*International Journal of Production Research*,
Vol. 43, No. 20, 15 October 2005, 4471–4492

Taylor & Francis
Taylor & Francis Group

# Knowledge provenance in enterprise information

M. S. FOX* and J. HUANG

Enterprise Integration Laboratory, University of Toronto, 40 George Street, Toronto, Ontario M5S 3G8 Canada

(*Revision received January 2005*)

Knowledge Provenance (KP) addresses the problem of how to determine the validity and origin of web information by means of modelling and maintaining information sources, information dependencies, and trust structures. Four levels of KP have been identified: (1) static KP develops the fundamental concepts for KP, and focuses on provenance of static and certain information; (2) dynamic KP considers how the validity of information may change over time; (3) uncertainty-oriented KP considers uncertain truth values and uncertain trust relationships; (4) judgement-based KP focuses on social processes necessary to support KP. This paper presents the fundamental concepts and models of KP by providing motivating scenarios, KP ontologies and examples regarding how to use KP.

*Keywords*: Enterprise integration; Knowledge management; Information trustworthiness; Knowledge Provenance; Web of trust

## 1. Introduction

The information revolution continues unabated throughout the Enterprise. The confluence of broadband global communications, inexpensive computers, information standards such as the World Wide Web and visualization technologies such as web browsers has made it possible to acquire, store, refine and disseminate vast quantities of information throughout the enterprise. Consequently, web-based Enterprise Portals have fast become the primary means by which people find and publish information across the enterprise. But there is a problem lurking behind the portal, and it arises from many directions: information may no longer be relevant (e.g. discontinued products, old operating procedures, old financial information), may contain incorrect information (e.g. news stories), and may even be outright lies. Anyone can publish information on the web, the information may be true or false, valid or dated, but no tool exists to discern the differences.

This paper introduces Knowledge Provenance (hereafter referred as KP) (Fox and Huang 2003) to create an approach to determining the origin and validity of web information by means of modelling and maintaining information sources, information dependencies, and trust structures. The major questions KP attempts to answer include: Can this information be believed to be true? Who created it? Can its creator

*Corresponding author. Email: msf@eil.utoronto

*International Journal of Production Research*
ISSN 0020–7543 print/ISSN 1366–588X online 
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540500142415

be trusted? What does it depend on? Can the information it depends on be believed to be true? This proposed approach could be used to help people and web software agents to determine the validity of web information.

Philosophically, we believe the web will always be a morass of uncertain and incomplete information. But we also believe that it is possible to annotate web content to create islands of certainty. Towards this end, KP introduces four levels of provenance that range from strong provenance (corresponding to high certainty) to weak provenance (corresponding to high uncertainty). Level 1 (static KP) (Fox and Huang 2003) develops the fundamental concepts for KP and focuses on provenance of static and certain information; Level 2 (dynamic KP) (Huang and Fox 2004a) considers how the validity of information may change over time; Level 3 (uncertainty-oriented KP) (Huang and Fox 2004b) considers uncertain truth value and uncertain trust relationships; Level 4 (judgement-based KP) focuses on social processes necessary to support KP.

The content of this paper is organized as follows: section 2 provides an overview of related research; section 3 introduces an Ontology for static KP; section 4 illustrates how to use KP including annotating web documents, provenance reasoning and implementations; section 5 extends static KP into dynamic KP; section 6 further extends static KP into uncertainty-oriented KP; finally, section 7 concludes the paper.

## 2. Related research

In the context of enterprise integration, Goranson *et al.* (2002) stated that 'knowledge' in Knowledge Management is 'justified true belief', and two types of trust may be involved: 'deductive' trust based on understanding the cause and effect mechanics; 'inductive' trust based on 'authority', 'votes', and 'experience'. 'Deductive' trust is preferred, for it is 'auditable'. This argument raised an important point: knowledge in EI needs to be justified and to be auditable. Consider the features of the knowledge in EI: uncertain, complex, incomplete, inaccurate, qualitative, informal, and so forth. 'Trust' is a necessary and efficient approach to knowledge justification, and a combination of 'inductive' and 'deductive' trust is needed.

Information source evaluation criteria, such as, Authority, Accuracy, Objectivity, Currency and Coverage, have been developed in library and information science, and have been extended to online information (Alexander and Tate 1999). Oliver (1997) collected hundreds of evaluation criteria from different sources, and consolidate into 125 indicators in 11 groups of criteria. However, these criteria were proposed for people to judge web information, that is, people have to be involved in judgement. KP attempts to create a computational model.

Interest in addressing the issue of web-information trustworthiness has appeared under the umbrella of the 'Web of Trust' that is identified as the top layer of the Semantic Web (Berners-Lee 2003, slides 26 and 27).

Digital signature and digital certification (Simon *et al.* 2001, Tan 2002, Bartel *et al.* 2002) play important roles in 'Web of Trust'. However, they only provide an approach to validate author identification and information integrity; they do not determine information trustworthiness. In the context of KP, they can only be used to determine who is the information creator and whether the information is the same

as originally defined, but they do not indicate whether the information is trustworthy. Trustworthiness is supposed to be evaluated by each web application. For the purposes of secure web access control, Blaze *et al.* (1996) first introduced 'decentralized trust management' to separate trust management from applications and created the fundamental concepts of policy, credential, and trust relationship. Chu (1997) introduced trust protocol in the REFEREE system; Khare and Rifkin (1997) proposed basic principles of trust management. However, Trust Management focuses on 'is this individual trusted to do a specific operation in my system', whereas KP attempts to answer 'is the information given by this individual trusted to be true?'

The concept of 'Web of Trust' perhaps was first developed in PGP (Zimmermann 1995), a public key cryptosystem used for encryption and digital signature, as a model of trust for a public key recipient to validate the authentication of the key, where 'trust' has a specific meaning of trusting people to validate other people's certificates. The 'Friend Of A Friend' project (FOAF, Dumbill 2002) creates social networks (Watts 1999) on the web by facilitating people to describe acquaintance relationships in machine-readable webpages. Even though acquaintance relationships do not equal trust relationships, FOAF is a good start point.

Many projects focusing on 'web of trust' based on social networks have emerged. For example, Yu and Singh (2000) proposed a model of reputation (trust) propagation and building among agents in electronic communities; Golbeck *et al.* (2002) proposed trust networks that extend the FOAF model by introducing levels of trust in acquaintance relationships and used for filtering emails; Richardson *et al.* (2003) proposed a model of trust management for the semantic web and used for bibliography recommendation. These projects created models for indirect trust calculating and could be used for justifying knowledge in the manner of 'inductive' trust.

Coming from an automated reasoning perspective, McGuinness and Pinheiro da Silva (2003) developed 'Inference Web (IW)', which enables information creators to register proofs with provenance information in IW, and then IW is able to explain the provenance of a piece of requested knowledge. IW provides provenance information (registered by creators) for users to support them deciding by themselves to trust or not trust the requested knowledge.

Finally, technologies developed in AI, such as the Truth Maintenance System (de Kleer *et al.* 1989) and Temporal Logic (Allen and Ferguson 1994), provide approaches for knowledge representation and reasoning in KP. Technologies developed in Semantic Web (Berners-Lee *et al.* 2001) provide approaches to the web implementation of KP.

## 3. Static KP

Static KP is concerned with the provenance of knowledge that is both certain and does not change over time. It is the simplest yet strongest form of provenance. Basically, any statement has a truth value of: True, False or Unknown. The default truth value is 'Unknown'. Its truth value does not change over time.

In order to give a formal and explicit specification for Static KP and to make it available on the web, a static KP ontology is defined in this section. Following the ontology development methodology of Gruninger and Fox (1995), we specify Static

KP ontology in four steps: (1) provide a motivating scenario; (2) define informal competency questions for which the ontology must be able to derive answers; (3) define the terminology (i.e. predicates); (4) define the axioms (i.e. semantics). We already discussed motivating scenarios in the earlier section. This section presents informal competency questions, terminology, and axioms.

### 3.1 *Motivating scenarios*

Case 1 (Asserted information): Consider a statement in a document found on the Intranet in an enterprise. This states that 'a delay of more than one minute in answering a phone call may result in the customer being dissatisfied'. From a provenance perspective, there are three questions that have to be answered:

- Is the statement true?
- Who created this statement?
- Should we believe the person or organization that created it?

A further examination of the text of the web document provides the answers: it can be believed to be true, created by a retired customer service manager, who most people in the company believe is an authority on the subject. Questions are:

- What is the basis for us to believe this statement as true?
- How can the provenance process be formalized?

In the remainder of the paper, we will refer to statements like the above as 'propositions'. A proposition is the smallest piece of information to which provenance-related attributes may be ascribed. A proposition is either true or false.

Case 2 (Dependent information): Consider the following proposition found in another web document: 'This new approach to reduce response-delay to less than one minute may increase customer loyalty, because a delay of more than one minute in answering a phone call may cause the customer to be unsatisfied'. This is actually two propositions composed of a premise, 'A delay of more than one minute in answering a phone call may cause the customer to be unsatisfied' and a conclusion, 'This new approach to reduce response-delay to less than one minute may increase customer loyalty'. Just as in the previous case, the same questions need to be answered for each proposition. What makes this case more interesting is that answering these questions is dependent upon propositions found in other web pages. There are two types of dependency occurring. First, the truth of the premise is dependent on the truth of the proposition found in another web document. Second, the truth of the conclusion depends on the truth of the premise and upon some hidden reasoning that led to the deduction. These types of propositions are called 'dependent propositions' in KP.

It is common to find information in one document that is reproduced in another. The reproduction of a proposition in a second document leads to an equivalence relationship between the two propositions, i.e. the truth values of the two propositions are equivalent. However, the relationship is also asymmetric; one proposition is a copy of the other. The copy of one proposition is classified as 'equivalent proposition'. Furthermore, a proposition can be derived using logical deduction. Hence, the truth value of the derived proposition depends on the truth values of its antecedent propositions. This type of derived proposition is classified as 'derived proposition'.

Returning to the example, determining the provenance of the premise requires that we link, in some way, the premise to the proposition in the other web document from which it is copied. The same is true of the conclusion. Minimally, we should link it to its premise, maximally we should link it to the axioms that justify its derivation. These links will also require some type of certification so that we know who created it and whether it is to be trusted.

From these two cases, a number of concepts required for reasoning about provenance emerge:

- Text is divided into propositions. Once so designated, they are assumed to be indivisible.
- A proposition must have a digital signature so that we can guarantee the identity of the creator.
- An assertion is believed to be true, if the information user trusts the person or organization that created the assertion in the corresponding topic.
- As propositions are reused across the web, a link between where it is used and where it came from must be maintained. These links, or dependencies, must be included in the digital signatures with propositions.
- Dependencies can be simple copies or can be the result of a reasoning process. If the latter, then axioms used in the reasoning should also be identified and signed by an acceptable organization.

### 3.2 *Informal competency questions*

The competency of an ontology is defined by a set of questions. In other words, the ontology contains the terms and axioms necessary to answer the competency questions. The competency of Static KP ontology is illustrated by the following questions:

- Is this proposition true, false, or unknown?
- Who created this proposition?
- What is the digital signature verification status?
- Which knowledge fields does this proposition belong to?
- In these fields, can the information creator be trusted?
- Does the truth of this proposition depend on any other propositions? If so, what?

### 3.3 *Terminology*

There are five main classes in static KP ontology: Propositions, Documents, Information Sources, Trust Relationships and Signature Status. From the motivating scenario in section 3.1 and the nature of propositions, a taxonomy of propositions in the Static KP Ontology is constructed as shown in figure 1.

**3.3.1 Propositions.** KP-Prop is the most general concept used to represent propositions in a document. Table 1 defines the predicates for depicting a KP proposition and its attributes.

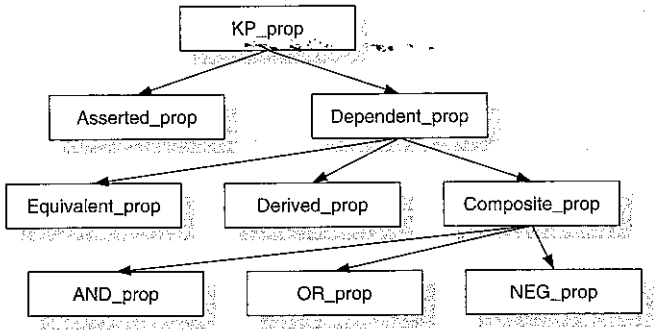


Figure 1. Proposition taxonomy in KP.

Table 1. Proposition-related predicate definitions in static KP ontology.

| Predicate | Description |
|---|---|
| *type(x, 'KP_prop')* | *x* is defined to be a proposition, signified by being of type KP_prop |
| *proposition_content(x, s)* | *s* is the content of the proposition *x*; In html files, the content of a proposition usually is a string; in xml files, the content of a proposition can be an xml element |
| *assigned_truth_value (x, v)* | Proposition *x* has a truth value *v* assigned by proposition creator. |
| *trusted_truth_value(a, x, v)* | Agent *a* trusts that proposition *x* has a truth value *v*. *v* may be one of 'True' 'False', or 'Unknown' |
| *type(x, 'asserted_prop')* | *x* is an assertion and not dependent upon any other proposition |
| *type(x, 'dependent_prop')* | *x* is a proposition whose truth value is dependent upon another proposition; dependent-prop class is further divided into three subclasses: equivalent-prop, derived-prop, and composite-prop |
| *type(x, 'equivalent_prop')* | An equivalent-prop is a copy of and its truth value is the same as the proposition it depends on. |
| *type(x, 'composite_prop')* | Composite-prop's is defined to be the logical combination of its constituent propositions; a composite-prop is divided into three subclasses: neg-prop, and-prop, and or-prop |
| *type(x, 'derived_prop')* | A derived-prop indicates that the proposition is a derived conclusion based on some premises; for example, derived-prop B has dependency-link pointing to composite-prop A, which means that A is a premise of B |
| *is_dependent_on(x, y)* | Proposition *x* is dependent on proposition *y*. *x* is called the dependent proposition, and *y* is called the support proposition |
| *has_same_content(x, y)* | Proposition *x* has the same proposition content as *y* |

**3.3.2 Documents.** To facilitate the determination of the provenance of a proposition, properties of the document in which it appears may need to be considered. For example, knowing who created the document may be important in determining the validity of a proposition within. A document can be any type of file. For the purposes of this paper, we restrict our attention to standard web files such as html files, xml files and xhtml files. Document-related KP predicates are defined in table 2.

Table 2. Document-related predicate definitions in static KP ontology.

| Predicate | Definition |
|---|---|
| *type(x, 'document')* | *x* is defined to be a KP document |
| *in_document(y, d)* | Proposition *y* is contained in document *d* |

Table 3. Information source-related predicate definitions in static KP ontology.

| Predicate | Description |
|---|---|
| *has_infoCreator(x, c)* | KP-prop or Document *x* has infoCreator *c*; here, infoCreator may be either creator or publisher |
| *has_signature(x, s)* | The proposition or document *x* has a signature *s* |
| *has_sig_status(x, v)* | The digital signature verification status of *x* is *v*, where *v* may be one of three status: 'Verified'—the signature is verified successfully; 'Failed'—the signature verification is failed; and 'NoSignature'—does not have digital signature |

Table 4. Trust-related predicate definitions in static KP ontology.

| Predicate | Description |
|---|---|
| *trusted_in(a, c, f)* | Provenance requester *a* (directly or indirectly) trusts information creator *c* in knowledge field *f* |
| *trusted(x, a)* | Proposition *x* is trusted by agent *a*; this means its information creator is trusted by *a* in one of the fields which proposition *x* belongs to |
| *in_field(x, f)* | Proposition *x* belongs to knowledge field *f* |
| *subfieldOf(x, y)* | Knowledge field *x* is a sub-field of knowledge field *y* |

**3.3.3 Information source and signature.** For any document and proposition, its creator can be defined. Along with this, a digital signature and the verification status of the signature can be defined. Assume that a digital signature validation software provides the result of signature verification. Table 3 defines the related predicates.

**3.3.4 Trust relationships.** Earlier we stated that KP is context-sensitive, where the context is trust relations that define whether the provenance requester trusts whose propositions in what topics. A trust relationship in KP is defined as a triple $(a, c, f)$ where the provenance requester (information receiver) $a$ 'trusts' information creator $c$ in a topic or a specific knowledge field $f$; here, 'trust' means that $a$ believes any proposition created by $c$ in field $f$ to be true. Table 4 defines the trust-related predicates.

### 3.4 *Axioms*

In the following, a set of axioms is defined to specify truth conditions of KP-props. Basically, the truth value of an asserted proposition depends on whether the proposition is 'trusted'; the truth value of an equivalent proposition depends on

the truth value of its support KP-prop that the equivalent proposition points to by its dependency link; the truth value of a derived proposition depends on whether the proposition is 'trusted' and whether its support KP-prop is true. In addition, a KP-prop is 'trusted' if the creator or publisher of the proposition is trusted in one of the fields of the proposition, and the digital signature verification status is 'Verified'. Finally, the 'close world assumption' is applied to handle 'not' in this paper.

**3.4.1 Asserted propositions.** An asserted-prop is trusted to have its truth value as assigned, if the asserted-prop is trusted by the provenance requester.

Axiom SKP-1:

*for-all (a, x, v)((type(x, 'asserted_prop') ∧ trusted(x, a) ∧ assigned_truth_value(x, v))*
*→ trusted_truth_value(a, x, v)).*

A KP-prop is 'trusted', if the creator or publisher of the proposition is 'trusted' in one of the fields of the proposition, and the digital signature verification status is 'Verified'.

Axiom SKP-2:

*for-all (a, x, f, c, w)((type(x, 'KP-prop') ∧ has_sig_status(x, 'Verified') ∧ has_infoCreator(x, c)*
*∧ in_field(x, f) ∧ trusted_in(a, c, w) ∧ subfield_of(f, w))*
*→ trusted(x, a)).*

For a KP-prop that has no creator specified, the creator of the document is the default creator of the KP-prop.

Axiom SKP-3:

*for-all (x, d, c)*
*((type(x, 'KP-prop') ∧ (not(exist(c2) has_creator(x, c2))) ∧ in_document(x, d) ∧ has_*
*creator(d, c))*
*→ has_creator(x, c)).*

If a proposition does not have a creator, then the digital signature verification status of the KP-prop is determined by the digital signature verification status of the document.

Axiom SKP-4:

*for-all (x, d, c, v)((type(x, 'KP-prop') ∧ (not (exist(c2) has_creator(x, c2))))*
*∧ in_document(x, d) ∧ has_creator(d, c) ∧ has_sig_status(d, v))*
*→ has_sig_status(x, v)).*

**3.4.2 Equivalent propositions.** The trusted truth value of an equivalent-prop is the same as the trusted truth value of its support proposition, if this equivalent-prop has exactly the same proposition-content as its support proposition has.

Axiom SKP-5:

*for-all (a, x, y, v) ((type(x, 'equivalent_prop')*
*∧ is_dependent_on(x, y) ∧ has_same_content(x, y) ∧ trusted_truth_value(a, y, v))*
*→ trusted_truth_value(a, x, v)).*

**3.4.3 Composite propositions.** The trusted truth value of a neg-prop is the negation of the trusted truth value of the KP-prop it is dependent on.

Axiom SKP-6:

*for-all (a, x, y)((type(x, 'neg_prop')*
*∧ is_dependent_on(x, y) ∧ trusted_truth_value(a, y, 'True'))*
*→ trusted_truth_value(a, x, 'False')).*

Axiom SKP-7:

*for-all (a, x, y)((type(x, 'neg_prop')*
*∧ is_dependent_on(x, y) ∧ trusted_truth_value(a, y, 'False'))*
*→ trusted_truth_value(a, x, 'True')).*

The trusted truth value of an and-prop is 'True' if all its support KP-props are 'True'; and the trusted truth value of an and-prop is 'False' if at least one of its support KP-props is 'False'.

Axiom SKP-8:

*for-all(a, x)((type(x, 'and_prop')*
*∧for-all (y) (is_dependent_on(x, y) → trusted_truth_value(a, y, 'True')))*
*→ trusted_truth_value(a, x, 'True')).*

Axiom SKP-9:

*for-all(a, x)((type(x, 'and_prop')*
*∧ (exist(y) (is_dependent_on(x, y) ∧ trusted_truth_value(a, y, 'False'))))*
*→ trusted_truth_value(a, x, 'False')).*

The trusted truth value of an or-prop is 'True' if at least one of its support KP-props is 'True'; and the trusted truth value of an or-prop is 'False' if all its support KP-props are 'False'.

Axiom SKP-10:

*for-all(a, x)((type(x, 'or_prop')*
*∧ (exist(y) (is_dependent_on(x, y) ∧ trusted_truth_value (a, y, 'True'))))*
*→ trusted_truth_value(a, x, 'True')).*

Axiom SKP-11:

*for-all(a, x)((type(x, 'or_prop')*
*∧ (for-all(y) (is_dependent_on(x, y) ∧ trusted_truth_value(a, y, 'False'))))*
*→ trusted_truth_value(a, x, 'False')).*

**3.4.4 Derived propositions.** The trusted truth value of a derived proposition is 'True' or 'False' as specified, if it is 'trusted' and its support KP-prop (condition) is 'True'. Note that the axioms used to derive the truth value do not have to be included as part of the dependency.

Axiom SKP-12:

*for-all (a, x, y, v)((type(x, 'derived_prop') ∧ trusted(x, a) ∧ assigned_truth_value(x, v)*
*∧ is_ dependent_on(x, y) ∧ trusted_truth_value(a, y, 'True'))*
*→ trusted_truth_value(a, x, v)).*

**3.4.5 Default assigned_truth value.** The default truth value of an asserted or derived proposition assigned by the proposition creator is 'True'.

Axiom SKP-13:

*for-all (a, x, y, v)((type(x, 'asserted_prop') ∨ type(x, 'derived_prop')*
*∧ triple(x, assigned_ truth_value, v))*
*→ assigned_truth_value(a, x, v)).*

*for-all (a, x, y, v)((type(x, 'asserted_prop') ∨ type(x, 'derived_prop')*
*∧ not (triple(x, assigned_ truth_value, v)))*
*→ assigned_truth_value(a, x, 'True')).*

**3.4.6 Default trusted_truth value.** The default trusted truth value of a proposition is 'Unknown'.

Axiom SKP-14:

*for-all (a, x, v)((type(x, 'KP_prop')*
*∧ not (trusted_truth_value(a, x, 'True')) ∧ not (trusted_truth_value(a, x, 'False')))*
*→ trusted_truth_value(a, x, 'Unkown')).*

## 4. Implementing KP

In order to apply KP in practice, information creators need to annotate web documents with KP metadata, users (provenance requesters) need to define their trust relationships, and a KP reasoner is needed to conduct provenance reasoning on annotated web documents.

The motivating example introduced in section 3 is used to illustrate how to determine the provenance of propositions as shown in figure 2. In order to facilitate the annotation of web documents with KP metadata and define trust relationships,

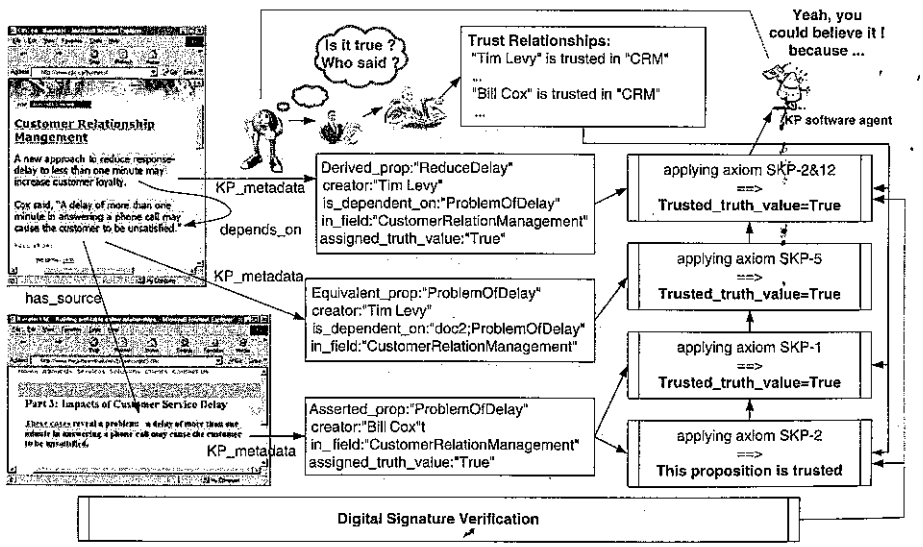


Figure 2. Example to illustrate how to determine the provenance of web information by KP.

we have defined a KP markup language in RDFS (Resource Description Framework Schema; see http://w3.org/rdf/). The following is an example of how to annotate a html file with KP metadata.

***Document1: http://www.crm-examples.com/ex1.html***

```
<HTML xmlns="http://www.w3.org/1999/xhtml"
dsig = "http://www.w3.org/2000/09/xmldsig#"
kp = "http://www.eil.utoronto.ca/kp#" >
<HEAD >
<kp:Document rdf:about = "http://www.example.com/ex1.html" creator = "Tim
Levy"/ >
</HEAD >
<BODY >
<kp:Derived_prop rdf:id="ReduceDelay" is_dependent_on = "#ProblemOfDelay"
in_field = "Customer Relationship Management" >
The new approach to reduce response-delay to less than one minute may increase
customer loyalty.
</kp:Derived_prop >
<kp:Equivalent_prop rdf:id="ProblemOfDelay"
is_dependent_on = "http://www.crm-exam.net/ex2.html#ProblemOfDelay" >
A delay of more than one minute in answering a phone call may cause the customer
to be unsatisfied.
</kp:Equivalent_prop >
<Signature ID="ex1" >
<SignedInfo >
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/ >
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/ >
<Reference URI="#ReduceDelay" >
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/ >
<DigestValue > j6hm43k9j3u5903h4775si83 </DigestValue >
</Reference >
<Reference URI="#ProblemOfDelay" >
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/ >
<DigestValue > g79lk20rjf023rr032kr93kjr </DigestValue >
</Reference >
</SignedInfo >
<SignatureValue > M459ng9784t . . . </SignatureValue >
<KeyInfo >
<X509Data >
<X509SubjectName > . . . </X509SubjectName >
<X509Certificate > MIID5jCCA0+gA . . . lVN </X509Certificate >
</X509Data >
<KeyInfo >
</Signature >
</BODY >
</HTML >
```

In this sample document, there are two KP-props: (1) a 'derived-prop' with id 'ReduceDelay', entailed by KP-prop 'ProblemOfDelay' and (2) an 'equivalent-prop' with id'ProblemOfDelay'. Let us assume that the digital signature (in XML-Signature syntax) of these two propositions is verified successfully, and Tim Levy is trusted in the field of 'Customer Relationship Management' by the person requesting KP. Therefore, according to Axiom SKP-2, the derived-prop 'ReduceDelay' will be trusted. But to determine whether its truth value is to be trusted, we have to determine whether the support proposition it depends on has a trusted_truth_value of true. The proposition 'ProblemOfDelay' is an equivalent proposition, its truth depends on its source, another proposition in another web document: 'http://www.crm-examp.net/ex2.html#ProblemOfDelay'.

***Document2: http://www.crm-examples2.net/ex2.html***

```
<HTML xmlns="http://www.w3.org/1999/xhtml"
dsig="http://www.w3.org/2000/09/xmldsig#"
kp="http://www.eil.utoronto.ca/kp#">
<HEAD>
<kp:Document rdf:about="http://www.crm-exam.net/ex2.html#ProblemOfDelay">
<kp:creator>"Bill Cox"</kp:creator>
</kp:Document>
</HEAD>
<BODY>
<kp:asserted_prop rdf:id="ProblemOfDelay" in_field="Customer Service">
A delay of more than one minute in answering a phone call may cause the customer
to be unsatisfied.
</kp:asserted_prop>
<Signature ID="Bill-ProblemOfDelay"> ... </Signature>
</BODY>
</HTML>
```

In document 2, 'ProblemOfDelay' is an 'asserted-prop'. Its trusted truth value depends on whether it is trusted. The creator of the document is 'Bill Cox'. Assume the digital signature is verified successfully, and the requestor trusts Bill in the field of 'Customer Service'. Then, according to axioms SKP-2 and 1, 'ProblemOfDelay' has a trusted_truth_value of 'True'. Consequently, the equivalent proposition 'ProblemOfDelay' in document 1 also has a trusted truth value of 'True' by using axiom SKP-5, and finally, the derived proposition 'ReduceDelay' has a trusted truth value of true by using axioms SKP-2 and 12. We have implemented the KP reasoner with Prolog. The system can infer the truth of any KP-prop.

## 5. Dynamic KP

This section extends static KP into Dynamic KP to address the problem of how to determine the truth values of web propositions that change over time.