact corresponds to a message sent between agents or to an object acted on while in a given state. An example is shown in Figure 1, where a part supply process receives a request for a part and generates that part, passing it to a process that takes a broken engine and produces a repaired engine. Arrows denote causal dependencies, where a process *used* an artifact or an artifact *was generated by* a process.

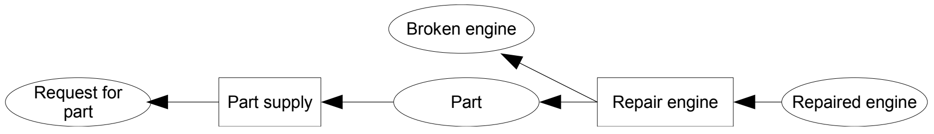

Fig. 1. Sample OPM graph fragment: artifacts are ovals, processes are boxes

OPM has serialisations in formats such as XML and RDF (though they are still being refined). We use the current XML serialisation in our experiments, a sample of which is shown in Figure 2.

5 Contract Trust Algorithm

We can view the problem of determining the trust of a previously unseen contract in different ways. As a *classification* problem, we aim to determine to which category of trust the contract belongs, e.g. we could classify a contract into 'trustworthy', 'somewhat trustworthy' or 'untrustworthy'. Or, as a *regression problem*, the goal is to compute a *trust value* for the given contract.

For either case, we can apply an approach based on the simple *k-nearest neighbour* algorithm [3]. Briefly, the algorithm can be described as follows. Given an instance, q (in our case a contract proposal) find the k instances (prior contracts) in a historical dataset that are closest to q. The instances then *vote* as to the category/value of q. In our evaluation, we use the k-nearest neighbour algorithm to perform regression that determines a numerical trust value. It is then for an agent to decide whether the trust value is adequate to accept the proposal, or to choose between multiple potential proposals. To use the k-nearest neighbour algorithm, we need to define the *features* or attributes to be used, a *distance* function, a *voting* function, and k. Below, we define each of these elements for our purposes.

5.1 Features: Measuring Success of Contracts

The historical dataset used by our algorithm is the set of prior contracts the agent has been party to. We are interested in one attribute of a contract: whether the contract was successful, defined as one in which the outcome of the process the contract governed is achieved without breaking any contractual clauses. For example, if a service site was obliged to repair an engine by a certain time according to a contract and the service does so, then the contract is successful. However, the success of a process' outcome is not a binary decision but lies in a range. For example, if the engine is fixed but the repair was delayed, the outcome of the repair process is still successful but less so than if the engine was repaired on time. Additionally, it is important to note that the outcome of a process is not merely its output; the outcome may also include effects such as repairs being late. To determine all these effects, our approach is to use documentation of a process. We write D_p^c to identify the provenance data for a process p that was the execution of a contract C.

Our algorithm relies on a success-outcome function, $s(D_p^C) = 0 \ldots 1$, that computes, given the provenance data for a process, a numeric value for the success of the process outcome. In practice, this is a series of tests performed on the provenance data to see if particular effects were present in the documentation; for example, whether penalties were paid when a clause was violated. Thus, the success-outcome function provides the value for the success attribute of each contract in the historical dataset. This is a domain-specific and often user-specific function.

The measure of success of a past contract is calculated by the agent assessing the new proposal, and so the values given to that success are otherwise arbitrary reflections of what that agent considers important. If two agents use similar calculations of success from the events recorded in the documentation, they have similar outlook on trust. Section 6.3 describes the function we use for the aerospace case study.

5.2 Distance: Measuring Similarity of Contracts

We now define a distance function to determine how close a proposal is to the prior contracts. A contract C_1's distance from another contract C_2 is a normalised similarity value. We rely on a general function, Equation 1, to produce all permutations of pairings of elements of any sets A and B:

$$map(A, B) := \Big\{ P \subset A \times B \Big| \text{all elements in a pair occur only once in } P \Big\}. \tag{1}$$

$map(A, B)$ is used by Equation 2 to compute the summation that maximizes the additive similarity score between two sets. Equation 2 assumes that there exists a similarity function for the elements within the provided sets. This equation is necessary because clauses and other constructs do not have a fixed comparison order.

$$max\_additive\_score(A, B) = \max\left(\{ \sum_{(x,y)\in P} sim(x, y) \big| P \in map(A, B) \} \right) \tag{2}$$

Based on Equations 1 and 2, we define the distance function, Equation 3, as the maximum similarity score between two contracts, C_1 and C_2. The function assumes that a contract, C, is a set of clauses, hence, we can write $cl \in C$.

$$dist(C_1, C_2) = max\_additive\_score(C_1, C_2); \tag{3}$$

The similarity between two clauses, Equation 4, is defined as the similarity between the clauses conditions. We use a dot accessor notation to denote accessing each type of condition from a clause. Thus, $cl.act$ denotes accessing the activation condition from a clause. Likewise, $cl.mant$ denotes accessing the maintenance condition (the expiration condition is excluded, as it merely aids agents in knowing when clauses no longer apply, rather than affecting the semantics of what is required/permitted). Similarily, each condition contains a formula, F, that can be accessed in the same fashion. Additionally, we use the function $type()$ to allow for the retrieval of the type of a given object. In the case of the deontic statement this is whether it is an obligation or permission. In the case of other entities such as variables this is the class of that entity.

$$sim(cl_1, cl_2) \quad := \quad \frac{sim(cl_1.act, cl_2.act) + sim(cl_1.mant, cl_2.mant)}{2} \tag{4}$$

The similarity of two conditions is the similarity between the two conditions formulas as seen in Equations 5, 6, 7.

$$sim(exp_1, exp_2) := \begin{cases} 0 & \text{if } type(exp_1)! = type(exp_2) \\ sim(exp_1.F, exp_2.F) & \text{otherwise} \end{cases} \tag{5}$$

$$sim(act_1, act_2) := sim(act_1.F, act_2.F) \tag{6}$$

$$sim(mant_1, mant_2) := sim(mant_1.F, mant_2.F) \tag{7}$$

For convenience, we assume all formulas are in disjunctive normal form, i.e. each formula is a set of disjunctive clauses, DIS, and each of these is a set of conjunctive clauses, CNJ, in turn a set of atoms. The equations for determining formula similarity are given below. At each step, the combination that maximizes the similarity are chosen.

$$sim(F_1, F_2) := max\_additive\_score(F_1, F_2) \tag{8}$$

$$sim(DIS_1, DIS_2) := max\_additive\_score(DIS_1, DIS_2) \tag{9}$$

$$sim(CNJ_1, CNJ_2) := max\_additive\_score(CNJ_1, CNJ_2) \tag{10}$$

Similarity between two atoms is defined by Equation 11. Each atom consists of a relation, $a.r$, a list of variables, $a.vars$, and a list of individuals, $a.inds$. The Equations 13, 14, and 15 define similarity for each component respectively.

Note $d(x, y)$ denotes a *domain specific function* for comparison, returning a similarity score from 0 and 1. In the case study below, we used a function for time periods that states if two periods are within 5 calendar days then they are equivalent. Similarly, for two non-equal payments, if the difference is less than 2000 euros, we state a 0.75 similarity score, or 0.5 if between 2000 and 6000 difference.

$$sim(a_1, a_2) := \begin{cases} 0 & \text{if } |a_1.vars|! = |a_2.vars| \vee |a_1.inds|! = |a_2.inds| \\ sim`(a_1, a_2) & \text{otherwise.} \end{cases} \tag{11}$$

$$sim'(a_1, a_2) := \frac{\Big(sim(a_1.r, a_2.r) + \sum_{i=0}^{|a_1.vars|} sim(a_1.vars[i], a_2.vars[i]) + \sum_{i=0}^{|a_1.inds|} sim(a_1.inds[i], a_2.inds[i])\Big)}{1 + |a_1.vars| + |a_1.inds|} \tag{12}$$

$$sim(r_1, r_2) := \begin{cases} 1 & \text{if } r_1 = r_2, \\ d(r_1, r_2) & \text{if } \exists d(r_1, r_2), \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

$$sim(v_1, v_2) := \begin{cases} 1 & \text{if } type(v_1) = type(v_2), \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

$$sim(ind_1, ind_2) := \begin{cases} 1 & \text{if } ind_1 = ind_2, \\ d(ind_1, ind_2) & \text{if } \exists d(ind_1, ind_2), \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

5.3 Voting: Evaluating Trustworthiness of Proposals

Using this distance function, *k-nearest neighbour* can determine which contracts are closest to the input contract. To assign a success value to the new contract, a voting function is required. We adopt a simple normalised weighted average approach. Here, the input or proposed contract, c_p, is assigned a success value that is determined as follows: first, the sum of the success values from the k-nearest contracts is calculated, where each success value is weighted by its distance from the input contract. Then, this sum is divided by k. Because we are using this weighted approach, we can use a very high value for k because the weighting will discount any contracts that are far away from the input contract. In our tests, k is set to be equal to the entire historical dataset. Thus, Equation 16 describes the voting function, n is equal to the number of contracts in the historical dataset.

$$v(c_p) = \frac{\sum_{i=0}^{i=n} dist(c_p, c_i) s(D_p^{c_i})}{n} \tag{16}$$

k-nearest neighbour has a number of properties that make it useful for this application. In particular, it allows the agent to apply all its historical knowledge to determining the trust value of a new contract. It also removes the need for a training phase in the algorithm so that new information can be used immediately. Finally, the approach can be easily adapted to support categories of trust instead of a trust value.

6 Case Study

In our preliminary case study, we simulate the aerospace scenario through messages being sent by the various entities, indicating what they have done and what has happened. This allows compliance or violation to be determined. Now, in our particular example, we assume that an engine manufacturer is offered a contract proposal by a service site of which it has no prior experience. The manufacturer judges the contract proposal on the basis of its prior contracts with other sites. Each contract consists of the following relevant clauses.

1. An obligation on the service site to repair each engine within D days of it arriving for maintenance.
2. An obligation on the service site to pay a penalty P to the manufacturer for each repair not completed in D days (with the payment having its own deadline, not varied in this case study).
3. A set of permissions and prohibitions, one for each of a set of part suppliers, allowing or denying the service site to source parts from that supplier. We assume three relevant part suppliers exist.

The formalisation of the first of the above is characterised in Table 2, following the data structure shown in Table 1.

Type	Obligation
Target	Service Site
Activating Condition	Engine E requires repairing at time T
Normative Condition	Engine E has been repaired or time $T + D$ has not been reached
Expiration Condition	Engine E has been repaired or time $T + D$ has been reached

Table 2. Model for obligation 1 in the case study contract

6.1 Simulation

Simulations of the execution of the above contract were run, varying the following simulation factors.

- The time to repair, D, may be short or long.
- The penalty payment, P, may be high or low.
- For each part supplier, the site may be permitted or prohibited from sourcing parts from that supplier.
- The service site may be honest or dishonest.

These factors combine to influence which of four scenarios is executed in the simulation. The scenarios have the following outcomes.

1. The repairs are completed successfully using permitted part suppliers.
2. The repairs are not all completed successfully, but a penalty payment is received.
3. The repairs are not all completed successfully, and no penalty payment is received.
4. The repairs are completed but using a prohibited part supplier.

Long time to repair, high penalty payment and more permitted suppliers increase the chance of repairs being completed. Long time to repair and more permitted suppliers also increase the chance of correct part suppliers being used. Honesty increases the chances of penalties being paid when repairs are not performed.

Specifically, the following calculation is used. First, we calculate the *repair success chance* as a base probability increased if D is long, P is high, and for each permitted supplier. *Allowed supplier chance* is increased by D being long and for each permitted supplier. *Pays penalty chance* is 0.1 or 0.8 depending on whether the site is honest. Repair success chance is the probability that repair is successful. If yes, allowed supplier chance is the probability that scenario 1 occurs, else 3. If no, pays penalty chance is the probability that scenario 4 occurs, else 2.

6.2 Provenance

Each of the four scenarios above produces a different provenance graph. Each OPM graph documents the processes that occurred during the scenario, and the data produced and exchanged in that scenario, with the documentation being recorded by some or all of the agents involved. Included in that graph are the messages sent between agents, which means that the engine manufacturer can identify the critical messages sent between participants indicating success of and compliance with the contract.

– A message received by the engine manufacturer notifying of successful repair, and containing a timestamp less than the deadline, indicates that repairs were completed successfully. The absence of such a message, or a later timestamp than the deadline, indicates lack of successful repair.
– Where there was not a successful repair, a message received by the engine manufacturer notifying of payment of penalty indicates that such a payment was made. Absence of the message indicates that the payment was not made.
– A message received by the service site from a part supplier notifying of delivery of a part indicates that that supplier was used by the service site.

A snippet of the XML serialisation in the OPM for a scenario is shown in Figure 2.

```
<opmGraph xmlns="http://openprovenance.org/model/v1.01.a">
  ...
  <processes>
    <process id = "SiteServiceScenario1Process">
      <value xsi:type="xs:string" ...>ServiceSite</value>
    </process>
  ...
  <artifacts>
    <artifact id = "SiteServiceScenario1OrderPart1">
      <value xsi:type="xs:string" ...>orderPart(order1,type112)</value>
    </artifact>
  ...
  <causalDependencies>
    <used>
      <effect id    = "SiteServiceScenario1Process"/>
      <role   value = "requestReceived"/>
      <cause  id    = "EngineManufacturerScenario1Request1"/>
    </used>
    <wasGeneratedBy>
      <effect id    = "SiteServiceScenario1OrderPart1"/>
      <role   value = "requestSent"/>
      <cause  id    = "SiteServiceScenario1Process"/>
    </wasGeneratedBy>
```

Fig. 2. Snippets of documentation recorded from scenario 1 following the OPM XML schema

6.3 Success and Similarity

The success of the contract execution should depend on the outcome and reflect the importance given to different factors by the agent making the trust assessment. For the scenarios above, scenario 1 should be rated as 1.0 success; scenario 2 has 0.75 success; scenario 3 has 0.25 success; and scenario 4 has 0.0 success. Scenario 4 has the worst outcome because using a faulty part from a bad supplier is more likely to cause the plane to fail than merely not receiving maintenance. A prior contract is, then, more similar to the proposal if it has the same time to repair, same penalty payment, and same permissions and prohibitions on part suppliers.

7 Evaluation

In this section, we describe our evaluation of the efficacy of our algorithm, using the aerospace case study introduced above. In one test case, N prior contracts are generated with a random set of factors. For each contract, a scenario is selected randomly but influenced by the factors as described in Section 6.1, e.g. long time to repair increases the chance that a scenario in which repairs are completed will be chosen. The scenarios are enacted and the execution documented as provenance graphs.

A contract proposal is randomly generated just as was done for the prior contracts. The engine manufacturer uses the documentation to judge the success of each prior contract, and the similarity algorithm to judge the similarity of the content of the new proposal to each prior contract. They combine these measures, as specified in the trust algorithm above, to judge whether to trust the proposal.

We evaluate how more information on prior contracts has a beneficial effect on decisions to trust proposals. For each randomly generated proposal, we first enact it multiple times, getting an average score for its success: called the *average actual success*. We compare this with trust rating generated by our algorithm.

In our experiment, we generated 10 contract proposals. For each proposal, we computed the average actual success by enacting the proposal 100 times. We then compared this to the trust rating computed using our algorithm as we increased the number of prior contracts from 1 to 200. We measured the difference between these values after adding increments of 10 prior contracts. Figure 3 shows the average difference when combining the measures for all 10 proposals. As can be seen in the graph, the difference between these two scores steadily decreases as the number of prior contracts increases. This shows that the algorithm is effective: by calculating trustworthiness of a proposal using our algorithm, based on prior contract success and similarity, we achieve a closer and closer match to the actual success rate which would be achieved from enacting that contract. Therefore, the trust measure accurately reflects the worth to be gained by the agent by accepting the proposal.

8 Conclusions

Assessing a contract proposal can be difficult when there is little public information on the proposer. In this paper, we show how a content-based assessment, comparing the proposal with prior contracts based on its similarity of content to them, and the success of those prior contracts, can inform a decision on whether to trust the proposal. As shown in our results, as the number of prior contracts used in the assessment increases, the difference between the trust value and the actual outcome decreases; that is, the evaluation becomes more accurate.

Our approach relies on two pieces of technology: a contract model and its data representation on the one hand, and a model for representing the history of processes within the system (the provenance of contract enactments' outcomes) on the other. Our work effectively integrates these different aspects to provide an effective means of assessing content-based trust in contracts.

As with any approach which is improved by drawing on ever more data, optimisations need to be made to realistically scale. First, we use the k-nearest neighbour

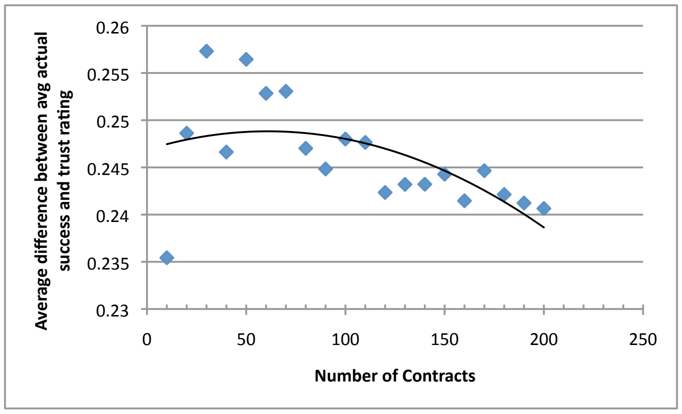

Fig. 3. The difference between trust evaluation and actual success of a proposal (Y-axis) as the number of prior contracts increases (X-axis).

measure as an example similarity measure, but the same overall technique described here applies with other algorithms. Second, optimisations are possible even when retaining the k-nearest neighbour approach, e.g. by pre-computing the similarity of past contracts A and B, to approximate the similarity of proposal P to B based on its calculated similarity to A. In either case, this is an interesting avenue of work, for which we would not reinvent optimisations but draw on existing machine learning research.

While the work presented here considers proposals from unfamiliar agents, we may also expect to see unfamiliar contract proposals from familiar agents, i.e. those for which the agent receiving the contract has some direct or indirect reputation metrics. We cannot automatically assume that an agent providing a robust contract in one area is doing so in another area, e.g. your mobile phone provider may be excellent, but you may not wish to rely on them providing reliable medical insurance contracts. In such cases, the trustworthiness of the contract content can be complemented by the metrics regarding the agent, to ensure that we can expect the contract both to be reliably fulfilled and its effect to be as we expect and desire. A simple way to combine contract and agent trust metrics would be to perform a weighted sum. Exactly how best to weight each aspect, and how beneficial this combination would be, is a topic for future work.

Acknowledgements The research described in this paper is partly supported by the European Commission Framework 6 funded project CONTRACT (INFSO-IST-034418). The opinions expressed herein are those of the named authors only and should not be taken as necessarily representative of the opinion of the European Commission or CONTRACT project partners. We also gratefully acknowledge support from the US Air Force Office of Scientific Research (AFOSR) with grant number FA9550-06-1-0031.

References

1. Aerogility. http://www.aerogility.com/, 2009.

2. IST CONTRACT project. http://www.ist-contract.org, 2009.
3. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
4. G. Boella and L. W. N. van der Torre. Regulative and constitutive norms in normative multiagent systems. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pages 255–266, 2004.
5. C. Castelfranchi. Prescribed mental attitudes in goal-adoption and norm-adoption. *Artificial Intelligence and Law*, 7(1):37–50, March 1999.
6. V. Dignum, J. J. Meyer, F. Dignum, and H. Weigand. Formal specification of interaction in agent societies. In *Proceedings of the Second Goddard Workshop on Formal Approaches to Agent Based Systems*, pages 37–52, 2002.
7. D. Gambetta. *Trust: Making and Breaking Cooperative Relations*, chapter Can we trust trust?, pages 213–237. Basil Blackwell, 1988.
8. Y. Gil and D. Artz. Towards content trust of web resources. *Journal of Web Semantics*, 5(4):227–239, 2007.
9. G. Governatori. Representing business contracts in ruleml. *International Journal of Cooperative Information Systems*, 14(2–3):181–216, 2005.
10. D. Grossi. *Designing Invisible Handcuffs*. PhD thesis, Dutch Research School for Information and Knowledge Systems, 2007.
11. F. Guerin and J. Pitt. Proving properties of open agent systems. In *The First International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS-2002)*, pages 557–558, 2002.
12. A. J. I. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of the IGPL*, 3:427–443, 1996.
13. S. Miles, P. Groth, S. Munroe, and L. Moreau. Prime: A methodology for developing provenance-aware applications. *ACM Transactions on Software Engineering and Methodology*, June 2009.
14. 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*, 51(4):52–58, April 2008.
15. L. Moreau, B. Plale, S. Miles, C. Goble, P. Missier, R. Barga, Y. Simmhan, J. Futrelle, R. E. McGrath, J. Myers, P. Paulson, S. Bowers, B. Ludaescher, N. Kwasnikowska, J. Van den Bussche, T. Ellkvist, J. Freire, and P. Groth. Open Provenance Model. http://openprovenance.org/, 2009.
16. N. Oren, S. Panagiotidi, j. Vazquez-Salceda, S. Modgil, M. Luck, and S. Miles. Towards a formalisation of electronic contracting environments. In *Proceedings of the Workshop on Coordination, Organization, Institutions and Norms in Agent Systems at AAAI 2008 (COIN 2008)*, 2008.
17. S. Panagiotidi, S. Alvarez, J. Vazquez, N. Oren, S. Ortega, R. Confalonieri, M. Jakob, J. Biba, M. Solanki, and S. Willmott. Contracting language syntax and semantics specification. Available from http://www.ist-contract.org, 2009.
18. S. Ruohomaa and L. Kutvonen. Trust management survey. In *Proceedings of iTrust 2005*, 2005.
19. J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33–60, 2005.
20. Y. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36, 2005.
21. L. van der Torre and Y. Tan. Contextual deontic logic. In Pierre Bonzon, Marcos Cavalcanti, and Rolf Nossum, editors, *Formal Aspects of Context*, pages 143–160. Kluwer Academic Publishers, Dordrecht, 1997.