Trust Assessment Using Provenance in Service Oriented Applications

Shrija Rajbhandari, Arnaud Contes, Omer F.Rana, Vikas Deora, and Ian Wootten
School of Computer Science
Cardiff University, UK
S.Rajbhandari@cs.cardiff.ac.uk

Abstract

Workflow forms a key part of many existing Service Oriented applications, involving the integration of services that may be made available at distributed sites. It is possible to distinguish between an "abstract" workflow description outlining which services must be involved in a workflow execution and a "physical" workflow description outlining the particular instances of services that were used in a particular enactment. Provenance information provides a useful way to capture the physical workflow description automatically especially if this information is captured in a standard format. Subsequent analysis on this provenance information may be used to evaluate whether the abstract workflow description has been adhered to, and to enable a user executing a workflow-based application to establish "trust" in the outcome.

An analysis tool that makes use of provenance information to assist in evaluating trust in the outcome of a workflow execution is presented. The analysis tool makes use of a rule-based engine, supporting a range of queries on the recorded provenance information by one or more workflow enactors. This paper presents performance evaluations of the analysis tool by featuring a rule and show that the tool is scalable. We also present the implementation of our trust calculator that uses the analysis tool and describe this with a real-world application scenario.

1 Introduction

Computational scientists in recent years have been increasingly relying on distributed computing technologies as an essential part of their everyday research. Although the concept of sharing distributed resources amongst geographically distributed groups is not new, increasing advancement in Service Oriented Architectures (SOA) in Grid and Web Services makes the vision more realistic. Amongst the consequences of the progress toward SOA in scientific domain is an increased emphasis on provenance data, and the need for mechanisms to acquire, use and manage such data. Workflow forms a key part of many existing Service Oriented applications that involves the integration of services that may be made available at distributed sites. It is possible to distinguish between an"abstract" workflow description outlining which services must be involved in a workflow execution and a "physical" workflow description outlining the particular instances of services that were used in a particular enactment. Provenance information provides a useful way to capture the physical workflow description automatically especially if this information is captured in a standard format [7]. Subsequent analysis on this provenance information may be used to evaluate whether the abstract workflow description has been adhered to, and to enable a user executing a workflow-based application to establish "trust" in the outcome of the physical workflow.

Many research scientists make use of distributed resources or services in their experimental workflows, and at some point they may wish to share the produced results with their fellow researchers. Our work aims to assess user's "trust" in the result that is the outcome of a workflow execution. Such assessment can be useful if the user decides to utilize the result of a workflow enactment in other workflows or take important actions. It may be ideal for potential users wishing to use the result to acquire an approach to establish some degree of trust in the result. Provenance information along with the result improves a user's ability to judge the validity of the result. Although provenance provides justification for the result, the notion of how much trust can be placed in the result is completely implicit – to the extent that such concern has not been fully addressed in existing workflow systems. This paper introduces an analysis tool that makes use of provenance to assist in the "trust assessment" of the result that has been produced through a distributed workflow session. The analysis tool makes use of a rule-based engine, supporting a range of queries on the recorded provenance information by one or more workflow enactors. The analysis tool provides information that is consumed to elicit/attain the measure of "result trustworthiness".

Related work on trust models is presented in Section 2. Section 3 presents our trust architecture. Section 4 provides the rule-based analysis tool that is the basic for our trust architecture and section 5 provides an evaluation of our model with the workflow scenario from BioDiversityWorld(BDW) project.

2 Related Work

There exists a large number of proposals in the literature for calculating "trust" commonly based on reputation or QoS for actors [9, 17, 11, 21]. Such approaches achieve trust evaluation in two parts. Firstly, to allow actors to trust each other, there is a need to endow them with the ability to reason about the reliability, or honesty of their counterparts. This ability is captured through trust models. The latter aims to enable actors to calculate the amount of trust they can place in their interaction partners. A high degree of trust in an actor would mean it is likely to be chosen as an interaction partner. Hence, trust models aim to guide an agent in deciding how, when, and who to interact with. However, in order to do so, trust models initially require actors to gather some knowledge about the characteristics of their counterparts. Based on existing work, this may be achieved as follows:

1. **A presumption drawn from the actor's own experience:** Trust is computed as a rating of the level of performance of the actor. The actor's performance is assessed over multiple interactions checking how good and consistent it is at doing what it says it does. To this end, Witkowski et al. [19] propose a model whereby the trust in an actor is calculated based on its performance in past interactions. Similarly, Sabater et al. [17] propose a similar model but do not just limit the overall performance to the actor's direct perception, but they also evaluate its behavior with other actors in the system.

2. **Information gathered from other actors:** Trust in this approach is drawn indirectly from recommendations provided by others. As the recommendations could be unreliable, the actor must be able to reason about the recommendations gathered from the other actors. The latter is achieved in different ways: (1) deploying rules to enable the actors to decide which other actors' recommendation they trust more [1]; (2) weighting the recommendation by the trust the actor has in the recommender – EigenTrust [9] and PageRank [13] are examples of this approach.

3. **Socio-Cognitive Trust:** Trust here is drawn by characterizing the known motivations of the other actors. This involves forming coherent beliefs about different characteristics of these actors and reasoning about these beliefs in order to decide how much trust should be put in them [5].

Refer to [2] for more details on trust and reputation approaches. The aim of such existing work is to help in selection of a trustworthy actor based on the evaluated trust for each actor. Other approaches exists such as [20, 10, 12] which involves trust assessment for service composition. But the objective in these approaches is either to (1)select trustworthy services for activity such as VO formation or (2)compute optimal execution plans for a workflow. Our framework differs from such models as the concern is towards trustworthiness of an outcome that is the result of a scientific experiment – performed in a distributed, service oriented environment. To achieve this we recognize the importance of provenance data and exploit this in our trust architecture. Thus, apart from provenance data providing the explanation about how a result came to be, it also provides a way to formulate the trustworthiness placed in the result. A trust framework consisting of an analysis tool that makes use of provenance information to assist in evaluating trust in the outcome of a workflow execution is presented.

3 Trust Architecture

The trust architecture shown in Fig 1 consists of two main parts; (1) trust calculator and (2)analysis tool. This section describes the trust calculator. The analysis tool is described in section 4. We focus on generating a trust measure for a result that is an outcome of a workflow. This "trust assessment" via trust calculator is achieved by utilizing the analysis tool that performs a rule-based analysis on the provenance data that has been recorded about a workflow (see Fig.1). The necessary provenance describing the "physical" workflows are recorded in a standard form within a repository which is referred as Provenance Store [7].

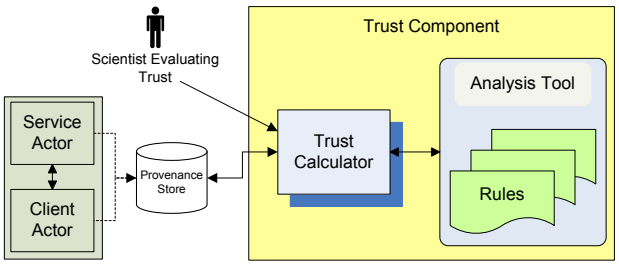

Figure 1. Trust Architecture

Provenance information utilized for trust evaluation may be categorized as follows: (1) Process Provenance: corresponds to the steps involved in the workflow that lead to a

result. It also include the inputs and outputs for each service involved in the process; (2) Actor Provenance: records the state of the actors/services involved in a particular workflow instance. This also includes static data such as actor's ownership and identity.

3.1 Trust Calculator

The trust calculator allows users to pose queries to the Provenance Store for retrieving provenance information of past workflows, for example a bioclimatic experiment [8]. A user may query data for each stage of the workflow. Trust calculator adopts a decision process for analyzing the queried provenance. The conceptual decision process is presented in [15] that uses a decision tree model which is traversed to generate trust measure for a workflow result. The decision tree consists of a set of nodes where each node has a question representing an analysis that helps to assert some trust in the workflow result. The response to a question is a boolean value. In our work presented in [15], the questions are processed only through user intervention. However, our current work involved mapping the questions in the decision tree as rules in the analysis tool so the answer for a particular question can be retrieved automatically. Trust calculator commits analysis requests (based on the questions) to the analysis tool for executing the required analysis. The analysis tool executes the analysis requested and returns the results. A boolean value is returned as a result of every analysis executed. For example, a result from checking for "no-conflict" on data passed between two services in the workflow can be either (1) "True" (positive) if there is no conflict or (2) "False"(negative) if conflict exist. The analysis tool uses a rule-based engine where different rules could be written to undertake different types of analysis. The architecture of the analysis tool is discussed in section 4 and its performance evaluation is carried out with the conflict detection rule. Trust calculator makes use of the boolean values returned by the analysis tool to generate a probable trust measure on the workflow result.

We adopt beta probability distribution for combining analysis results and for expressing trust measures. This simple approach is used as it is applicable for our current application scenario. In particular, the Bayesian theory uses standard beta distributions to model posterior probability estimates of observed binary events with two possible outcomes. The mathematical analysis that leads to the posterior probability estimates of binary events can be found in Bernardo and Smith [3]. In our case, the binary events are the different analysis performed by the analysis tool with two possible outcomes; positive and negative.

We choose the beta density function that takes the integer number of these two possible outcomes represented as parameters α and β (which represents total numbers of positive and negative outcomes respectively) to express the uncertain probability that the workflow result can be trusted upon.

3.1.1 The Beta Density Function:

The Beta Distribution is a continuous probability distribution with the probability density function defined on the interval [0, 1]. Beta distribution is defined in terms of parameters α and β. A continuous random variable has a beta distribution with parameters α and β, its density function $f(x|\alpha,\beta)$ can be expressed as;

$$f(x|\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B[\alpha,\beta]} \quad (1)$$

where, $0 \leq x \leq 1$, $\alpha > 0, \beta > 0$ and $B[\alpha,\beta]$ is the beta function with parameter α and β which is given as;

$$B[\alpha,\beta] = \int_0^1 x^{\alpha-1}(x-1)^{\beta-1}dx \quad (2)$$

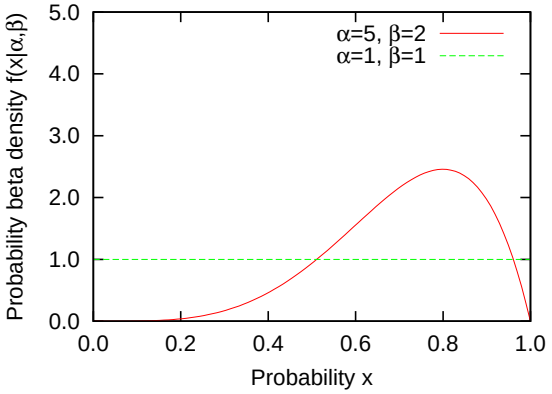

Figure 2. Uniform and an Example distributions

A special case in beta distribution is when $\alpha = 1$ and $\beta = 1$, x is said to have a uniform beta distribution. Thus when nothing is known as, i.e., no analysis is performed on the workflow, the distribution is uniform (Fig.2). Let us consider an analysis process of two possible outcomes $\{positive, negative\}$, let p be the total number of observed $positive$ outcomes and n be the total number of observed $negative$ outcomes. After this observation, the posterior distribution is the beta function with $\alpha = p + 1$ and $\beta = n + 1$. An example in Fig. 2 illustrates the distribution with 4 positive and 1 negative outcomes. This provides a firm mathematical basis for combining the responses from the analysis tool and expressing trust measures for the

results of past workflows under evaluation. Here, the probability distribution of x is continuous, so it is only meaningful to compute $f(x|\alpha, \beta)$ for a specific interval$[0, 1]$. We consider the maximum point in the distribution of x to be the "trust probability" of the result given the amounts of positive and negative outcomes from the analysis. In Fig. 2 example with 4 positive and 1 negative outcomes, the "trust probability" is 0.8. This indicates that given the number of analysis outcomes, the probability to trust the result is 0.8. In a case with 5 positive and 0 negative outcomes, the trust probability is approx. 1 reflecting complete trust in the result.

4 Analyzing Workflow to Elicit Trust

As stated previously in section 3, the assessment of the amount of trust for a particular result is obtained by processing the decision tree. In [15], these questions have been shown under a natural language representation. In this section, we describe the underlying mechanism behind the computation of the answer to a particular question.

4.1 Analysis Tool

Our analysis tool makes uses of the Java Expert System Shell (`JESS`), a java rule engine. `JESS` uses an enhanced version of the `Rete` algorithm to process rules. `Rete` is an efficient mechanism for solving the difficult many-to-many matching problem (see for example [4]). The `Rete` algorithm expects two different types of input, (1) a set of *rules* which represent the logic of the computation (also called *production rules*) and (2) a set of facts which represent the data to be analysed (also called *working memory*).

The data produced by the execution of a provenance-aware workflow is composed of a set of *p-assertions*. Such set of p-assertions provide the description of the physical workflow. A p-assertion can be used to record one of the following events : an interaction between two actors (each actor records their view of an interaction), the state of an actor at a particular moment or a relation between two events.The analysis tool can also be used to detect possible conflicts in the p-assertions recorded. The nature of detected conflicts is large and various, from detecting a difference between the data submitted by the sender and by the receiver of a given interaction, to the detection of unexpected behaviour during the execution of a workflow.

In the current implementation, p-assertions are provided under an `XML` format, defined by a particular schema referred as a `PStructure`. A p-assertion is composed of two kinds of data : the first describes provenance-specific data like issuer and receiver actors, unique identifier for the p-assertion and the second represents application-specific data. This application-specific data is also under an `XML` format but defined by external schemas. Indeed, the meaning of this data varies with the application. So the analysis tool has to manipulate data whose a part of the structure (and the meaning) is known (the provenance-specific part) and another part could vary according to the monitored application.

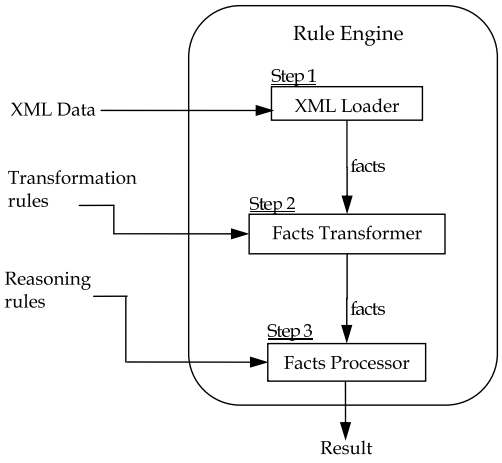

Figure 3. Architecture of the analysis tool

Our analysis tool avoids the fact to have dedicated `XML` parser, it is able to import any kind of `XML` content and performs some reasoning on these data. The architecture of the analysis tool, shown in Figure 3, makes use of three components : the *XML Loaded*, the *Facts Transformer* and the *Facts Processor*. The processing of data is composed of three successive steps involving the three components introduced previously:

- *Step 1 - Populating the Rule Engine*: the *XML loader* is in charge of converting the `XML` into a set of *generic facts*. We have defined two generic templates to map the `XML` structure into a set of facts. The first, called `Element`, represents an `XML` element. The second one, called `Attribute`, represent an attribute associated with a particular `XML` element. Each element of the `XML` document is loaded into the memory of the rule engine as a *fact* called *Element*. Each `Element` is identified by a unique identifier `ElementID`. An `Attribute` shares the same `ElementID` that the `Element` belongs to. The relations between elements (parent, sibling, sub-element, ...) are enforced by inserting in each `Element` the list of its parent, children and attributes if any. Based only on these two templates, any `XML` document can be transformed into a set of facts and loaded into the rule engine memory.
- *Step 2 - Converting Generic Facts into User-Defined*

Data Structure: Once, the data is loaded in the memory, the facts could be used directly, however it is possible that all the data is not relevant for a particular query, or that the data must be reformatted prior to be used with external decision rules. The *Facts transformer* converts the generic facts into enhanced facts. This conversion is done by introducing a set of *transformation rules*, whose goal is to transform these generic facts into a meaningful format. Although this step could be optional, it allows to format the raw data into structured ones and so simply the writing of reasoning rules. These transformation rules are expressed in the language used by the rule engine. It allows an high level of flexibility in the transformation process and in the definition of the final structure of the data. Each time that a new schema is encountered, end users have only to create or update some of the transformation rules to create schema-specific enhanced facts. Transformation rules can also be used to trigger additional rules whether a given condition is detected.

- *Step 3 - Facts Processing*: the *Facts Processor* processes the enhanced facts with a set of *decision rules* and returns the result of the computation.

4.2 Scalability

We performed experiments to evaluate performance of the analysis tool using p-assertions data. In order to evaluate the performance of the XML loading, we have performed benchmarks and have measured the time spent by this process depending on the number of generated facts. The number of generated facts ties in with the length of the XML document to load.

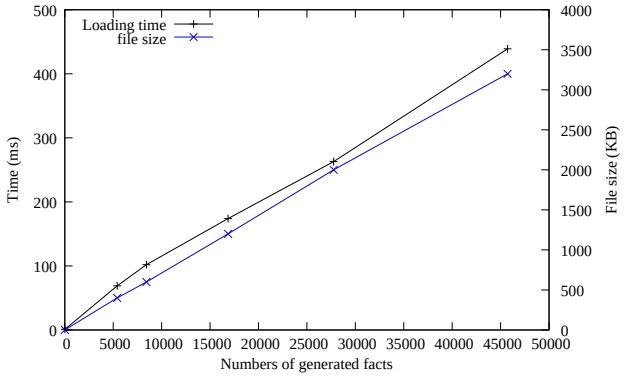

Figure 4. Populating the engine memory

The computer used for these benchmarks is a laptop with Pentium M @ 2.13Ghz, 1GB, Linux fedora core 4 with a kernel 2.6.14. The hard disk is an Hitachi 5400RPM, ATA/100, 12 milliseconds (ms). The p-assertions data used in this evaluation represents the execution of a workflow involving two actors (one client and one service) performing a simple mathematical operation of adding two numbers. The data contains the description of all the interactions exchanged.

The *loading time* curve in Figure 4 presents the average duration of the populating process depending on the number of generated facts. The average time is computed after 100 consecutive loadings. The *file size* curve presents the size of the XML document depending on the number of generated facts. With this performance evaluation, it can be concluded that :

- the loading process has a linear complexity $O(n)$.
- the average loading rate is about 103 000 facts per second, or average accepted throughput is about 7.2MB per second.

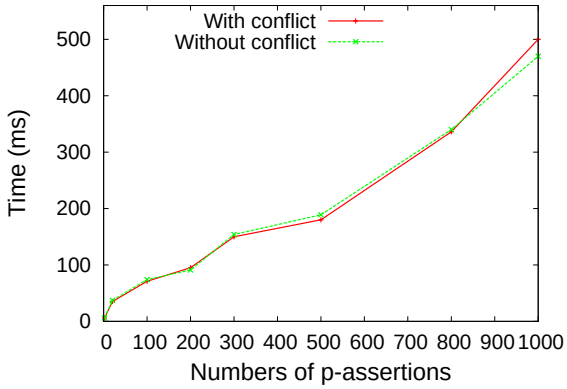

Figure 5. Conflict Detection : Computation Time

Benchmarks has been used to evaluate the performance of the conflict detection module. The conflict detection module represents a rule to confirm, for example, if data d sent by a client A to a service B is interpreted correctly by B as d. The hardware and software environment is the same as previous. The results are shown in the figures 5 and 6. Each value represents the average value of 20 successive tests. The performances of the conflict detection module has been tested on a set of p-assertions without conflict and also by introducing one conflict in a randomly selected p-assertion. The results are shown in Figure 5. Figure 6 shows the size of a set of p-assertions when stored on the filesystem and the amount of memory used by its equivalent representation when loaded inside the analysis tool. The memory usage represents the amount of Heap memory used by the Java Virtual Machine executing the conflict detection code. The values have been collected through the Java Management eXtension [18].

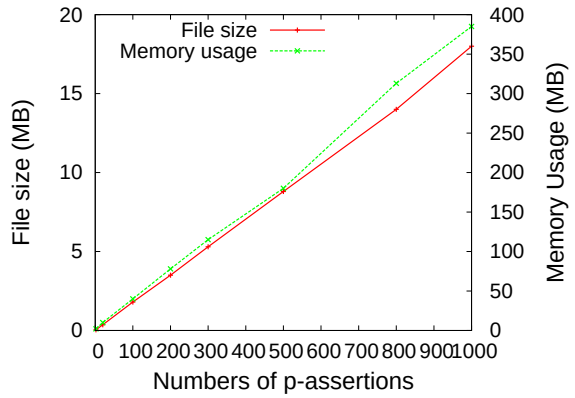

Figure 6. Conflict Detection : Memory Usage

Figure 5 shows that the time spent to detect a conflict is quite the same with or without conflict in the input data. There is a small difference in the computation time when the amount of p-assertions increase. We consider this difference insignificant. As possible explanation, it could be caused by the large amount of memory used, and the fact that Java's garbage collector can be triggered at any time. Also the fact the computer was not just running the performance code but also additional programs causing background workloads.

Figure 5 shows that the memory used to represent a p-assertion remains constant. From the data, it is possible to calculate the amount of memory used to represent a p-assertion. Given that a set of 1000 p-assertions use 18MB on the filesystem and its representation in memory uses 380MB, we can deduce that, on average, a p-assertion uses 18KB on the hard disk and its representation in memory 380KB. The ratio memory representation divided by the filesystem space used is 21. This ratio is explained by the fact that each XML element is mapped into a Java object and that the data contains within one element rarely exceeds the size of 2 characters. In [16], the author demonstrates that a plain Java `Object` takes 8 bytes, an `int` uses 16-byte result, an empty `String` takes 40 bytes – enough memory to fit 20 Java characters. So an empty XML element with a 4 characters-long name uses: 2*4 characters + 2 * `"<"` + 2 * `">"` = 12 octets whereas its representation in memory uses at least: `Object` + 4 * `String`(element name, prefix, namespace, content) = 168 bytes. The ratio is here already $168/12 = 14$.

The performance analysis demonstrates that the complexity of the tool remains linear even when the data is altered by introducing randomly generated conflicts.

5 Implementation

This section describes the implementation of our trust decision tree that performs analysis on the provenance data using rules passed to the analysis tool. We also describe the implementation with a bioclimatic modelling scenario [8].

5.1 Bioclimatic Modelling

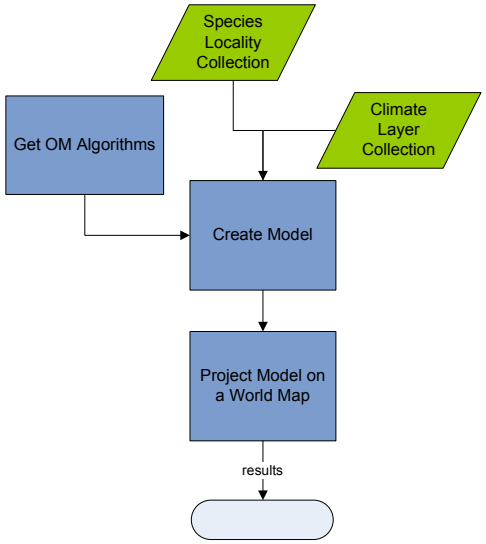

Figure 8. Bioclimatic Modelling in BDW

Our workflow representing the bioclimatic modelling of species distribution is from the BioDiversityWorld(BDW) project [8]. In Figure 8, the *Create Model* produces a bioclimatic model given the following sets of data: (1)locality data for a species, (2)a climate preference profile that is produced by referring to present day climate data for various locations and (3)a specific selected Open Modeller (OM) algorithm (e.g., Bioclimatic distance algorithm-to identify the species presence in suitable regions). Using the selected algorithm, a bioclimatic model is produced by interpolating the climatic data at the points of locality of the species. Such bioclimatic model distributions are then projected upon a world map by the *Project Model* service. Use of bioclimatic modelling allow the prediction of how species will be distributed under changing climate and to examine where a conservation priority area should be in the future for that species. It is possible to have several cases that leads to the evaluation of the projected world map produced by the given workflow [15].

5.2 Trust Decision Tree Implementation

Figure 7 shows a snapshot of trust decision tree interface where a user can perform specific query for *p-assertions* to

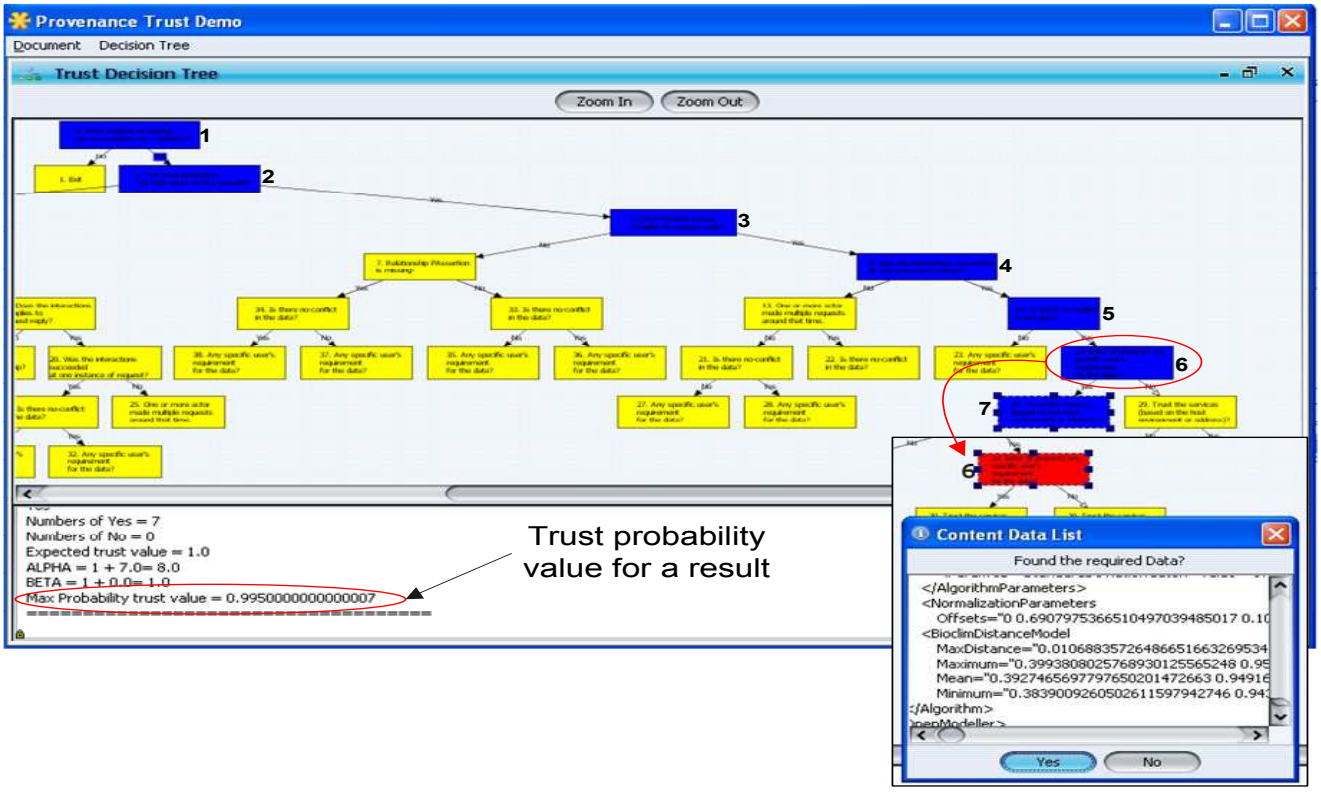

Figure 7. Trust Decision Tree Interface

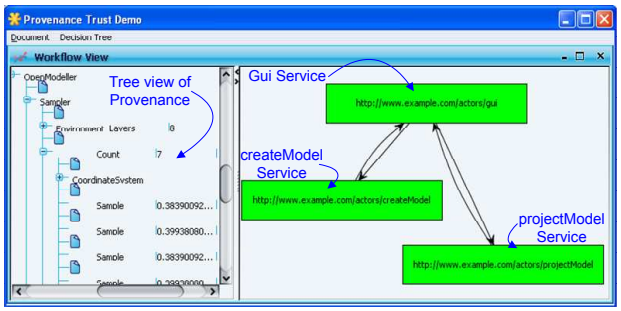

Figure 9. Example BDW physical Workflow

any given provenance store. The results retrieved can be for a particular workflow that can then be interpreted to represent a physical workflow. The analysis can be performed on the results by traversing the trust decision tree. Each tree node represents specific rule to analyze part or whole of the workflow's provenance data. Once, a leaf node is reached, trust value can be calculated using the algorithm of eq.1.

A set of *p-assertions* is generated that describes an enacted workflow consisting of two services of the BDW scenario; (1)*Create Model* service that uses the algorithm, the species locality data and the climate layer data (2)*Project Model* service projects the created model upon a world map image. The two services are enacted from a gui service. The queried provenance of this workflow can be seen in Figure 9 as a process graph that is based on the interactions between the actors/services. Figure 7 illustrates the analysis performed on this p-assertions, and the decision path is represented by numbers on the nodes. A consecutive node during the analysis process is automatically selected based on the boolean result of analysis performed on the preceding node.

An analysis in the trust decision tree contemplates the fact that the workflow consists of application specific data. Thus, user verifications of such data is as significant as automatic detection of conflicts to have some degree of trust in the final result. At node 6, in Figure 7, decision path is interrupted and a rule is executed to return the workflow's data contents for user verification. Based on the user's boolean input, the next node (node 7) is selected. Detailed discussion on the trust model evaluations can be found in [14].

6 Conclusion

This paper demonstrates that provenance information captured from a workflow enactment engine could be used in determining "trustworthiness" of the result generated

from such enactment. In this paper, we have presented an analysis tool based on JESS rule engine that is used in performing subsequent analysis on such provenance information aiming to deliver trust on workflow result. A workflow trust model is presented that adopts a decision process to execute analysis written using JESS rules within the analysis tool. The results of such analysis is eventually utilized to produce a probabilistic trust measure for the outcome of the workflow under evaluation. We use a simple beta distribution for such trust calculation as it is applicable for our current application scenario. We intend to explore more complex probability methods to measure result trust in future.

The analysis tool is used as an underlying computation for automating the trust assessment decision process to generate the "trustworthiness" that may be associated with a workflow result. We have demonstrated how a generic reasoning system like JESS engine could be integrated alongside an on-demand platform. A scalability experiment on the analysis tool is conducted to examine the performance when the provenance information (p-assertions) loaded in the engine increases. Also, the performance of the conflict detection rule has little or no effect on the computation time of increased p-assertions. Thus, both the results show that the performance decreases at a constant rate relative to load increases. Further evaluations with different data set would also confirm that if the size of the exchanged data increases, the ratio file/memory usage will decrease. For future work we intend to perform evaluations on our trust model with applications from EU provenance project [6].

References

[1] A. Abdul-Rahman and S. Hailes. Using recommendations for managing trust in distributed systems. In *Proceedings IEEE Malaysia International Conference on Communication*, 1997.

[2] Ali Shaikh Ali and Omer F. Rana and Rashid J. Al-Ali. *High Performance Computing: Paradigms and Infrastructure*, chapter Evidence-aware Trust Model for Dynamic Services. Laurence Yang and Minyi Guo, John Wiley, 2005.

[3] J. M. Bernardo and A. F. Smith. *Bayes Theorem*, pages 116–117. John Wiley & Sons, West Sussex, England, May 2000.

[4] C. L. Forgy. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, 19:17–37, 1982.

[5] R. Falcone and C. Castelfranch. Social trust: A cognitive approach. *Trust and Deception in Virtual Societies Journal*, pages 55–90, 2001.

[6] gridprovenance. *http://gridprovenance.org*, 2005.

[7] 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*, February 2006.

[8] A. Jones, R. White, N. Pittas, W. Gray, T. Sutton, X. Xu, O. Bromley, N. Caithness, F. Bisby, N. Fiddian, M. Scoble, A. Culham, and P.Williams. Biodiversity world: An architecture for an extensible virtual laboratory for analysing biodiversity patterns. In *UK e-Science All Hands Meeting, EPSRC*, pages 759–765, Nottingham, UK, 2003.

[9] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the Twelfth International World Wide Web Conference*, 2003.

[10] W. Liu. Trustworthy service selection and composition - reducing the entropy of service-oriented web. In *3rd IEEE International Conference on Industrial Informatics (INDIN)*, 2005.

[11] E. M. Maximilien and M. P. Singh. Toward autonomic web services trust and selection. In *Proceedings of 2nd International Conference on Service Oriented Computing (ICSOC 2004)*, 2004.

[12] N. Milanovic and M. Malek. Architectural support for automatic service composition. In *IEEE International Conference on Services Computing (SCC05)*, 2005.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Stanford Digital Library Technologies Project*, 1998.

[14] S. Rajbhandari. Evaluation of Provenance Trust Model. *Technical Report at http://users.cs.cf.ac.uk/S.Rajbhandari/reports/12-06-06.pdf*, 2006.

[15] S. Rajbhandari, I. Wootten, and O. Rana. Evaluating provenance-based trust for scientific workflows. In *Sixth IEEE International Symposium on Cluster Compution and the Grid(CCGrid06)*, pages 365–372, Singapore, 2006.

[16] V. Roubtsov. Do you know your data size? http://www.javaworld.com/javaworld/javatips/jw-javatip130.html.

[17] J. Sabater and C.Sierra. Regret: a reputation model for gregarious societies. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agents Systems*, 2002.

[18] Sun Inc. Java Management eXtension. http://java.sun.com/products/JavaManagement/.

[19] M. Witkowski, A. Aritikis, and J. Pitt. Experiments in building experiential trust in a society of objective-trust based agents. *Trust in Cyber-societies*, pages 111–132, 2001.

[20] S. J. H. Yang, J. S. F. Hsieh, B. C. W. Lan, and J.-Y. Chung. Composition and evaluation of trustworthy web services. In *BSN '05: Proceedings of the IEEE international workshop on Business services networks*, pages 5–5, Piscataway, NJ, USA, 2005. IEEE Press.

[21] B. Yu and M. P. Singh. A social mechanism of reputation management in electronic communities. In *Proceedings of the Second International Conference on Trust Management(iTrust'04)*, 2004.