# A Personalized Framework for Trust Assessment

Trung Dong Huynh
School of Electronics and Computer Science
University of Southampton
Highfield, Southampton SO17 1BJ, UK
tdh@ecs.soton.ac.uk

## ABSTRACT

The number of computational trust models has been increasing rapidly in recent years, yet their applications for automating trust evaluation are still limited. The main obstacle is the difficulty of selecting a suitable trust model and adapting it for particular trust modeling requirements, which vary greatly due to the subjectivity of human trust. The Personalized Trust Framework (PTF) presented in this paper aims to address this problem by providing a mechanism for human users to capture their trust evaluation process in order for it to be replicated by computers. In more details, a user can specify how he selects a trust model based on information about the subject whose trustworthiness he needs to evaluate and how that trust model is configured. This trust evaluation process is then automated by the PTF making use of the trust models flexibly plugged into the PTF by the user. By so doing, the PTF enable users reuse and personalize existing trust models to suit their requirements without having to reprogram those models.

## Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Types of Systems—*decision support*; D.2.11 [**Software Engineering**]: Software Architectures—*domain-specific architectures*; I.2.m [**Artificial Intelligence**]: Miscellaneous—*trust, reputation*

## Keywords

trust, reputation, extensibility, personalization, framework

## 1. INTRODUCTION

Trust research in computer science has recently generated numerous computational trust models (see [11] and [8] for examples). The main goal of these models is typically to capture the trust dynamics in human societies and replicate it (along with its benefits) in computer environments. In other words, they aim to automate the human trust evaluation process by delegating this task to computers. Despite the abundance of available trust models, their success has been limited. The main reason is that existing trust models typically make assumptions about the availability of the input information they require and, as a result, do not generalize well to a different application domain than those they were designed for. As a result, although some might work well within their target domains, none of them is ready for generic applications. Such limitation is understandable (and expected). Trust models are domain dependent because they rely on information about the trust subjects — the entities whose trustworthiness is being evaluated — which varies with different types of subject and is typically unique in a particular application domain. For instance, evaluating the trustworthiness of a sensor requires the examination of a (significantly) different set of information than that when evaluating the trustworthiness of an online news agency. In addition, trust, by nature, is subjective [4], and even for the same application, each person has their own view about how trust should be modeled in that application's domain. Hence, mainstream users have yet to adopt existing trust models (with the exception of very simplistic models like eBay's [12]) since they cannot adapt a computational trust model to sufficiently match their mental ones (i.e. change its behavior) without resorting to re-programming it. This is cumbersome and requires the end users to understand the internal working of particular trust models. In addition, different trust models use different trust representations that are an inherent part of their mathematical underpinnings. Some examples are: trust labels (e.g. very trustworthy, trustworthy, untrustworthy, very untrustworthy [1]), scalar numbers (e.g. a real number in $[-1, 1]$ [7]), probability density function [14], etc. An end user might not be familiar with any of these variants except the one that his organization has already been using. Therefore, results produced by trust models, where possible, should be converted into a customized representation that can be accepted and shared by end users from a particular organization.

Against this background, in this paper, we presents a novel framework called the *Personalized Trust Framework* (PTF) that facilitates the utilization of existing trust models by allowing different trust models to be flexibly plugged into the framework and customized by end users. The PTF provides a number of notable support functions. First, it enables end users to control the trust evaluation process by defining which trust models will be used in which circumstances and in what ways. Second, end users can define their


*SAC'09* March 8–12, 2009, Honolulu, Hawaii, U.S.A.

own trust value representation and how trust values of a different representation are automatically converted to theirs. Third, all such user personalization is stored in a *trust profile*, which can be easily transferred to other users for reuse. Customizations to the trust evaluation can be made via editing the trust profile without having to reprogram the PTF. Finally, in addition to the above, despite the high automation of trust evaluation afforded by the PTF, end users will still maintain a degree of control over the process. For example, they can trace how an automated trust decision was arrived at by examining its provenance recorded by the PTF. The framework will also notify the users of any abnormality (for example when it cannot produce a reliable trust value) and prompt for user intervention in order to correct this.

For the purpose of illustrating the design and concepts of the PTF in this paper, we will discuss the application of the PTF for the problem of evaluating the reliability of intelligence information by military analysts. This problem is particularly interesting because of the variety of intelligence information (e.g. open-source intelligence, human intelligence, and signal intelligence) requires different approaches to trust evaluation depending on the type of information being evaluated. In this particular domain, information reliability is rated using the Intelligence Source Reliability rating [5], which uses labels from $A$ to $F$ to denote the reliability of the intelligence source. $A$ means 'Reliable', $B$ 'Usually reliable', $C$ 'Fairly reliable', $D$ 'Not usually reliable', $E$ 'Unreliable', and $F$ 'Cannot be judged'.

The rest of the paper is organized as follows. Section 2 introduces the PTF and its components. Section 3 then discusses related work. Finally, Section 4 concludes the paper and outlines potential future work.

# 2. THE PERSONALIZED TRUST FRAMEWORK

As introduced in the previous section, the PTF is designed to automate the trust evaluation process by providing a mechanism for human users to capture their reasoning process in order for it to be replicated by computers. The focus of this paper is, therefore, on developing such a mechanism, not on trust modeling (which has already been the focus of much research recently). Hence, it is assumed that end users will provide the trust models they want to automate (which can be selected from the many available models, such as those reviewed in [11] and [8]) and plug them into the PTF in real-world applications. At a more detailed level, the PTF aims to address the following issues:

- How a user can define his trust model selection strategy, i.e. matching a trust subject to be evaluated with a suitable trust model.
- How an arbitrary trust model can be instantiated and used by the framework.
- How trust values of different representations can be transformed into a common representation.

In order to cater a wide range of applications, the PTF makes extensive use of semantic web technologies [2], which allow information to be represented in a machine-understandable format, to enable the PTF's information processing capabilities to be extensible to any application domain. More details on this will be provided in the subsequent sections.

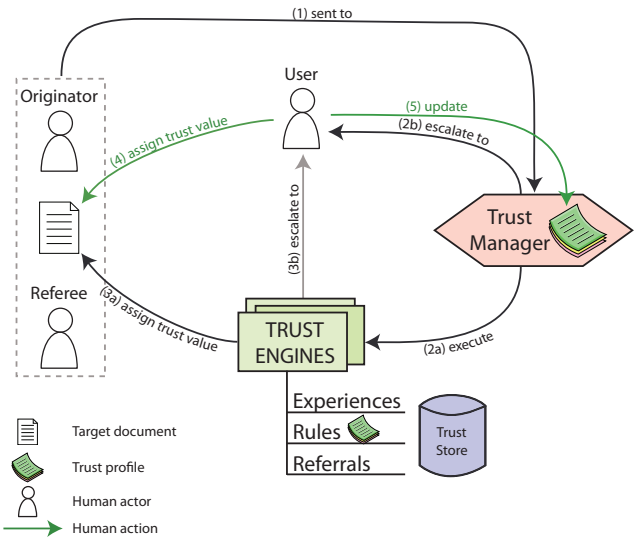


**Figure 1: The trust evaluation process by PTF.**

In the next section, we present how trust evaluation is carried out in the PTF (Section 2.1). We will then discuss about the main components of the PTF: the Trust Manager (Section 2.2), converters (Section 2.3), and trust engines (Section 2.4). Since trust evaluation may have a significant impact on critical decisions, Section 2.5 shows how the provenance of every trust decision is recorded for future auditing.

## 2.1 An Overview of the PTF

The PTF makes use of existing trust models to evaluate the trustworthiness of trust subjects sent to it. In the context of intelligence information, a trust subject is a piece of information that needs to be evaluated to determine its credibility. For the sake of simplicity, we will subsequently be using the term "*document*" to refer to any piece of information to be assessed independently by the framework. Figure 1 provides an overview of the whole trust evaluation process, coordinated by a software component called the *Trust Manager*. The normal flow of execution is as follows:

1. A document is sent to the Trust Manager along with its meta-data (1), such as its document type, content type, originator, referee, etc.
2. The Trust Manager matches the document's meta-data with its trust profile to determine a suitable trust model to be used. A corresponding trust engine — an implementation of the selected trust model — is initialized with appropriate parameters as specified by the trust profile. The engine then evaluates the document's trustworthiness (2a).
3. The selected trust engine, depending on which trust model it implements, will derive the trust value of the document from its meta-data using rules, from previous experiences with its provider, or from referrals from trusted sources. The trust value is then returned back to the main application (3a).

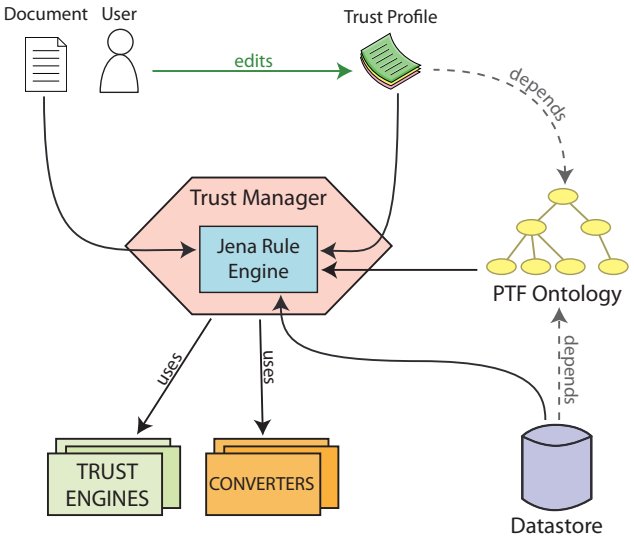


**Figure 2: The PTF's components.**

This normal flow is automated without human intervention. However, if the Trust Manager detects exceptions in the process, it will notify the user. The main exceptions are: the Trust Manager cannot select a trust model based on the policy in its trust profile (2b), for example when it receives a document of an unknown type; and the selected trust engine is not confident that it produced a reliable trust evaluation (3b), for example when it has no information, or too little information, for its evaluation. The user can then examine the document and decide on its trustworthiness (4). Where possible, the user can also add new rules or update the ones in his trust profile so that the Trust Manager will know how to deal with similar cases in the future (5).

As mentioned above, semantic technologies allow the PTF to be extensible to new types of information without limitation. This is achievable thanks to the central role of the *PTF ontology*, which defines the building blocks of the PTF and provides the core language elements for writing trust profiles. The PTF ontology is represented in OWL[1]. New concepts (described in OWL) that are extended from, or can be mapped to, those in the PTF ontology are automatically supported by the PTF (hence its unlimited extensibility). The PTF ontology will be introduced in part along with the corresponding PTF components in the subsequent sections.

## 2.2 The Trust Manager

The PTF's Trust Manager is responsible for coordinating the trust evaluation process according to the policy in a trust profile provided by end users (see the relationships of the Trust Manager with the other PTF components in Figure 2). The trust profile is described in OWL using the concepts defined in the PTF ontology. Essentially, it contains a set of rules specifying how documents sent to the Trust Manager are classified and which trust engines are used to evaluate those documents. It also contains concepts (i.e. OWL classes) of the information domain concerned, enabling the Trust Manager to understand information fed

[1]OWL Web Ontology Language: a standard of the World Wide Web Consortium, http://www.w3.org/TR/owl-ref/.

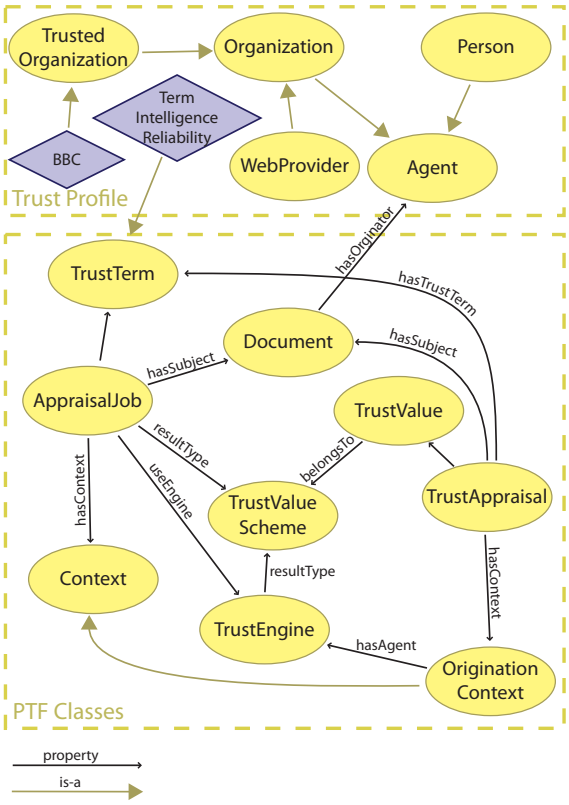


**Figure 3: The PTF ontology's main classes.**

to it from that domain. In order to execute the rules specified in the trust profile, the Trust Manager uses the Jena Rule Engine[2] [13]. The Trust Manager operates broadly as follows (see Figure 3 for the PTF ontology's classes that are referenced in italics):

1. Initially, the Trust Manager loads the PTF ontology and the trust profile into the rule engine. It then connects the rule engine with the Datastore, a RDF triple store containing all the knowledge that the system has recorded.

2. For each document sent to the Trust Manager, it loads the document's metadata into the rule engine for reasoning against the rules in the trust profile and the knowledge in the Datastore.

3. If the reasoning process results in one or more new

[2]JENA is a popular open-source Semantic Web Framework for Java. It provides libraries for working semantic web data (e.g. RDF and OWL) in various ways. For more details, see [3]. In the reference implementation of the PTF, the Jena Rule Engine is used because it supports both OWL entailments and reasoning on generic rules in one single engine with reasonable performance. As a result, we must use the Jena's rule format for writing rules. From our review, SWRL [6], an emerging rule language standard for semantic web (submitted to the World Wide Web consortium in 2004), offers better expressivity and simplicity than Jena's rule format. However, SWRL's reasoning support has yet sufficiently matured for our reasoning requirements and, thus, we decided to use Jena rule engine until a better candidate emerges.

trust evaluation jobs, stored in an instance of the *AppraisalJob* class, the Trust Manager loads the relevant instances of the *TrustEngine* class (see Section 2.4 for more details) and sends the jobs to those engines.

4. The results, in the form of instances of the *TrustAppraisal* class, will be returned to the main application.

In order to illustrate the process above, consider a scenario where online news articles (as open-source intelligence) are collected and evaluated on their reliability using the Intelligence Source Reliability rating. In this scenario, each *Document* instance represents an online news article. The metadata accompanying a *Document* contains only the URL (i.e. the web address) of the original article. For instance, we have the following documents:

| URI | *hasURL* |
|---|---|
| demo:DocBBC1 | news.bbc.co.uk/1/.../7219312.stm |
| demo:DocAJ1 | english.aljazeera.net/...925381.html |

In the trust profile, we define the classes *Agent*, *Organization*, *Person*, and *WebProvider*, and a property *hasOrginator* (see Figure 3), allowing us to represent the fact that a document is originated from an agent, which can be a person, an organization, or a web provider. The trust profile also has a number of instances of *Organization* class: *BBC*, *Al_Jazeera*. In addition, each organization has a property called *hasURLPattern* to store the pattern of the addresses of web pages published by it. For classifying the documents in this scenario, three rules are added to the trust profile:

1. *Rule_ClassifyWebProviders*:
   `(?org rdf:type ptf:Organization)`
   `(?org ptf:hasURLPattern ?pattern)`
   → `(?org rdf:type ptf:WebProvider)`
   This rule, written in the Jena's rule format, states that if an organization `?org` has the property *hasURLPattern* asserted then `?org` is an instance of the class *WebProvider*. From now on, for the sake of simplicity, we will only explain what a rule does and omit the rule's (usually lengthy) definition in Jena's rule syntax.

2. *Rule_URLPatternOriginator*: If the URL of a document matches an organization's URL pattern then the document must have been originated from that organization.

3. *Rule_WebDocumentEngineSelector*: If a document was originated by a *WebProvider* then create an *AppraisalJob* to evaluate the document on the *Term_Intelligence_Reliability* using the engine *WebProviderRuleBasedEngine* and the job must return a trust value from the scheme *Intelligence_Reliability_Rating* (see Figure 4).

With these three rules, when *DocBBC1* is sent to the Trust Manager, it can infer that the document was originated from BBC (Rule 2) and BBC is a *WebProvider* (Rule 1). Since *DocBBC1* was originated from a *WebProvider*, an *AppraisalJob* called *DocBBC1Job* is created for the document to be evaluated by *WebProviderRuleBaseEngine* (Figure 4). In the PTF, trust subjects are evaluated not on the general trustworthiness (which is an ambiguous term), but on a specific trust term. For instance, the third rule above specifies that the document will be evaluated on its intelligence reliability (*TermIntelligenceReliability*), which is a custom *TrustTerm* defined in the trust profile. Hence, with

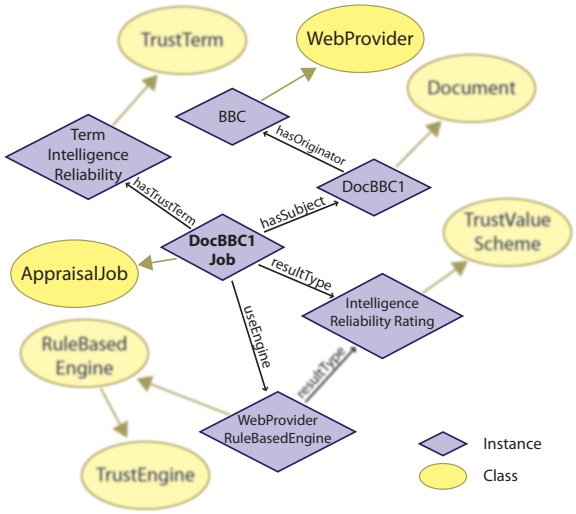


**Figure 4: *DocBBC1Job*: an instance of *AppraisalJob* generated by the Trust Manager (with the new relationships/facts highlighted).**

the PTF, end users can flexibly define different trust terms according to their needs and a document can have multiple trust values on different trust terms (e.g. honesty, cooperativeness, and so on).

## 2.3 Converters

In addition to trust terms, the *AppraisalJob* in the previous section also specifies the expected type of the trust value, which is *IntelligenceReliabilityRating* in this case. Since the *WebProviderRuleBaseEngine* happens to produce trust values of the type required (see Figure 4). The resultant trust value can be returned to the main application immediately. However, in the case that a trust engine does not produce trust values in the required representation, the Trust Manager will look for a suitable converter in the trust profile, and if it finds one, it will execute the converter to transform the resultant trust value into the required representation. If no suitable converter is available, the Trust Manager will raise an exception and prompt for user intervention. Therefore, end users should define necessary converters in the trust profile to enable seamless trust evaluation.

In the PTF, converters can be defined using Jena rules. For example, consider the FIRE trust model which produces scalar trust values in the range $[-1, +1]$, where $-1$ means absolutely negative, $+1$ means absolutely positive, and 0 means neutral. To translate a trust value using FIRE's scalar representation to the Intelligence Reliability rating, we define a converter in the trust profile with rules like this: "if the FIRE scalar value is greater than or equal to 0.25 and less than 0.75 then the Intelligence Reliability rating is B."

In cases where Jena rules are not sufficiently expressive for the required complex conversion calculations (e.g. converting from a probability density function as used in [14]), a custom Java class can be provided with its class name registered in the trust profile. The Trust Manager will load the Java class when the converter is required and the custom Java class will be responsible for the conversion.

## 2.4 Trust Engines

Having created appraisal jobs for incoming documents, the Trust Manager puts the jobs into a queue and executes them independently. For each job, it loads the specified trust engine and sends the job to the engine. A trust engine is identified by a unique URI[3], for example the URI for the *WebProviderRuleBasedEngine* is `ptf:WebProviderRuleBasedEngine`. The URI is used to locate the trust engine's specification in the trust profile, which helps the Trust Manager know how to build the engine when required. After a trust engine is successfully executed, it produces an instance of the class *TrustAppraisal* which contains links to the target document, the resultant trust value, the trust term on which the document was evaluated, and a confidence value. The confidence value reflects the reliability of the trust value as judged by the trust engine and is calculated in different ways, depending on the particular trust model used. The Trust Manager uses this confidence value to determine if the resultant trust value can be returned to the main application immediately or a user decision is needed (based on a preset threshold).

In the PTF, a trust engine is typically provided in a Java class. A *TrustEngine* instance in the trust profile then provides the link to the Java class of the corresponding engine, allowing the Trust Manager to load the engine when required. An exception to this is the class *RuleBasedTrustEngine*, a special sub-class of *TrustEngine*. Trust engines of this class use rules to derive trust values and rely on the Trust Manager's rule engine for reasoning capabilities. When building such engines, the Trust Manager simply loads their rules directly from the trust profile. Hence, the class *RuleBasedTrustEngine* allows end users to build simple trust engines in their trust profiles very quickly. For example, in our evaluation, the *WebProviderRuleBasedEngine* is built from the following two Jena rules:

1. *Rule_TrustedOrgDefault*: If a document originated from an organization of the class *TrustedOrganization* then its Intelligence Reliability rating is B (i.e. 'usually reliable') and the confidence value of this assessment is 0.5.

2. *Rule_UnknownOrgDefault*: If a document was originated from an unknown organization (i.e. not in the class *TrustedOrganization*) then its Intelligence Reliability rating is F (i.e. 'cannot be judged') and the confidence value of this assessment is 0.5.

Since we asserted in our trust profile that only the BBC is a *TrustedOrganization* (Figure 3), the execution of *WebProviderRuleBasedEngine* on the sample documents in Section 2.2 will give rating B to *DocBBC1* and rating F for *DocAJ1*, a simple and quick classification based on the documents' URLs.

In addition to the above, the PTF ontology also allows users to define composite trust engines (class *CompositeEngine*) that produce trust values by combining the results of two or more trust engines (which can even be composite engines themselves) in some way. For example, in our reference implementation, we defined a trust engine called *MaxConfidenceEngine*, which executes the *WebProviderRuleBasedEngine* and the *FIREEngine* (using the FIRE model) on the same document and selects the result with a higher confidence value. By so doing, the *MaxConfidenceEngine* always has a trust value from the simpler *WebProviderRuleBasedEngine* as a backup in the cases where the more sophisticated *FIREEngine* cannot find sufficient evidence to produce a reliable trust value. The possible way of combining different trust engines are unlimited.

[3]A Uniform Resource Identifier (URI) is a string of characters used to identify a resource on the Internet.

## 2.5 Auditing

Since automated trust evaluations may have a significant impact to critical decisions (in military operation planning, for example), from time to time, end users will need to know how certain trust values were produced. In the PTF, this is captured using the class *OriginationContext*. Each *TrustAppraisal* instance produced by the PTF is accompanied with an *OriginationContext* instance that records the minimum following information[4]:

- Subject: the link to the *TrustAppraisal* concerned.
- Timestamp: when the evaluation was carried out.
- Agent: the link to the trust engine used (and that to the converter if the original trust value was converted).
- Originator: the link to the original *AppraisalJob* that led to the *TrustAppraisal* concerned.

In addition to providing a way for end users to trace back the trust evaluation of a particular document, the information in an *OriginationContext* instance can be useful for other information management purposes. For example, when the user updates one of his trust engines, trust appraisals produced by that engine can become outdated or invalidated. The context associated with such appraisals can help to flag documents with outdated trust appraisals when they are displayed to the user.

# 3. RELATED WORK

An automated trust framework that allows trust models of different types to be used together and allows user personalization to the trust evaluation process is a novel idea that has not been investigated in the current literature. However, a number of the PTF's components have some similarities with existing work in other areas. First, the problem of rules/policy-based management has been studied extensively in various work (see [9]), whose focus is mainly on how to apply rules and regulations on various processes (e.g. identity authentication, access rights, obligations). The PTF's Trust Manager, however, uses rules as an extensible means to replicate a human's reasoning process (in selecting a suitable trust model). The idea of using rules for deriving trust (i.e. *RuleBasedTrustEngine*) is not new in trust research. Nevertheless, it is usually considered rather simplistic (and too domain-dependent) and, as a result, most of trust research has not followed this direction. We agree with this view, but still provide *RuleBasedTrustEngine* in the PTF because it provides a simple approach for those who are not trust experts. Such an approach can also work well in combination with more sophisticated trust models.

[4]A trust engine can add extra contextual information to the *OriginationContext* of a trust appraisal, if applicable.

In [10], a taxonomy was developed for different types of reputation (a form of trust value) representation. It also defined a number of conversion functions to facilitate reputation information exchange between different trust models. This work has similarity with the converters in the PTF. The main difference is that the conversion functions developed in their work target computational trust models while the PTF's converters aim to consolidate different trust representation into a common, human-defined representation. The work in [10] can serve as a reference for developing PTF's converters, but it cannot replace the end user's role in this process.

# 4. CONCLUSIONS

This paper has presented the PTF, a novel approach to trust evaluation automation to support information quality assessment. It is unique in providing mechanisms for end users to flexibly define trust engines and to plug in existing trust models to suit their trust evaluation requirements. Trust profiles in the PTF allow end users to capture their reasoning process in trust model selection (and controlling the trust evaluation in general), in addition to capturing domain knowledge, and enable computers to replicate the process. Moreover, with the use of semantic technologies, the PTF is extensible to any information domain and application. Despite being designed to be a highly automated solution, the PTF still provides ways for end users to audit its trust evaluation process. It will also notify the users of any anomalies and prompt for their decisions. The users thus still play an important role in this process by making decisions about trust and actively improving their trust profiles, but only when this is necessary. By doing all the above, the most important contribution of the PTF, perhaps, is that it paves the way for existing computational trust models to get closer to mainstream adoption.

The PTF is a complex framework, which can benefit from improvements in a number of areas. First, editing trust profiles is currently carried out by ontology editors, which are, inherently, not tailored for the PTF. Therefore, we plan to develop a simple editor that facilitates editing and managing trust profiles (e.g. creating rules from PTF and domain concepts, testing rules). Second, we will investigate further the reuse of trust profiles. An example reuse scenario can be as follows: (1) an organization uses a default trust profile to bootstrap its members' PTF; (2) each member improves their trust profile when prompted for decisions by their PTF; (3) the improvements (i.e. emerging trust evaluation practice) can then be analyzed and incorporated back into the organization-wide trust profile. Finally, reasoning over a large dataset stored in a database backend (by the Trust Manager) performs poorly as a result of the vast numbers of database interactions involved. This is an open research problem that is not particular to the PTF but affects its performance nevertheless. On this front, we are exploring a number of caching strategies that selectively load only a relatively small set of relevant data to computer memory (before a trust subject is evaluated) for a significant improvement of the reasoning performance yet can still maintain the correctness of the reasoning.

# 5. ACKNOWLEDGMENTS

This work was carried out in the research project Multi-variant Information Management & Exploitation (MIMEX) at the University of Southampton. The MIMEX initiative is supported by a grant awarded to General Dynamics UK Ltd, the University of Southampton and the University of Cardiff as part of the DIF DTC initiative funded by the UK Ministry of Defence. General Dynamics UK Ltd act as cluster lead for the MIMEX initiative and sponsor the research undertaken therein. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the UK Ministry of Defence, or the UK Government.

# 6. REFERENCES

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, volume 6, Jan 2000.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[3] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83, 2004.

[4] D. Gambetta. Can we trust trust? In D. Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*, pages 213–237. Department of Sociology, University of Oxford, 2000.

[5] Headquarters, Department of Army, USA. *Human intelligence collector opperations*. 2006. Field Manual No. FM 2-22.3.

[6] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML. Technical report, W3C Member Submission, 2004.

[7] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. An integrated trust and reputation model for open multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 13(2):119–154, September 2006.

[8] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[9] T. Phan, J. Han, J.-G. Schneider, T. Ebringer, and T. Rogers. A survey of policy-based management approaches for service oriented systems. In *Proceedings of 19th Australian Conference on Software Engineering (ASWEC)*., pages 392–401, 2008.

[10] I. Pinyol, J. Sabater-Mir, and G. Cuni. How to talk about reputation using a common ontology: From definition to implementation. In *Ninth Workshop on Trust in Agent Societies*, pages 90–102, 2007.

[11] S. Ramchurn, T. Huynh, and N. R. Jennings. Trust in multiagent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.

[12] P. Resnick and R. Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of eBay's reputation system. In M. R. Baye, editor, *The Economics of the Internet and E-Commerce*, volume 11 of *Advances in Applied Microeconomics*. Elsevier Science, 2002.

[13] D. Reynolds. Jena 2 inference support. Technical report, HP Labs, Dec 2007.

[14] W. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. TRAVOS: Trust and reputation in the context of inaccurate information sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 12(2):183–198, 2006.