NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 — "Semantic-based knowledge and content systems"

D4.3.2 Conceptual model of trust and access control for the "NeOn Scenario"

Deliverable Co-ordinator: **Martin Dzbor**

Deliverable Co-ordinating Institution: **The Open University (OU)**

Other Authors: **Dnyanesh Rajpathak (OU); Aldo Gangemi (CNR); Simon Schenk (UKO-LD); Noam Bercovici (UKO-LD); Claudio Baldassarre (FAO)**

We propose an approach to modelling various situational aspects that may lead a cognitive agent (usually, a human being) to make an assertion about the trust relationship(s) in a given situation. The deliverable perceives trust as one of the contextual modifiers that may be attached to ontological elements. The approach we take is agent-driven – i.e., trust assertions are declared by cognitive agents, often ontology users or developers. In our opinion, this is a natural way of treating trust, since it complies with the philosophical view of trust attributed to a system of beliefs. However, this assumption does not prevent our model from expressing multi-way trust relationships (agent – agent, agent – object, and even object – object).

Document Identifier:	NEON/2008/D4.3.2/v1.0	Date due:	August 31, 2008
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	August 31, 2008
Project start date	March 1, 2006	Version:	v1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

2006–2008 © Copyright lies with the respective authors and their institutions.

NeOn Consortium

This document is part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta}@open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 11 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.ump.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SL–1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 665 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier, France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield, United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Kolenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Marino della Battaglia 44 – 00185 Roma-Lazio Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed writing parts of this document:

- The Open University (OU)
- University of Koblenz-Landau (UKO-LD)
- Food and Agriculture Organization of the United Nations (FAO)
- Consiglio Nazionale delle Ricerche (CNR)
- iSOCO, S.A. (ISOCO)

Change Log

Version	Date	Amended by	Changes
0.1	20-06-2008	Martin Dzbor, Aldo Gangemi	Initial trust models sketches
0.2	20-08-2008	Martin Dzbor, Dnyanesh Rajpathak	Description of core modelling decisions for trust metamodel
0.3	27-08-2008	Simon Schenk, Claudio Baldassarre	Motivation for trust models
0.4	10-09-2008	Simon Schenk, Noam Bercovici	Access rights models and trust orders
0.5	10-09-2008	Dnyanesh Rajpathak	Gaps of existing trust models
0.6	16-09-2008	Martin Dzbor	Instantiations of generic trust metamodel with worked examples
0.7	22-09-2008	Martin Dzbor, Aldo Gangemi	Cleanup, introductions, etc.
0.8	10-10-2008	Jose Manuel Gomez, Guilin Qi	Review of deliverable
1.0	13-10-2008	Martin Dzbor	Review comments, cleanup, finalization

Executive Summary

In this deliverable we propose an approach to modelling various situational aspects that may lead a cognitive agent (usually, a human being) to make an assertion about the trust relationship(s) in a given situation. The deliverable perceives trust as one of the contextual modifiers that may be attached to ontologies and ontological elements; essentially, with a purpose of explicitly disclosing some emotional or rational processes inside an agent that are 'manifested' as trust.

The approach we emphasized here is agent-driven – i.e., trust assertions are made, declared by cognitive agents, often ontology users or developers. In our opinion, this is actually a very natural way of treating trust, since it complies with the philosophical assumptions of trust being attributed to a system of beliefs. However, we should note that the fact that a declaration is made by a cognitive agent does not prevent our model from expressing multi-way trust relationships. These include agent – agent, agent – object, and even object – object assertions. The actual meta-model is available online from http://neon-project.org/ontologies/2008/10/trust-metamodel.

Another contribution of this deliverable is a definition of a conceptual model to operationalize the theoretical proposals made in deliverable D4.4.1 in terms of access rights management. An OWL version of this fragment of the model is accessible online at http://isweb.uni-koblenz.de/ontologies/2008/10/accessRight.owl.

Nonetheless, the authors want to see the model presented in this deliverable as a work in progress. In particular, we intend to follow up on proposing and formalizing additional contextual factors to fit the proposed generic framework. Of a particular importance for our demonstration effort in the next few months is the merger of access control model into the contextual 'branch' of the trust ontology. Another important area where we intend to pilot the proposed approach is capturing trust assertions from the actions occurring during an ontology editing workflow – this particular application area also needs to be formally expressed as an extension of our generic 'Context' concept.

Since both tasks T4.3 and T4.4 proposed to deliver a formal framework for capturing trust and access rights respectively, in our latest Description of Work we decided to collate the two strands of work into one deliverable. This decision has been further motivated by the fact that there is an implicit relation between trusting and access (especially, access delegation), so the deliverable D4.3.2 has been collated and co-edited to reduce the unnecessary repetition of shared conceptualizations.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this deliverable we further develop the notion of trust and access control with respect to networked ontologies, which we started in the earlier report D4.4.1 [DKG$^+$07]. By ontology in this deliverable we mean a schema defining concepts, properties, restrictions, etc., which might be (optionally) populated with instances. By a network of ontologies we mean several ontologies or ontological modules that are connected with explicit, semantically meaningful relations, such as for example, version, similarity, complementarity, etc. – as defined in respective NeOn report [HRW$^+$06]. The purpose of our discussion in the subsequent chapters is threefold:

- to formally define the key aspects of the terminology that is typically used in connection with managing trust and access;
- to propose a model that would cover several perspectives on and implications of controlling trust and access in ontologies that were considered in the previous reports – our object from this angle is to realize advantages of an access control model and propose the ways to transfer different models to the world of networked ontologies;
- to serve as a potential foundation for a further, more formal development of trust and access rights metadata as a part of NeOn model and/or ontology metadata vocabulary – our objective here is to present access control as an application making use of the NeOn framework for expressing and representing context [QP08].

The document is structured in the following way: We start with introducing the subject and refreshing our motivation for this work in this section. Next, we briefly summarize the key aspects of treating access control as a type of context. Then, in chapter 4 we look at some specifics of access control model for ontologies in general, and of *networked* ontologies in particular. Namely, we consider how features like the distributed and collaborative nature of ontologies can be reflected in the access control models, and how these considerations affect accountability, transparency and legacy dependencies. Next, we discuss the realization of the access control model and its implications in chapter 6 – again, based on the use of contextual ontologies and OMV. Finally, we offer a few worked scenarios showing possible ways of operationalizing the theoretical consideration and we conclude then.

Before we continue, let us start with some motivation. Since ontologies are (by definition) conceptualization that are supposed to be shared, the question of why should we then think of restricting this shareability is more than appropriate.

1.1 Motivation for trust and access control metamodel

In the literature we can find numerous views and proposals for managing relationships between two parties, whether they are in a form of capturing the notion of trust or the notion of access. While in NeOn we started investigating the two issues independently, soon in became apparent that

- The notion of trust is very tightly interwoven with the notion of access control, especially, as we have shown in [DKG$^+$07] the act of *delegating access rights to another party* carries an implicit statement about the mutual trust between the two parties involved.
- The notion of trust is usually addressed in different applications rather opportunistically and it carries a substantial *implicit element*; i.e., one can identify many different types of trusting relationships, different sources, and hence, different implications for any operational framework.
- Since trust is the key carrier for capturing the information about the relationships between two parties, we found the existing formal models involving trust as rather incomplete and partial.

Hence, this deliverable was conceived as a potential glue enabling us to relate (if not completely unify) the different perspectives and views on the notion of trust that became used in the NeOn project. In the discussion and test cases for several deliverables we have seen a range of terms that to some extent allude to trust; e.g., a confidence of an agent in another agent, a preference for ranking purposes and/or in selecting resources, a provenance of metadata or of a discovered source, etc.

First, we therefore identify a few 'types' of trust that informed the definition of the metamodel proposed in the subsequent parts of this deliverable. Majority of existing work perceives trust as something coming from a person or an agent and aimed toward another person or agent. A typical example of this view is statement "*John trusts Jane*". An opposite view can be proposed whereby an object other than a person or an agent is personified and trust relationship is attached to such a target – a typical example of this statement is "*I trust axiom X in ontology O*". Thus, the first divergent criterion for handling trust is the capability to enable different types of target toward which a party may express their relationship of trust.

Next, although trust is tied to a particular person or agent, it is very rarely expressed in such a simplistic form as the above paragraph suggests. More often than not, one wants to express some "*mediating factor*" that circumscribes a particular trusting relationship to a specific scope, time, situation – or in general to *a particular context*. Examples of this can be seen in the following two statements: "*I trust Aldo is an expert in ontologies because of formalizing DOLCE*" or "*John trusts TAP ontology for the purpose of data classification*". Hence, some form of contextualizing the trust relationship is needed to be explicitly representable in our metamodel.

Third, trust is often seen as something declared by an individual; however, such a view contrasts with such notions as karma or trustworthiness in the blogging world, whereby the author of the trust statement is different from the individual about whom the statement has been made. A typical example of this indirect assertion about trust can be seen in the statement: "*Trustworthiness of Joe Bloggs is a sum of all the supportive reviews he has got in Slashdot*". Hence, we believe it is important to capture this difference between a party participating in the trust relationship and the asserter of the trust statement.

Next, trust is often presented as an outcome or a corollary of other relationships – for example, the notions of object provenance (origin), access delegation and confidence are conceptually different but all may carry an implicit statement involving some trust relationship. What is even more important is the fact that parties involved in these other relationships do not have to be persons or intelligent agents. One can easily attach an implicit notion of trust between an ontology O and a module M that is imported into the ontology by means of an owl:imports property – thanks to the semantics of the imports property, ontology O inherits all the assertions made in the imported module, which can be interpreted as "*Ontology O trusts module M (because of . . .)*" Hence, the metamodel for trust shall be flexible enough to allow for different entities participating in a trusting relationship, and at the same time, shall be sufficiently rich and expressive to allow the clarification of the intended meaning.

Why does it all matter? The key issue for managing trust is not in actually allowing its declaration and representation. A more critical question is what can be done once a trust assertion has been made, what is such an assertion useful for. One typical operation one may want to explore is trust inheritance, e.g., a transitive trust. Without more expressive means to capture the intended meaning of the trust as a relationship, its context, author, etc. it is formally impossible to make any meaningful differences e.g., between statements "*Joe trust Ann to drive his car*" and "*Ann trusts time ontology because it was designed by Jim*"

1.2 Relationships with other work in NeOn

Work package 1: networked ontologies

As will be presented later in the deliverable, we base our proposition of considering access rights on the capacity to identify within an ontology a sub-set of ontological entities that need to be treated (for whatever reason) differently than other parts of the ontology. In work package 1, the capacity to identify a sub-set within a larger ontology is investigated in the scope of *modularization*. An ontology module is seen as a tuple of imported ontologies or *partial* ontology partitions, certain import and export interfaces, and a set of mappings. While the actual research into defining and describing modules is carried out in WP1, there is an interaction with this work package.

The first possible interaction is that the level of access or authority may serve as a criterion for partitioning a larger ontology into smaller chunks. For example, in one of the motivational examples we suggested hiding certain aspects of a (populated) ontology from certain users. In order to achieve this, there is a need to partition or to modularize the ontology so that the individual blocks are clearly identifiable for the purposes of asserting something about the access rights.

Another possible interaction, which however, needs to be considered in a broader context, is the implication of giving subjects access to some modules of a larger ontology but not to others. Assume a situation where ontology comprises among other content three modules, M_1 through M_3. By 'other content' we may mean concepts or restrictions that relate entities from two different modules, so do not belong in either of them. These entities may form yet another module, 'the rest' of the ontology and we have the ontology fully covered. If we now remove (say) module M_1 due to access limitations but retain M_2 and M_3, what will happen to 'the rest'? Intuitively, some parts of 'the rest' should be included in the access controlled ontology – those that depend on M_2 and M_3, but not on M_1. However, these were not necessarily defined by whoever partitioned the ontology earlier. Moreover, there may be several different 'rests' dependent on which modules are permitted for a particular user. How to treat these dynamic consequences of certain modules being inaccessible is subject for further discussions with work package 1 and also NeOn use case partners.

Work package 2: collaborative aspects of ontology engineering

Access control rarely applies to the environments with a single user who creates, edits, publishes and consumes the outcomes of his or her engineering process. However in the open environment comprising both, public Web and private corporate Intranets, there will be many users engaged in using, but often also engineering ontologies. Indeed the notion of networked ontologies assumes several ontologies that may be developed by different stakeholders, sometimes independently, sometimes in a collaborative manner.

As the engineering of networked ontologies becomes more widespread it would be more principled. Hence, there would emerge subjects who would inherently have different access needs. For instance, a proposer may suggest a rough schema for a new problem; whereas an engineer may elaborate on it in terms of adding sub-concepts, editing it, etc. Then a domain expert may only need to annotate the propositions and assertions made by others in terms of their scientific plausibility. Furthermore, a librarian skilled in validating schemas may carry out ontology validation and approval, which may imply access in terms of retaining or deleting entities but not otherwise amending them.

All these aspects are to some extent investigated in work package 2; in particular in its models of collaborative engineering and workflows. Wokflows and business processes are especially interesting because (i) they are widespread in the organizational context, and (ii) they are used to control which subject carries out which action(s) on a shared artifact, and where does s/he pass the outcomes of that action. Intuitively, the notion of workflows and business processes lends itself almost seamlessly to the need for controlling access. Workflows and the subject assignment to its component stages imply certain authorities and these, in turn, imply certain levels and types of access.

Whilst workflow management is primarily the theme for WP2, our work on access rights intends to contribute to those developments in terms of safeguarding the content of the collaboratively developed ontologies in terms of existing processes and workflows. On the other hand, the access control must not be so stringent as to prevent any collaboration among subjects. In all organizations there are formal, process-driven collaborating teams, and there are more informal, ad-hoc, bottom-up emerging communities of collaborators that work together to address an unforeseen problem. The tension between more prescriptive role- and workflow-driven access on one hand side, and more liberal authority and access delegation is one of the themes that is sketched in this deliverable. However, it is also the theme that requires further investigation and collaboration with WP2 to better understand its principles and consequences.

Nevertheless, the collaborative aspects of engineering and networking ontologies are subject to a dedicated discussion (see content of sections **??**) from such angles as access granularity, inheritance and delegation of authorities, revocation of authorities, relationship to groups and organizational roles, etc.

Work package 3: contextual aspects of networked ontologies

With respect to work package 3 and its focus on contextual aspects and context awareness, the role of access rights and access control is less obvious. Nonetheless, taking a broader perception of what a context might comprise of, it is possible to argue that access rights are one of many context-determinants. In other words, access rights define not only what a particular subject is permitted to *do* with a given ontology, but also what s/he is permitted *to see* from that ontology. Obviously, this argument is subject to willing and intending to deny the authority to read a specific module of a larger ontology to some subjects.

Assuming there are such needs, then the resulting ontology would obviously miss some of the building blocks that were originally included in it – as it was already mentioned earlier in connection with our positioning to work package 1, above. While module algebra may easily deal with this issue on the syntactic level, what are the contextual implications of omitting module M_1 from a larger ontology O and only presenting O as comprising of M_2, M_3, and 'the applicable rest'? Are we still talking about the same ontology, only adapted to the *context of a particular subject's authority*? Or is this an entirely new conceptualization of the reality, which must not be, under any circumstances presented under the same label as the original O?

These issues are not cleared formally yet; nevertheless, they suggest an interesting challenge that has many parallels with our daily experiences. Take a simple example of using software products: we all get access to the executables of MS Word (for sake of argument), so the content of folder `../Microsoft Office/Word/*` is what we know as "Microsoft Word" bundle. Make a hypothetical journey to Redmond, Washington (HQ of Microsoft Corporation and its main development center), and the same bundle includes sources and content of numerous pictorial add-ons. So, two different groups of subjects see the same bundle in two entirely different constellations, each affording certain actions and disallowing others.

Move the same parallel to other software, e.g. operating systems, and we get "Redhat Linux X.y" with add-ons such as *. . . Enterprise edition*, *. . . Fedora*, or *. . . Home user*. These are essentially access-limited constellations of the same (in principle) underlying design, codebase and modules. Different users would get different modules but there are sufficient similarities or mappings for them to perceive the differences as *editions* of the same system (in our case, we make a parallel by replacing 'system' with 'ontology').

Work packages 7 and 8: use case

Having studied use case requirements and participating in numerous design discussions with the use case partners, there is a certain amount of need for a capacity to manage access to networked ontologies – at least within the organizations. However, most of these requirements are not well defined, so in places, access control is seen as an integral part of workflow definition and enforcement. In other places, access control is seen more as an equivalent of organizational roles being enforced in a collaborative environment.

Moreover, access control in the use case requirements is implicitly confined to restricting editing and publishing authorities. Hence, one of the objectives of this deliverable is to offer use case partners a slightly broader view on the issue of access management, so that it is possible for them to choose and/or refine which particular models of access control are indeed most suitable for their particular setting, their application and their users. Without this additional information, the use case partners make decisions about their requirements based on incomplete knowledge. And this knowledge, understandably, comprises what they are currently experiencing with respect to file-based access control. We will argue later that the view “*ontology as a file*” is not very useful, especially so with networked and modular ontologies.

2006–2008 © Copyright lies with the respective authors and their institutions.

Chapter 2

Gaps in Trust Modelling – Overview

The notion of trust is central to the interactions between people and systems, and it establishes a common basis to execute a transaction towards achieving a specific goal. Traditionally, in computer science the notion of trust is typically embedded in a subjective nature of human behaviour. Marsh [Mar94] provides one of the earliest attempts to define trust at a generic level from computational point of view. Various attempts have been made at developing formal models of trust; e.g., PolicyMaker [BFL96], KeyNote [BFK98], REFEREE [CFL$^+$97], Abdul-Rehman [ARH97], Daniel Essin [Ess97], Audun Josang [Jos96], Poblano [CY00], Free Haven [Din00], SULTAN [GS03], TERM [BZ97], SECURE [CNS03]. Also several trust ontologies have been developed [Vil05, CDH07, ZJ07, GPH03], which provide a common vocabulary that would enable different parties involved in the trust-related transaction to communicate their beliefs so that a goal of a transaction is achieved successfully. These trust ontologies also enable heterogeneous systems with different trust models to share trust-related information between them, and at the same time allow seeing how such trust relations are established between them.

In this chapter we provide an overview of these past attempts of developing the formal models of trust. In this review we concentrate on understanding the strengths and the weaknesses of these proposals.

2.1 A formal model of trust in dynamic network

Carbone et al. [CNS03] provide a formal model of trust informed by a Global Computing scenario. In their model the notion of trust is formalised as follows: “Trust involves entities, has a degree, is based on observations, and ultimately determines the interaction among entities” [CNS03].

The notion of entities is central to this framework, and is referred to as principals. The trust value represents a degree of trust that takes a simple value, e.g., *trusted, distrusted*, or a structured value, in which the first value represents an action, such as access to a file, while the second value represents the trust level that is related to the trust action. Each principal involved in a trust management has a ‘trust box’ associated with it. The trust box is realised as an object module that contains trust management operations and data. The notion of a trust box is used to represent principal’s mutual trust formalised as a function to each pair of principals with a trust value, t.

This model also provides a way to realise third party assessments, i.e., delegations of trust when trust originates from unknown source. In line with this proposal, a principal’s trust is defined by a policy and according to this formalism each principal has a local policy, which contributes in establishing a global trust. Moreover, the notion of policy provides a way of computing trust information based on the belief’s of all the principals. This framework also assumes that principals typically only have partial knowledge to their surroundings and therefore of the trust values. To deal with such approximation the model introduces the notion of approximate trust values and uncertainties, and also allows different types of ordering on trust values, e.g., complete and partial trust value ordering and trust information ordering. The trust value ordering provides the degree of trustworthiness such as absolute distrust and absolute trust. The trust information ordering indicates the degree of precision of trust information when no knowledge is present or greatest level of certainty is present.

This model also provides a way of realising a ‘context dependent’ trust that allows one to represent the situation in which a certain principal is trusted, e.g., trusting principal A about restaurants does not extend the notion of trust to the principal A about sailing. Thus, context is partially supported in terms of scoping the statements and values. This model takes into account interval construction to add uncertainties to trust lattice, which can be used to guide the design of policy language. The key technical contribution of this model is that it introduces bi-directed structures $(T, ?,)$ in which ‘?’ measures the information contents of data which is needed to compute the fixpoint of mutually recursive policies, while the trust ordering ‘?’ measures trust degrees and is used to make trust-informed decisions.

The main limitation of this model is the fact that it conceptualizes trust for one specific perspective and one application domain of distributed dynamic networks, e.g., Global Computing and it is difficult to realise how this model can be used in application domain that are not necessarily related to to this domain. Moreover, it is crucial to realise that this model is denotational in nature and not an operational one. And because the model is denotational in nature, it does not provide a way to calculate the notion of trust information. Finally, contrary to this model our aim is to provide a trust ontology metamodel, which is machine-processable and at the same time it also provides a way to represent the notion of trust at a generic level without subscribing to any application domain.

2.2 Poblano

Poblano [CY00] provides a decentralised trust model on a peer-to-peer foundation. In contrast with other trust models, e.g., Free Haven [Din00], in which the degree of trust is calculated with parameters like performance, reliability and honesty, the Poblano provides a way of calculating trust in a situation in which a peer is deprived of an opportunity to authenticate. To this end, in Poblano trust has several factors that are associated with the group’s interests, or group content relevance, and it introduces the notion of “codat trust component”, which is different from the traditional trust component. The following three classes are central to the model: *CodatConfidence*, *PeerConfidence*, and *Risk*. The class CodatConfidence evaluates the codatTrust component according to a keyword. It consists of four element – keyword, codatID, local flag for making local or remote codat, and confidence value. The notion of confidence value consists of popularity and relevance matrix. The notion of popularity increases incrementally each time the codat is requested. The relevance is in the range of (-1, 0, 1, 2, 3, 4). The class PeerConfidence provides a way of evaluating the codat peer trust component according to a keyword and it consiste of 3 elements – keyword, peerID, and confidence value. Finally the class Risk provides a way of evaluating peer’s risk trust component. It consists of following elements – peerID, integrity of the codat, accessibility and performance.

This model provides a way of representing degree of trust among peers. The peer can have one of the following six values to represents its trust: Distrust (-1), Ignore (0), Minimal Trust (1), Average Trust (2), Good Trust (3), and Complete Trust (4). The peer confidence represents carrier information to be used for weighing, whereas the codat confidence is the information to be propagated. Moreover, the notion of a cooperation threshold is introduced in Pablano which makes trust values more meaningful to the users. For instance, if the PeerConfidence is higher than the cooperation threshold then the peer is considered to be cooperative in nature, otherwise it is considered to be risky. Thus, the main contribution of this approach is the multi-value assessment of trust; the drawback, on the other hand, is the focus on agent-to-agent trustworthiness, to some extent ignoring other target entities.

At the core of the Poblano model is the ability to create and distribute public keys which creates peer-to-peer distributed model that takes into account security related issues like privacy, authentication, integrity and non-repudiation. Similarly as in [CNS03], this model has a limited applicability because it is only formalized to provide trust model in peer-to-peer distributed network. Our aim is to provide a more generic ontological metamodel of trust that provides a way of expressing the notion of trust for different application domains and allow for different perspectives. In contrast with Carbone [CNS03] the notion of context is missing from this model. In our opinion, the notion of context is central to the trust framework because it provides a clear basis to realize the situation for which the trust related assertions are made.

Moreover, in this model different trust values that peers can have are hard coded, but in contrast with this model our aim will be to provide a generic notion of ‘trust value’, which can be specialized according application domain in which the trust model is instantiated. Finally, in contrast with Poblano, our aim is provide a generic ontological metamodel of trust which can be shared and reused in different application domains.

2.3 REFEREE

Chu et al. [CFL$^+$97] propose a system called REFEREE (Rule-controlled Environment For Evaluation pf Rules, and Everything Else), which is a trust management system for web applications. REFEREE provides an environment in which policies are evaluated for its compliance with web applications.

The key primitives of REFEREE are – programs, statement lists and Tri-values. The program takes as an input an initial statement list along with additional statements. It may invoke another program that returns true or false based on the statements are sufficient to infer compliance or non-compliance. The statement lists are collection of assertions expressed in a specific format. Finally, the Tri-values take the form of true, false, or unknown. In REFEREE trust management system the action may take place because sufficient credentials exists for the actions to get certified as an approved one, it may be disapproved in the light of insufficient credentials, and unable to find sufficient credentials either to approve or disapprove the action. While dealing with the last case, the trust management system returns ÔunknownÕ value because the actions can be considered to be neither ‘true’ nor ‘false’.

In REFEREE, the statements are a two-element structure consisting of content and the context about the content. The notion of context, here, provides a way of interpreting the content whereas the interpretation of context is an affair between the calling application and REFEREE. Finally the key feature of the REFEREE trust management systems is that it places all dangerous operations under policy control instead of having to make arbitrary decisions.

Our aim is to engineer a generic ontological metamodel to formalise the trust which can be instantiated in different application domains. In contrast with our vision of engineering ontological framework, the REFEREE provides a trust management system to certify whether certain action can be considered as approved or disapproved. In our viewpoint, the REFEREE trust management systems is shallow in nature because it fails to provide detailed formalism of the key concepts associated with their framework, e.g. trust, assertions. For instance, REFEREE statements are assertions to determine the compliance of actions, but no trust value is associated with these assertions to indicate the level of trust attached to such assertions. As it can be imagines that usually not all assertions represent equal trust weight but while some assertions indicate high level of trust because it has high trust value and other assertions indicate low level of trust due to low trust value. Moreover, the REFEREE trust management framework only provides a partial representation of the key concepts associated with trust management such as trust, context, assertions, but their fine-grained categorisation is missing. For instance, the notion of context can be categorised into explicit and implicit context. In REFEREE statement lists are simply unordered list of REFEREE statements but in our point of view it is crucial is distinguish between ordered and unordered lists, and that of partially and total ordered lists to engineer clean reasoning engine. Finally, REFEREE is not an ontological model, which can be reused or shared among different application domains in a machine-readable fashion.

2.4 Formal logical framework of trust

Demolombe provides a framework of modal logic in order to provide the definitions of different types of trust. Moreover, this framework also provides a way to reason about trust. In this framework, the notion of trust is defined as a mental attitude of an agent with respect to some property held by another agent. As a result, the trust is a kind of interaction which involves minimum two agents. This framework takes into account following three types of problems associated with trust:

1. to define the facts that support trust, e.g. observations, reputation, or analysis,
2. to define a set of correct rules to derive consequences of a set of assumptions about trust, and
3. to make use of the information about trust to take decisions.

The notion of agent is central to this framework which is then classified into a human agent and an artificial agent. An agent has a belief which describe an attitude of an agent about some property of an agent. This framework distinguishes between belief, strong belief and knowledge. The key contribution of this work is that it provides a clean definitions defining epistemic properties of trust which consists of sincerity, cooperativity, credibility, vigilance, validity, and completeness. By making use of the epistemic properties it provides formal definitions of trust such as Tsinc a,b(p), Tcoop a, b(p), Tcred a,b(p), Tvigi a,b(p), Tval a,b(p), and Tcomp a,b(p). This can be read as trust of the agent a with regard to b about p for the property prop is denoted as Tprop a,b(p). Moreover, this formal model also provides axiom schemas and inference rules of propositional logic and logical properties. The deontic and dynamic properties about trust are also considered in this model to analyze possible links between the actions performed by the agents and the obligations to perform on the other hand. The following types of links are considered – obedience, laziness, active, honesty, and ability. In order to deal with the practical applications, this application introduces the notion of graded trust that allows to represent heterogeneous qualitative levels of trust, e.g. more or less. Moreover, the notion of context in this framework is realized in terms of the trust with related respect to topics. For example, the agent a may trust agent b with respect to the validity of all the propositions that are made that are relevant to topic, “best italian restaurants”. Finally the notion of conditional trust can be used in many situations where an agent trusts another agent only in particular circumstances.

In contrast with the frameworks described in the previous sections, this framework provides a classification of the notion of agent into human agent and artificial agent. However, it fails to provide a more abstract classification into cognitive agent and sentient agent depending on whether or not an agent conceive plans and makes decisions towards actions. Moreover, while this framework provides an initial classification of an agent, but it does not provide any indication to represent different roles that agent may play in heterogeneous application domains, e.g. an agent has a role of an author which posses some belief in order to assert trust statements. Although this framework provides a notion of context, no indication is provided about how the contextual features that may be associated with context can be used to describe a situation. This framework provides rigorous distinction between epistemic properties about trust but it fails to provides similar type of distinction while formalizing key notions associated with trust model, e.g. context, trust statement (direct trust statement and indirect trust statement), trust (direct and indirect trust, explicit and implicit trust). In this trust framework the notion of graded trust is used to represent the qualitative measure of trust, but it fails to provide any indication about quantitative measure, e.g. trust value, either as an integer or a vector. Finally, similarly with other frameworks described in the previous sections, this is not an ontological model which can be reused and shared in different application domain in order to capture trust related issues.

2.5 Trust networks on the Semantic Web

This framework [GPH03] subscribes to the techniques developed in the application domain of social network analysis and applies these techniques to the issues of trust in the semantic web. At a coarse-grained level of abstraction the notion of trust in this project is addressed as credibility or reliability in a human sense. This framework takes a position by which there are many measures of trust within social network and a common way of realizing trust here is simply to know someone. In the context of the semantic web trust requires users to explicitly specify their beliefs about other users.

This framework takes Friend-Of-A-Friend schema as an input as this describes the interlink statements about about a person such as name, email address, and homepage along with the information about people he knows. It extends foaf:person schema and enables users to specify the level of trust for people they know. This approach provides a way of specifying the level of trust on the scale of 1–9 that corresponds to – distrusts completely, distrusts highly, distrusts moderately, up to trusts absolutely. The minimum condition of this framework to exhibit the level of trust for a specific topic is that it must have at least two agents involved in it. The scope of a specific topic represents the (partial) notion of context and the trust levels are typically associated between any two agents w.r.t. a specific topic. The trust relations are established between two agents and the network of such trust relations is considered as a directed, weighted graph. This framework also provides a way to compute trust based on whether two nodes in a graph have a direct edge between them or not. To determine how much a source trusts the target, and to give recommendations, several basic calculations are made over the network – maximum and minimum capacity paths, maximum and minimum length paths, and a weighted average method. This framework also provides a way to access trust networks as a web service. The web service interface enables users to supply their own algorithms in order to calculate trust.

One of the main limitations of this work is that it formalises the notion of trust from a limited perspective, i.e., as a agents as 'foaf:person' specifications. While personification is an important component to formalize the notion of trust it restricts the applicability of this trust model to the situations which involve two human agents. Moreover, because this framework formalises trust from a narrow viewpoint, it fails to take into account more generic specifications of agents such as cognitive or sentient agents, and different roles these agents may play in heterogeneous application domains. More importantly, the trust relations in this framework are related to the association that takes place among two agents in a specific context, which is represented explicitly. However, it fails to support the situations in which the context among two agents involved in a given situation is more complex and cannot be specified as a topic. For instance, when a human-agent makes use of a software application over a certain period, this may represent the fact that the agent trusts the software application. Here, context of using the software application by a human agent is not explicitly represented but rather implicitly observed. This framework provides a way to restrict trust to a specific topic which essentially provides context to the interaction among two agents, but no indication is given about how the notion of context can be generalized. For instance, in the statement "Joe trusts Blogs highly regarding research topic X", we need to capture the context of Joe trusting Blogs in a *specific* research topic. This specific research topic may imply the trust into concrete *contextual features*, like author publishing in that research topic, researchers who carried out seminal work in the research topic during a certain time period, or managers who attracted funding to undertake projects in that research topic. Finally, this model does not take into account the notion of access as a form of security as a part of trust framework.

2.6 SchemaWeb

SchemaWeb[1] is a directory of RDF schemas that are represented in RDFS, OWL, and DAML+OIL schema languages. It consists of several ontologies and the trust ontology is also part of the directory formalized in OWL. The notion of trust in this ontology is realized as a way social agents cope with the uncertainties in everyday business. The trust ontology primarily subscribes to the way social networks function on the web in order to represent the explicit trust ratings. Moreover, it also looks at the matrix in order to infer the notion of trust over web based social networks and applications.

The central to this trust ontology is the notion of trust formalized as an association between any two agents. The 'trusted agent' represents the binary relation between the notions of topical trust and a person. The notion of 'topical trust' in this ontology formalizes the trust that is about a specific topic under focus. In other words, this topicality provides a type of context for the trust between two agents. The constraint is imposed on the notion of 'topical trust' according to which minimum one 'trust subject' and one 'trusted agent' must be available for a meaningful topical trust to take place. The notion of 'trust subject' is defined as a more specific type of trust topic. Moreover, the binary relation called 'trust regarding' is defined in order to associate an agent to a topical trust. The notion of 'trusted agent' is defined an agent which is associated with topical trust,

[1]Available on the URL: http://www. schemaweb.info/schema/SchemaInfo.aspx?id=171

and the notion of 'trusted person' is a person which is associated with topical trust. This trust ontology also contains the class called 'trust level' that provides an extent to which the level of trust can be represented between any two agents involved in a trust transaction. The trust level in this ontology is hard coded on the scale of 0 to 10, where trust level 0 represents low level of trust between two agents and the trust level 10 represents high level of trust.

As it can be seen from this trust ontology all the definitions in this ontology are shallow in nature which failed to provide a fine grained details associated with the key concepts in this ontology. For instance, this ontology provides the notion of trust level but it fails to provide a qualitative and quantitative distinction between trust level. Similarly no further distinction is provided while formalizing the notion of agent into sentient and cognitive agent in order to specify whether an agent can perceive a plan in order to achieve certain goal. Moreover, this ontology fails to consider any object properties that would provide an indication about how the notion of context is associated with a certain situation. This ontology failed to distinguish between different types of trust transaction that may take place between any two agents, e.g. direct trust transaction, indirect trust transaction, transitive trust transaction, etc. In our viewpoint it is important to provide such a type of distinction because it enables users to represent the trust transactions cleanly without having to blurred the distinction that exist among them. As described in the previous section, this ontology also fails to consider the distinction between explicit context and implicit context that is crucial to represent the trust transactions between agents which are not only human-agents but also between a human-agent and a system-agent. No indication is provides about whether this trust model takes into account the notion of access which is an outcome of a high confidence trust level between two agents.

2.7 Trust ontologies for web services

This work proposes a formalism for the trust requirements modeling framework in the service-oriented environments. The main aim of this framework is to provide a way to study trust worthiness. Because this model subscribes to the social trust view, here, the notion of trust is realized in terms of the interaction between software agents that typically represents the social nature along with cognitive attributes and human beliefs. In this model the trust concepts are divided into belief and trust reasoning.

In this ontology, the class agent represents an active entity and it plays the role of trustor and trustee. The class trustor is responsible for initiating the trust relationship while the class trustee is an agent which receives the trust estimation. The class goal in this ontology represents a condition or state of affairs of the world. Because this trust ontology is specifically developed for the web service application domain, the class service is a set functions. The class action represents an action behavior of an agent towards composing of a service. The consequence of any action is formalized by defining the class consequence, which has non-zero utility with a specific action effects. The class resource represents a physical and informational object. This trust ontology also provides associations between concepts with a specific source and target concept. At a certain level of abstraction as a part of trust analysis, the notion of belief is classified into core belief and confidence belief based on the roles they play in turst. To reason about the relevant beliefs of agents the operational part of the ontology provides a set of rules which take into account six different beliefs, e.g. rules about competence belief, intention belief, persistence belief, reputation belief, etc. It is crucial to realize the notion of reputation belief acquires agent's belief which is used in formulating a vector.

This trust ontology provides a clean distinction between the roles that agent such as trustor and trustee, but it fails to provide abstract generalization of the class agent into sentient and cognitive agents. In our point of view it is important to provide such a type of distinction mainly because it provides a clean indication about whether an agent perceives a goal. And if an agent, e.g. sentient agent, can not perceive a goal then it makes it difficult for such an agent to make trust related statements about another agent. More importantly this trust ontology fails to provide any indication about the notion of a situation that determines in which context the trust relation between trustor and trustee takes place. Moreover, this trust ontology takes into account the notion of a credit value to determine whether an agent trusts or distrusts another agent, but no ontological support is provided in this framework to specify the quantitative and qualitative credit value between any two agents. Finally, this trust model does not exploit different types of trust that may be essential to identify trust relations between trustor and trustee. For instance, in one type of trust a human-agent may trust another human-agent, or in other type of trust a human-agent trust an artificial agent, or in some other type of trust an artificial agent trust another artificial agent.

2.8 Trust ontologies in E-Service environment

This framework [CDH07] proposes a trust ontology which is suitable for different types of agents that exist in the service-oriented environment. In this ontology the notion of trust is defined as a belief or faith that a peer has in another peer in a specific context, such that source peer gets a confidence to carry out a transaction with another peer. The generic trust ontology is divided into following three categories – 1) agent trust ontology, 2) service trust ontology, and 3) product trust ontology. In generic trust ontology, trust is formalized by the concept, trust relationship. Here, each trust relationship has a trust value associated with it from trusting agent to trusted agent. The trusting agent in this ontology must be an intelligent agent while the trusted agent can be product or service which according to this framework does not contain intelligence. The trust relations also composed of the notion of context which provides a scope within which a trust relationship between trusting agent and trusted agent takes place.

The notion of context is further decomposed into the quality aspects of agent, product and service. These quality aspects associated with agent, product, and service provides an assessment criteria to measure the quality aspects. While the agent trust ontology inherits the generic trust ontology, it specializes the notion of agent trust which is always considered to be a direct trust interaction with another agent. In agent trust ontology, the notion of trust value is specialized as a trustworthiness value to estimate the trust. For instance, the notion of trusting agent is specialized as a service requester while the trusted agent into service provider. The service trust ontology focuses on a provider and not on a service to measure trust. As a result, the notion of a trust value is realized in terms of trustworthiness value to assess the quality of a service. Here the service itself represent the notion of a context within which a service transaction between service requester and service provider takes place. Finally the product trust ontology is realized as a specialization of the generic trust ontology in service-oriented network to characterize the trustworthiness a customer (trusting agent) in certain product (trusted agent). The other notions of context, assessment criteria quality aspect and time slot are realized similar to their ontological specification from generic trust ontology but in the service-oriented product based situation.

This trust ontology provides a systematic categorization of the top level trust ontology in different set ups like agent trust, service trust and product trust. In line with earlier proposals described in the previous sections this trust ontology also provides a consistent vocabulary to formalize the notion of a trust. However there are few areas associated with this trust ontology which in our perspective require further analysis. The key limitation of this trust ontology is that it only takes into account source of the trust relation as an intelligence agent, and a product or a service as an agent cannot represent its trust in other product or in service. In our point of view this limits the applicability of this trust ontology to tackle the problem areas in which, for instance, one ontology which is part of an intelligence software trusts either a module of another ontology or a new ontology. The ontology here as trusting agent can make such a type of trust related decisions because the set up in which this ontology is operating, i.e. a software has sufficient intelligence enable an ontology to make such decisions.

Moreover, this ontology fails to provide a fine-grained specialization of key concepts associated with the trust ontology, e.g. agent, context. Because the notion of agent is not further clarified, it cannot handle a situation in which a cognitive agent that can perceives a plan and takes a responsibility for its effects is making trust assertions on behalf of another agent that may perceive plan but may not be held responsible for its effects. Also, it is not clear how the notion of context makes use of different contextual features that may be associated with a situation involving a trusting and trusted agents. The notion of an agent trust in this ontology is always assumed to be a direct trust between trusting and trusted agents, but there is no provision to represent indirect trust that may have to be extended from trusting agent towards some third-party entity to which trusting agent is indirectly related in a business transaction (e.g., a product authored by a trusted agent).

2.9 Implications of the existing work on the NeOn

In this chapter we reviewed several approaches to modelling trust in different circumstances. To summarize the key features we want to focus on in our work, we propose the following table with the main functionalities being associated with individual approaches.

Table 2.1: A summary of key features of the reviewed approaches to modelling trust.

Functionality	**Approach (discussed in section)**							
	2.1	**2.2**	**2.3**	**2.4**	**2.5**	**2.6**	**2.7**	**2.8**
Trust is personal	1	0	-1	-1	1	1	1	0
Trust is situational	0	0	-1	0	-1	0	1	0
Trust applies to diverse entities	0	-1	1	1	0	-1	-1	1
Trust may have a value	MV	HW	HW	OP	MV	MV	HW	OP
Trust context is explicit	-1	-1	-1	0	0	0	0	1
Support for different perspectives	-1	-1	-1	1	-1	-1	-1	-1

As we see in table 2.1 there are several overlaps in the way trust is treated in different tools and frameworks. For clarification, 'MV' stands for multi-value support for expressing the level of trust, 'HW' stands for hard-wired, and 'OP' stands for open, i.e., user-pluggable value system. The presence of number '1' above means a presence of a particular feature as documented in the literature, on the contrary '-1' means the given functionality was not observed, and '0' means limited, partial, restricted support.

In the order of preference, as will also be documented in the practical requirements in the next chapters, we should highlight the prevalent presence of '-1' in the capability of modelling several different perspectives on trust and allowing the user to plug in his or her interpretation, e.g., in the form of explicit context. Another feature that is not well supported concerns the situational nature of trust – we argue that trust is not an objective characteristic but always applies to a particular situation, which in turn involves an agent, an entity, the level of trust, etc.

In the following chapters we will provide further particulars for justifying our perspective(s) on modelling trust to address the gaps highlighted in this part of the report.

Chapter 3

Needs and Motivations for Modelling Trust

After mentioning a few abstract issues with capturing trust as a relationship, we look at a sample of motivations and situations where access control play some role and thus determines some form of trust. The purpose of this short section is neither to define, nor to restrict the scope of trust to the access control. On the contrary, our objective in formulating these short scenarios is to show that there is more in access and authority control than what most readers are familiar with from their desktops and corporate Intranets.

3.1 Motivation: Cautious knowledge sharing

This scenario contains a necessary element of content sharing and protection on one hand side, as well as a necessary element of collaboration on the other side. Consider a hypothetical manufacturer of engines – for sake of generality and anonymity, let us assume this manufacturer is active in the car industry. The core activities of the manufacturer can be classified in several categories, among others: design of engine components, design of new engines, testing of produced engines, maintenance of engines sold to customers, servicing support for the customers, etc.

Being in a competitive market, the hypothetical manufacturer aims to attract as large a share of the market as possible by concentrating on the servicing aspect of the business. In other words, they may be willing to discount their engines for an initial purchase, subject to ongoing maintenance and servicing contracts. Obviously, focus on maintenance is economic if the engine is highly reliable and the manufacturer can obtain a detailed data feed from its customers about the engine performance. There is also a strong motivation for creating an annotated repository of known failures and servicing practices, so that an issue can be addressed rapidly. And here comes an issue: one needs to disclose some information about engine components so that data can be obtained and some customers (e.g., garage franchises) can solve some minor problems, but at the same time, one wants to keep the bigger picture and the failures fairly private, so that the competition cannot copy the core details.

Translated into the language of networked ontologies, we can describe the above situation as comprising several ontologies. Let us denote $O_{abstract}$ an ontology defining generic upper-level terminology, such as causality, temporality, agency, etc. Next, let us assume there is a generic ontology, a catalogue of engine components, their names, types, etc. as agreed and shared between car manufacturers and suppliers, and denote it as O_{shared}. Based on this ontology, there will be a more concrete ontology comprising engines, engine parts, and their configurations actually used by our manufacturer, O_{design}. Next, our manufacturer maintains an ontology of servicing, testing and maintenance procedures and best practices, $O_{practices}$. Finally, there is a richly instantiated and populated ontology containing concrete data gathered and inferred from the customers, O_{data}.

Since our engine manufacturer heavily relies on collaboration with both the franchise garages and the customers, it cannot adopt a too restrictive strategy to ontology (and in particular, ontology instance) sharing. For example, while $O_{abstract}$ and O_{shared} content can be freely used to search their repositories by anybody, they want to be more cautious with other ontologies: O_{design} may be exposed to both garages and customers, so that they know what they are buying/servicing and what do their data relate to; $O_{practices}$ shall be hidden from the customer and non-franchise garages, as it embodies the core know-how of these partners. Finally, O_{data} is likely to be hidden from all but a few departments and teams internal to the engine manufacturing company, as it contains implications for further development of new engine components, recommendations, decisions based on data gathered from the clients, partners, etc.

In this scenario of *cautious knowledge sharing* a similar situation as described earlier in Motivation 2 arises. A knowledge repository maintained by our manufacturer is ontology driven to enable semantic access to its content. This ontology, let us call it $O_{repository}$ has (and must have) a totally different conceptual meaning for different stakeholders, for example, as follows:

Joe Bloggs , a member of public, sees $O_{repository} \equiv O_{abstract} \cup O_{shared}$

Jane Smith , a fleet manager, sees $O_{repository} \equiv O_{abstract} \cup O_{shared} \cup O_{design}$

Bob Plier , a technician in a garage franchise, uses $O_{repository} \equiv O_{abstract} \cup O_{shared} \cup O_{design} \cup O_{procedures}$

Alice Best , an internal planner and analyst, uses $O_{repository} \equiv O_{abstract} \cup O_{shared} \cup O_{practices}$

etc.

In addition to using different ontologies, different stakeholders may be entitled to different interaction modes. For example:

Joe Bloggs may only read individual ontologies $O_{abstract}$ and O_{shared}

Jane Smith may only *read individual* as well as *networked* ontologies $O_{abstract}$ and O_{shared}

Bob Plier may *read networked* ontologies $O_{abstract}$, O_{shared}, O_{design} and $O_{procedures}$

Alice Best would be allowed to read the same as Bob, but also *update* $O_{procedures}$

Nigel Valve , an internal engine designer, would be allowed to read all ontologies, similar to Alice, but in addition he can *extend* O_{design}

etc.

3.2 Presence of trust in NeOn research

In NeOn we have several use cases where trust issues come into play. For example, the Open Rating System [SAd^{+}07] allows for ratings of ontologies and ontology modules by users of these ontologies. It can be used in a similar way to rating systems on e-commerce sites, i.e., the user can rate resources with 1 to 5 ‘stars’. For a more in-depth discussion of the Open Rating System we refer the reader to [SAd^{+}07, LSNM06].

Other use cases for trust arise in the context of distributed collaborative ontology editing and revision. As this use case is more complex and poses interesting research questions with respect to trust modeling, we focus on the ontology revision use case here. The proposed modeling is very flexible, such that it allows for the modeling of both use cases. Consider first the detailed walkthrough using AGROVOC maintenance, a case study related to work package 7.

2006–2008 © Copyright lies with the respective authors and their institutions.

3.3 Use case: Ontology Editing in AGROVOC

AGROVOC is a thesaurus of terms about agricultural domain. Lately, this thesaurus is undergoing a transformation into ontological format adopting OWL-DL language. The resulting ontology, called AGROVOC Concept Server, is the conceptualization of knowledge about lexical domain; typically the ontology will define relationships among the generic terms in a thesaurus, in a way that is more detailed than what is actually achievable (BT, NT, RT used, usedFor). A generic term will be connected to its lexical variants by a number of relationships; the model will then be instantiated in several other knowledge models focused on particular scope of agricultural domain (e.g., plant, rice, pests, etc).

Another objective related to AGROVOC ConceptServer is to be able to maintain the lexical information in the generated models, by means of a centralized web application (called AGROVOC Workbench) where all maintainers can connect and propose modifications to the lexical variations. AGROVOC maintainers will be professionals with heterogeneous agricultural domain expertise, will have different information management skills, and can eventually be simple project supporter with end-user capabilities.

3.3.1 Overview of the AGROVOC scenario

The scenario of lexical knowledge maintenance in AGROVOC modules, is very common to any editorial work flow (see e.g., D1.3.1 [PHWd08]) of a resource that is edited by knowledge workers assigned with specific editorial role. As a guiding idea we consider that the operations possible trough the Workbench affect primarily the lexical information (e.g., add/modify/delete term translation, synonym, acronym, etc). Moreover the users can contribute to local re-organization of the structure of the existing agriculture domain models (e.g., change class structure, add/modify/delete instances and other common ontology design operations). In both cases the actions follow the same workflow and all commands are limited by the grants assigned to the users when creating their account profile. For some of them the modifications will be assumed as suggestions, and verified by more expert users who can commit or reject the changes.

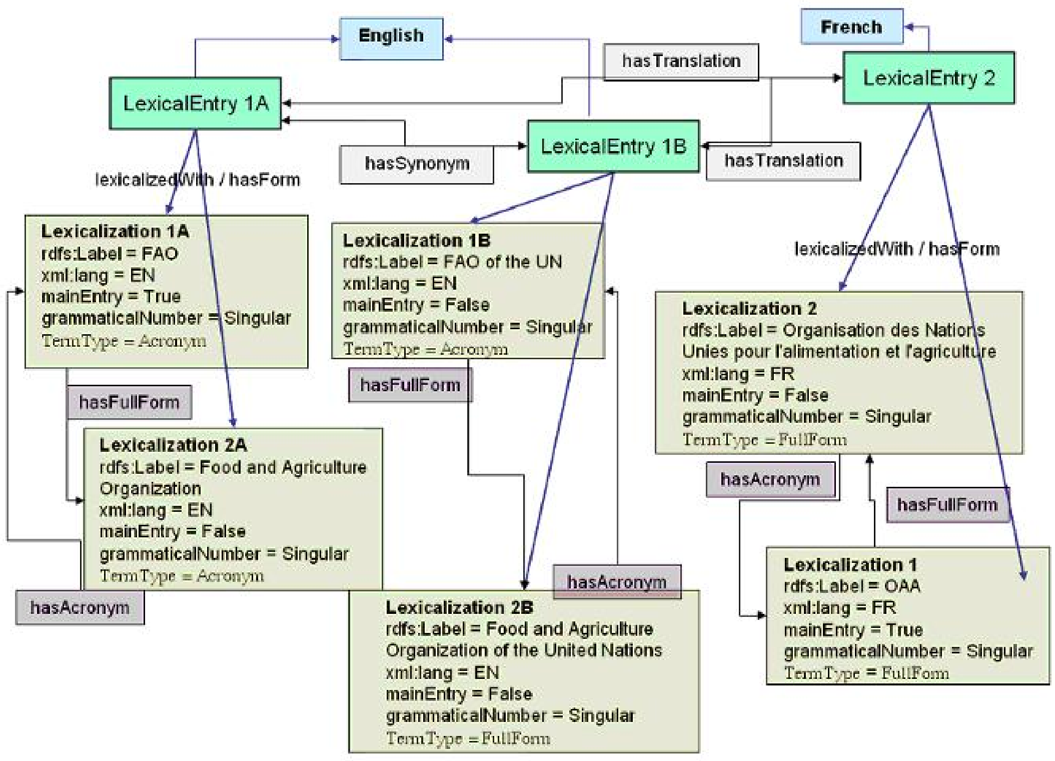

Figure 3.1: Agrovoc Schema Instantiation

An example of how to instantiate the knowledge in the ConceptServer is shown in Figure 3.1. All the entries in the picture can be modified at time by the users of the Workbench application. The project distinguishes among different categories of users:

Not-logged-in users: they consume all end user services of the Workbench. They can submit suggestions about term lexicalization in any language, they can also submit suggestions about local re-design of the specific domain models. All suggestions made by these users are associated with their IP address and identified as “guest-proposed”.

Term editors: their maintenance activity will have effect only in a given lexical scope, meaning that they have the right of add/edit/delete term lexicalization restricted to all and only the languages they have been assigned when creating their account profile. All other modifications performed at the level of mode design, will be passed over as suggestions, and processed by more experienced users.

Ontology editors: they can perform the same activities as the term editors, plus have right to submit modifications to the data model design.

Validators: experts that can verify and validate the work done by the editors (terms and ontology editors) restricted to all and only the languages they have been assigned when creating their account profile.

Publishers: validate editors and confirm validators" work for final publication outside FAO systems; they are also restricted to all and only the languages they have been assigned when creating their account profile.

3.3.2 The role of trust in the AGROVOC Workbench

Given that all knowledge workers contributing to AGROVOC maintenance may be seen as external information sources, and that the validation job is basically the assessment of the level of goodness and soundness of this external information sources, we can think to characterize each knowledge worker’s profile with *a level of trust*. This level could be initially set be the validator (this also means that the same editor can be evaluated differently on the trust perspective by different validator), and can evolve as a function of the rejection or acceptance that a single validator performs on one editor.

We can think of a use case where trust can be useful in ranking the objects in the change queue for a validator who prefers to first assess the changes from editors s/he trusts most, or the reverse s/he may want to spend time first on editors’ contribution that requires more attention to validate.

Another use case can be to quickly validate a job done by a highly trusted source of modification by using a “validate all” button that performs a global commit for a subset of editors enjoying a certain specific level of trust. Yet another use case can be to give trust values to entire organizations whose people participate in AGROVOC maintenance – i.e., to move from a direct authorship to a more general provenance. From the perspective of the end-users this means to quickly filter those AGROVOC elements edited, maintained, or ranked by organization they believe to be more valuable in a specific agricultural sub-domain.

When changes to the conceptual model result in contradictory information, for each of the possibilities a trust value can be computed referring e.g., to the aforementioned provenance or other types of context. The validator/publisher/administrator is shown all possible truth values for contradictory axioms, together with their respective trust levels. S/he can resolve the inconsistency taking into account the appropriate context and using it to validate the trust assertions (i.e., to decide which statement s/he believes in). In the future extensions of the use case, it will be possible to maintain contradictory information, when an agreement on a single version can not be reached – e.g., by displaying multiple thruth valuations to the users. Inferred trust levels allow to track the provenance of such information, i.e., to determine, which (groups of) users supports which view of the world. This information can then be used to asses, which version a user wants to trust.

3.3.3 Requirements arising from practice

From the use cases briefly sketched in the previous section we can derive the following requirements:

1. Trust must be expressible on various levels of granularity, e.g., Ontologies, Modules, Axioms, but also Agents or Organizations.
2. Trust must be expressible by and for both the information resources, e.g. Ontologies, but also for agents, e.g. a particular user.
3. It must be possible to use multiple measures – "trust orders" – to express trust values. While the open rating siystem has a rather simple model of trust, in the AGROVOC use case, every user might have a different trust order, different valuation technique, etc. Further, these orders need not be linear.
4. Trust model has to be context-dependent. In AGROVOC we may highly trust modfications from a user done in one part of an ontology, but distrust his modifications in a different part.
5. Trust is personal. Trust preferences between editors can vary from those of validators.

Chapter 4

Evolution of the Trust Metamodel for Networked Ontologies

As we mentioned in the introduction to this report, in the current networked environment of the Web, but also in any database-driven information management systems, one has to deal with a large number of resources mutually related not only in a syntactic way but increasingly also semantically. The Semantic Web is seen as an information space where not only human users but also various artificial agents would be able to share, access, update and extend semantically meaningful information and knowledge. However, in order to realize this sharing in a way that would reflect the needs of various stakeholders, as described in our introductory section 1, e.g., in the three motivational situations, some support is needed for so-called *cautious sharing*.

In addition to the requirements raised in the previous chapter, we should note the following. Trust between agents is often not transitive [AG07]:

Trust is not necessarily symmetric. If "*Alice trusts Bob*", it does not follow that "*Bob also trusts Alice*".

Trust is not always distributive. If "*Alice trusts (Bob and Carol)*", it does not follow that "*(Alice trusts Bob)*" and "*(Alice trusts Carol)*".

Trust is not inherently transitive. If "*Alice trusts Bob*" and *Bob trusts Carol*", it does not automatically follow that *Alice also trusts Carol*.

All these additional issues may only be used in certain situations, which we want to express in terms of allowing contextualization of trust assertions.

4.1 Initial models of trust in NeOn

We start by modeling trust in general, using a very high-level notion of trust and trust degrees. In the following section, we will explain, how complex trust degrees are modeled. In order to design the trust metamodel, we have here reused design methods from NeOn, and specially the ontology design pattern repository [ODP].

Figure 4.1 shows the concepts in our trust ontology. Figure 4.2 shows the modeling with properties. Later in this deliverable we will explain how the trust model integrates with OMV. Hence, we directly use UML in fig. 4.2. The ontology is accessible online at http://neon-project.org/ontologies/2008/10/trust-metamodel.

Logically speaking, trust is a rather complex, potentially polymorphic relation: new arguments can be added based e.g., on the number of roles: trustor, trustee, target, degrees, trustor ontology, trustee ontology, place, time, and other contextual roles, which cannot be predicted in advance. It is therefore straightforward to use an ontology design pattern [PGD$^+$08] that is able to turn a polymorphic relation into something expressible in OWL, with a general, reusable vocabulary. Such pattern is called `situation` [SIT], and can represent any relation ρ^n as a subclass S of the `owl:Class:Situation`, and any binary projection ρ^2 of ρ^n

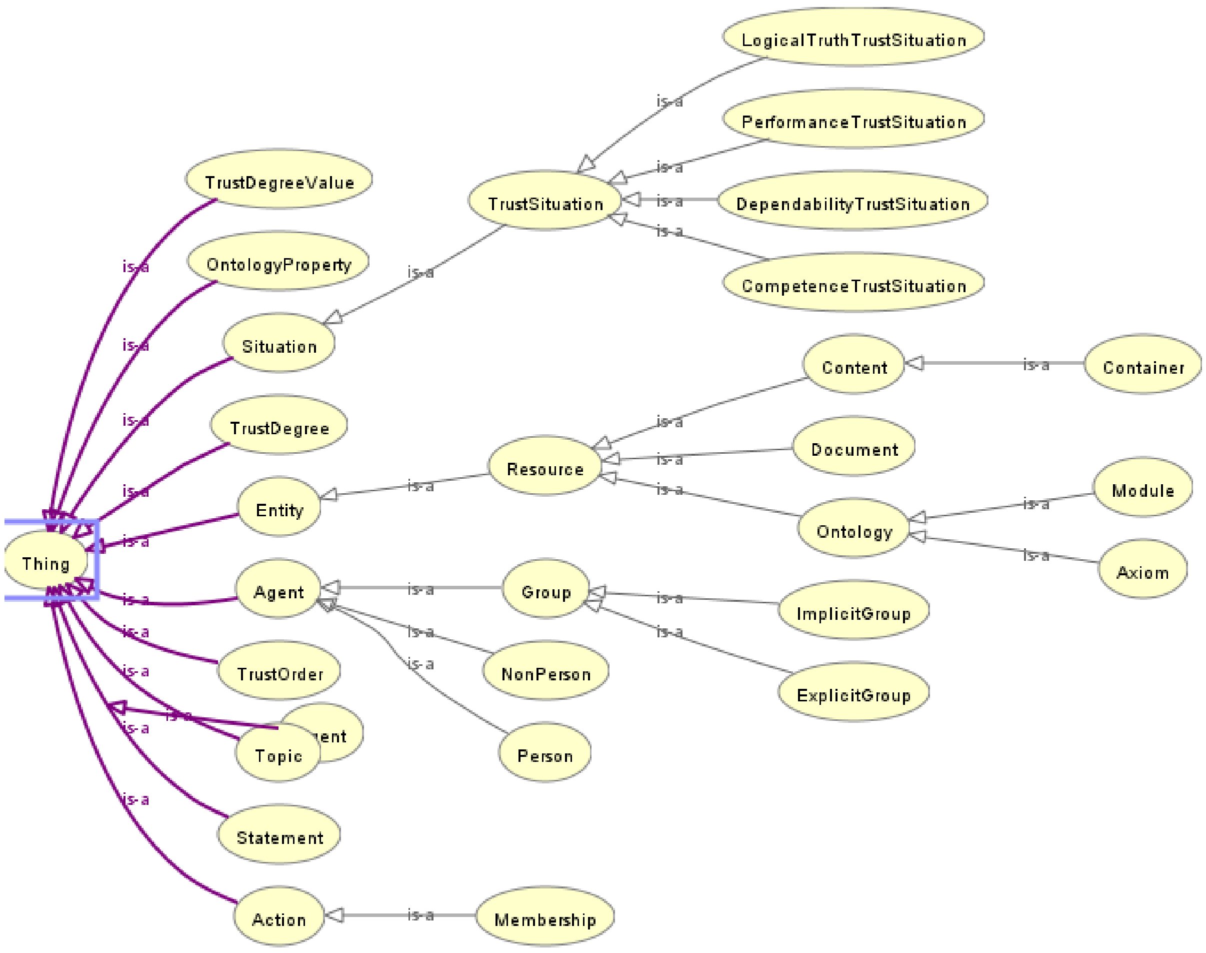

Figure 4.1: Classes in the initial trust ontology

as a subproperty P of the `owl:ObjectProperty:isSettingFor` that has S as its `rdfs:domain`, and the expected range of ρ^2 as its `rdfs:range`. Theoretical details on a social ontology including the `situation` pattern are presented in [Gan08].[1]

Accordingly, we formally represent trust as a subclass of `Situation`. Complying to this view, a *TrustSituation* represents the assignment of a trust value to an *Entity* by an *Agent*, optionally in a certain context. An *Agent* can be a legal or natural person, a group, an autonomous software agent, etc. Details of the concept of *Agent* are not specified here, but in a separate ontology module. For the NeOn use cases, we will have natural persons (e.g. validators) and legal entities (e.g. FAO) as agents. An *Entity* is anything, which can be trusted, i.e. an *Agent* or a *Resource*. The relevant resources in our use case are *Ontologies*, *OntologyModules* and *Axioms*. In this part of the ontology, the link to OMV can be found. *TrustSituations* can further be differentiated into *LogicalTrust*, i.e. trust in the truth value of Ontologies or Axioms, and *SocialTrust*, i.e. trust in the Competence, Performance and Dependability of an Agent.

CompetenceTrust means the trusted agent is competent to help in a situation. PerformanceTrust means, he is able to do this in a performant way. Dependability is the degree, to which an Agent trusts, he can make himself dependent on the trusted agent. These distinctions catch the cognitive trust distinctions presented e.g., in [FC04, HF06]. The rationale is here to take seriously the maximal trust relation, and to admit any relevant projection, be it trust between agents, agents and resources, between resources, etc.

Similarly with the `situation` pattern, a `region` pattern [REG] is used to associate the arguments that can hold for a trust order relation. As there exist various approaches to trust orders, we abstract the concept

[1]This implies that the `situation` pattern provides a basic vocabulary to N-ary relations.

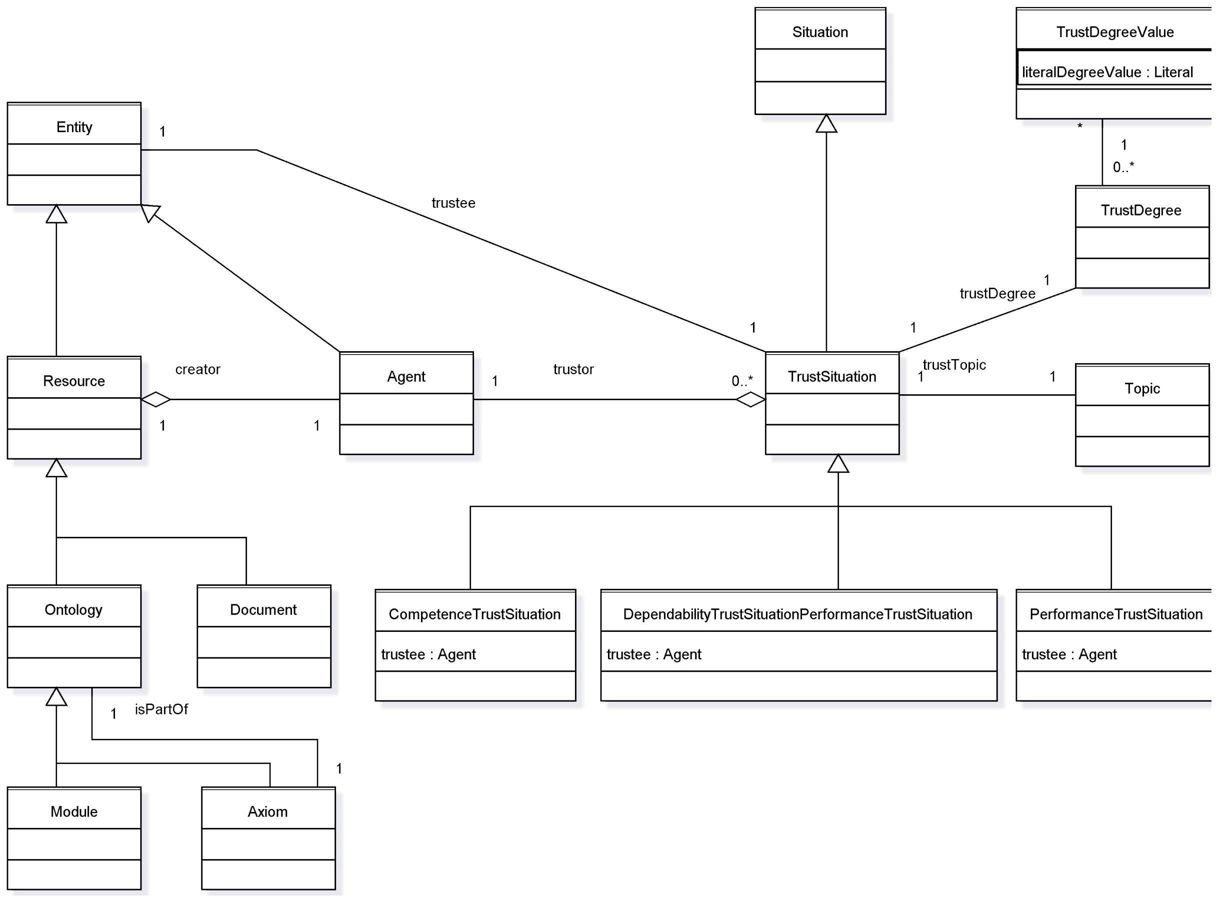

Figure 4.2: UML Diagram of the initial Trust Ontology

of a *TrustDegree* as a specialization of the `region` pattern. A TrustDegree can either have a numerical value (literalDegree), or an abstract degree for modeling partial trust orders. Partial orders will be discussed in the next section. The reification of trust values also allows to extend TrustDegree with meta information such as uncertainty degrees in the future.

As explained above, trust in general is not transitive. In certain cases, however, transitivity can safely be assumed. Some examples of transitive trust associations gleaned from our interaction with NeOn case studies include the following[2]:

Transitivity via partOf If an Ontology is trusted, also the Axioms in the Ontology are trusted. In general, this holds for partOf relations. This rule can be modelled as follows:
$\texttt{trustedEntity} \circ \texttt{hasPart} \sqsubseteq \texttt{trustedEntity}$

Transitivity via creator If the creator of a Resource is trusted, so is the Resource. This can be modelled as: $\texttt{trustedEntity} \circ \texttt{creator} \sqsubseteq \texttt{trustedEntity}$

4.2 Modeling trust orders

As we have seen in the use case, trust may not always been expressed in a linear order like the unit interval. Hence, a modeling of partial orders is neccessary. In particular, a trust order is a partial order with a supremum. The supremum is the information (information source, author) who is trusted most. The supremum usually corresponds to the agent creating the trust order, but this is not neccessary. As we have shown in

[2]Note that these are merely examples for illustration, formal and systemic analysis of when and how transitivity can be applied will be part of our operational implementation and realization of the trust meta-model in the coming months.

[Sch08], we can derive a lattice of truth values for paraconsistent reasoning with trust levels from every partial order of trust levels, which has a maximum.

Partial orders can not be specified based on XML schema datatypes, because these are based on strict orders. Hence, user defined datatypes in OWL are not very helpful here. Instead, we provide an OWL vocabulary to model arbitrary orders. It is based on [DRS$^+$06], which specifies a vocubulary for lists in OWL. Please note that the RDF vocabularies for collections are not available in OWL and have very weak semantics. We generalize [DRS$^+$06] to arbitrary orders. Further, we make use of OWL2 [GM08] features, in particular role chains, to model certain properties of orders, which are not expressible in [DRS$^+$06], which is based on OWL1 [S. 04]. We show a part of the order ontology in figure 4.3. In the following we explain the modeling and some interesting features.

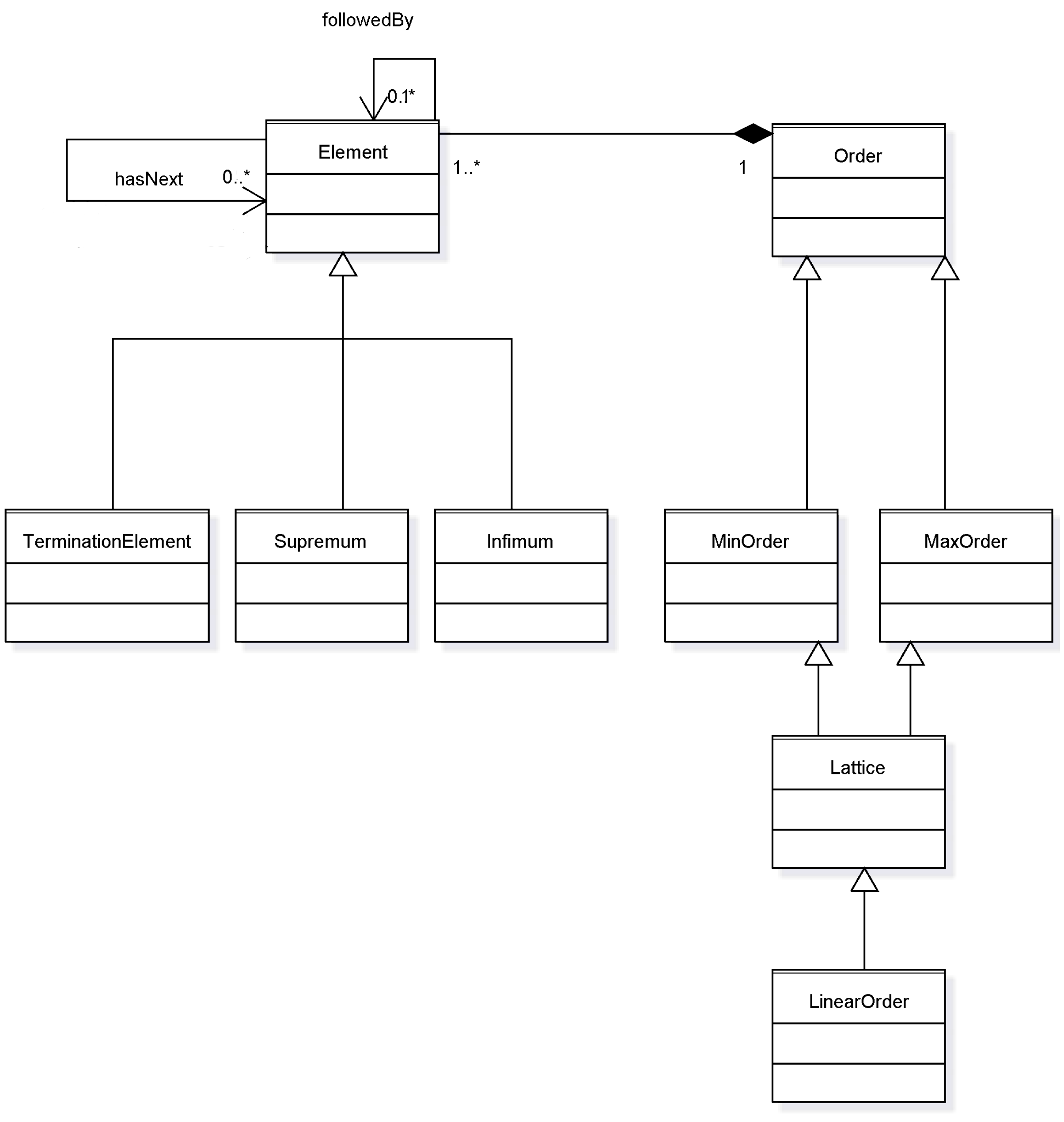

Figure 4.3: The Order Ontologies Class Hierarchy

An *Order* consists of a number of *Elements*, which are connected via *hasNext* relations. There is a special element, the *TerminationElement*, in each order, which is used to terminate the *hasNext* chains. It corresponds to the *nil* or *emptyLit* element usually used in linked lists. *isFollowedBy* is the transitive closure of the *hasNext* relation. It is modeled using role chains to avoid making *hasNext* transitive[3]. The *Supremum* (*Infimum*) of an order is defined as usual as an element which have no successors (predecessors) via *hasNext* (*isNextOf*). *Lattices* are orders, which have exactly one minimal and maximal element. LinearOrders are a special case of *Lattices*, wher every element has at most one successor.

As shown in [DRS$^+$06], this modeling allows for pattern matching within orders using DL reasoners. The expressivity is comparable to Regular Expressions. The authors also show, that performance allows for real-world applications. In our extension, we can additionally search for example for supremum and infimum of two elements using conjunctive queries.

4.3 Accounting for additional features of trust

As we mentioned in chapter 1 and earlier in this chapter, there are many relationships arising from the practice and the real world that carry an explicit or implicit, direct or indirect factor of trust. For the purposes of this deliverable, trust is seen as an act or an outcome of an act, in which one party can be said to put some reliance in other party or in other party’s acts. In principle, trust can be an emotional or a logical/rational act arising from one agent (so called, emitor) in the direction of another agent (target) with no assumption of a full knowledge of the target’s intentions, qualities, etc. Typically, the act of trusting is closely bound and circumscribed by/in a specific situation. However, one additional aspect that has not been extensively addressed by past approaches and by our initial ontology, is the need to increase the flexibility and the cleanliness.

In other words, we would like to allow in our trust framework to express also referential trust assertions, whereby a trustor can be an ontology – or better, trust can be emitted from an Ontology, OntologyModule, etc. In a similar tune, we would like to present a strictly divided modelling of trust as a relational view on a situation that is complemented (justified, grounded, etc.) by a trust context.

Thus, we see trust as an abstract belief that can be captured in a form a trust assertion. In fact, it would be more appropriate to talk about a partial assertion about the relationship in a situation involving trust, but this is a bit a mouthful; hence the abbreviated use of “trust assertion”. Nevertheless, the key aspect to take away from our definition is the fact that trust does not exist objectively – it only makes sense to talk about trust as something being *situational*, bound to a concrete (explicit or implicit) situation. In this work, a situation is seen as the relative position or a combination of circumstances at a given point in time enabling one to relate things, agents and various entities to each other. Our notion of “assertion” serves two important roles in our model:

1. Assertion is *an expression* about a trust being observed to partially make sense or interpret some situation; i.e., it describes some aspect of a situation in which a particular relationship holds.

2. Assertion is *always attributable* to an agent with a system of beliefs (i.e., a cognitive agent). Literally, an assertion is a declaration made by a cognitive agent with/within a certain belief comprising an expression that can be, in principle, evaluated as true or false. Generally *assertions are made as ‘true’ with no objective supporting evidence requried*.

In understanding and conceptualizing the notion of situational trust, we apply two mutually complementary perspectives on how a situation may be modelled. In order to capture the core relationship between the source and target, we opt for so-called *transactional perspective* – the perspective that perceives a situation as a transaction. This view is enabling an agent to recognize two or more parties involved in some form of relationship (e.g., business, communication or negotiation). This perspective allows one to understand relational and temporal characteristics of a situation.

[3]An alternative modeling, which would introduce this problem, would be to make isFollowedBy a super property of *hasNext*

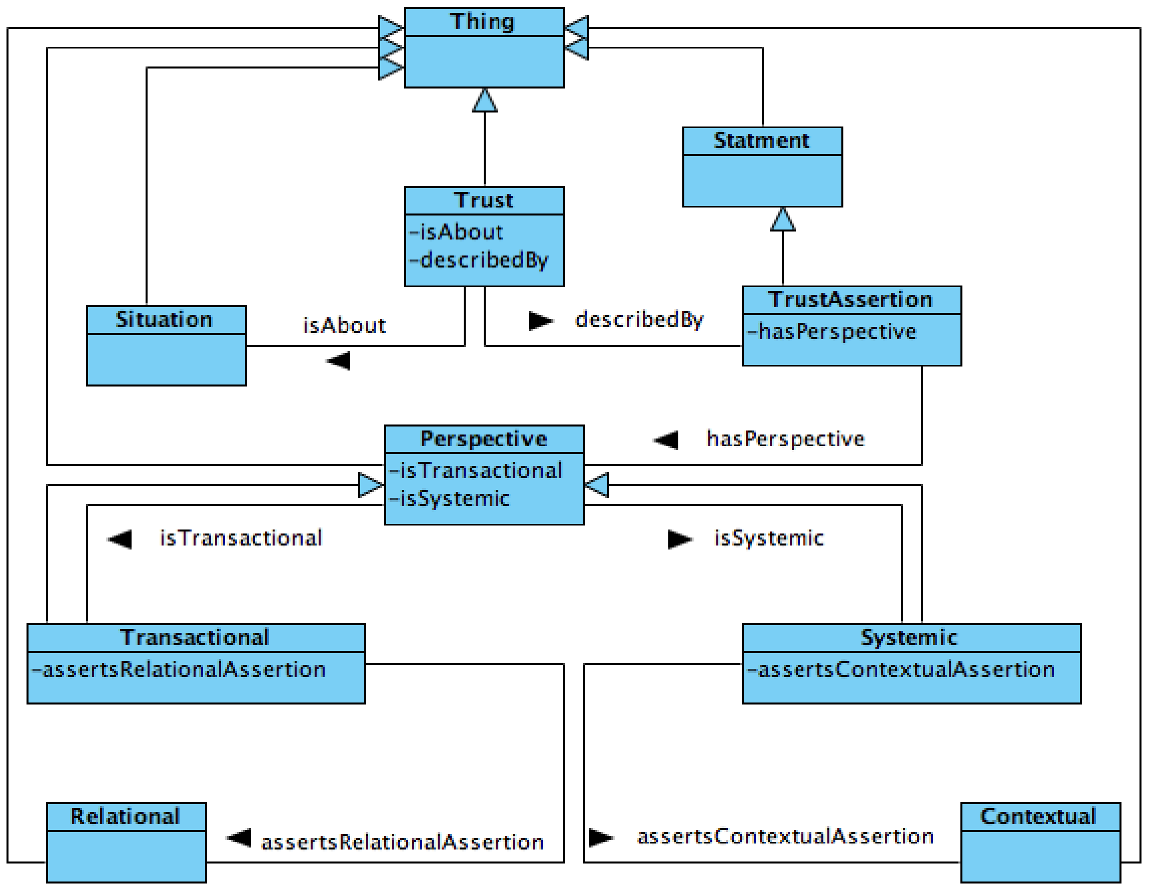

Figure 4.4: Basic approach to modelling trust as a situational relationship comprising a transactional and systemic (contextual) aspects and embodied in a trust assertion.

To complement this viewpoint we proposed applying a *systemic perspective*. This is a perspective that perceives a situation as a system, i.e., it is enabling an agent to recognize the individual functional parts and the interactions among them. Usually, a systemic perspective leads to a formulation or description of a system or of a (partial) model of the situation.

To reflect the natural language statements from the earlier chapters, the two perspectives can be roughly aligned to two clauses forming a usual trust assertion – as schematically shown in Figure 4.4 and illustrated in Table 4.1. The three statements illustrated and broken down in Table 4.1 correspond to the following three types of situations:

1. Alice asserts that she trusts ontology O, because it was authored by Jim.

2. Jane asserts that she believes Bob must be trust John to let him drive a car.

3. Watson (search engine) interprets that ontology O must be trusting module M, based on the semantics of owl:imports property.

We start developing our modelling framework in steps, addressing different issues that were alluded to in the introduction. We start by briefly describing the structure of the proposed model and continue by discussing and distinguishing between two faces of trust – relational and contextual.

Table 4.1: An association of informal clauses to two perspectives on a situation.

	Perspectives on the situation				
Believed by	**Transactional clause**			**Systemic clause**	
Author:	**Emitor**	$\ggg$	**Target**	**|**	**Context(s)**
Alice:	*Alice*	trusts	*Ontology O*	because	*authoredBy (O, Jim)*
Jane:	*Bob*	trusts	*John*	in the scope of	$\exists$ *c* $\in$ *Car: drivenBy (c, John)*
Watson:	*Ontology O*	trusts	*Module M*	by virtue of	'O <owl:imports> P'

4.4 Preliminaries: Structure of the trust metamodel

In order to reflect the above-mentioned perspectives on a situation, we decided to formally structure and modularize our metamodel alongside the same criteria. Thus, our metamodel for capturing trust comprises (and formally draws upon) two sub-modules:

- Module for capturing the transactional aspects of trusting (i.e., allowing a cognitive agent to assert the parties and direction of trust), and
- Module for capturing the systemic or contextual aspects of trusting (i.e., allowing a cognitive agent to enrich and ground the transactional statement in a specific context, scope, time interval, and similarly.

To allow for our definitions being more formally grounded in existing upper-level ontologies and to enable the application of the trust metamodel in different application domains, we formulated the key abstract concepts (such as *Situation*, *Agent* or *Expression* in a small upper-level ontology that can be, in principle, aligned or replaced with other upper-level ontologies (such as SUMO or DOLCE). This abstract module is imported by both transactional and systemic modules, which are described afterwards.

4.5 Modelling trust as transactional statements

The core idea behind this module is based on the transactional perspective, view upon a situation – i.e., a perspective that perceives a situation as *a transaction*. This view enables an agent to recognize *two or more parties involved* in some form of relationship (e.g. business, communication or negotiation). This perspective allows one to understand *relational and temporal characteristics* of a situation.

The notion of trusting is seen here as a form of transaction occurring between two agents. It allows for grounding trust as an act of realizing any relationships (e.g., carrying out business, negotiations, communication, etc.) and typically involving two or more parties (e.g., a buyer vs. a seller). Transaction being an abstract event, we formally introduce the notion of a statement about a transaction as an explicit message representing the fact that a transaction occurred.

In particular, we formalize transactional statements about trust using the concepts of 'ordered relations' (generally, N-ary relations) and so-called *transactional associations in trust*. A transactional association representing trust is an ordered relation expressing the notion of trust as something being *emitted from* or by a particular agent (i.e., TrustEmitor) *toward* or with respect to another agent (i.e., TrustTarget). The optional *value* element of this relation in this case may reflect the level or the degree of trust (e.g., an emitor's confidence in the target).

The concept of TransactionalAssociationInTrust is restricted to such relations, in which we are able to distinguish at least two agentive entities (whether cognitive or sentient) and are able to assign each agent one of the two allowed roles. In other words, relationship of trusting assumes a TrustEmitor ("the agent that is trusting") and a TrustTarget ("the agent that is trusted"). As can be seen from this definition, we do not consider abstract statements in the form of "*This car is trusted.*" as admissible trust assertions. In other words, we argue that trust cannot be declared as a unary property of an entity (e.g., of a car); trust always needs

Table 4.2: Summary of the abstract module underlying trust formalization.

Concept	subClassOf	Definition
Agent		Something or someone that is self-contained and defined in its own right, and as such is capable of participating or being involved in situations. In AI and knowledge modelling it is an entity that is capable of observing and acting on the environment (situation), and is goal-driven (in our case capable of 'trusting' or 'being trusted').
Sentient Agent	*Agent*	An Agent that has rights but may or may not have responsibilities and the ability to reason.
Cognitive Agent	*Sentient Agent*	A sentient agent with responsibilities and the ability to reason, deliberate, make plans, etc. This is essentially the legal/ethical notion of a person. In our framework this would also cover 'referential' agents, such as software frameworks running algorithms and thus expressing (their authors') capabilities to reason, plan, etc.
Belief		Belief is a psychological state, in which an individual considers a proposition (assertion) to be true. (source: Wikipedia[4]). In our model this concept reflect the acceptance by a cognitive agent of a fact that there is a trust relationship between an emitor and a target.
Event		Something that takes place or occurres in a given situation, in terms of physical world it is something that can be observed and located at a single point in Space/Time. In our model defined for the purpose of grounding the notion of transaction later on (source: MerriamWebster[5]).
Expression		An act, instance or process of representing in a medium (e.g. as words) something that manifests or symbolizes (symbolically represents) something else. (adapted from MerriamWebster).
Feature		A prominent or distinctive aspect, quality or a characteristic of an object, situation, etc. defined e.g., for a system, a situation or an agent.
Perspective		A position, a mental view or a way of regarding situations – usually attributable to a cognitive agent.
Role		A functional part (usually in a particular) operation or a process, or (usually a socially) expected behaviour pattern, usually attributable to an agent. (extended from MerriamWebster).

someone, an agent to take an emotional or rational position toward another, target entity. Thus, the above could be transcribed e.g., into "*This car is trusted by the customers (in the US).*"

Overall structure of the transactional module is depicted by the UML diagram shown in Figure 4.5. And the key definitions are given below:

$$TransactionalPerspective \sqsubseteq Perspective$$

$$Transaction \sqsubseteq Event$$
$$Transaction \sqsubseteq \forall\ hasPerspective\ \ TransactionalPerspective$$

$$Statement \sqsubseteq Expression$$
$$Relation \sqsubseteq Statement$$

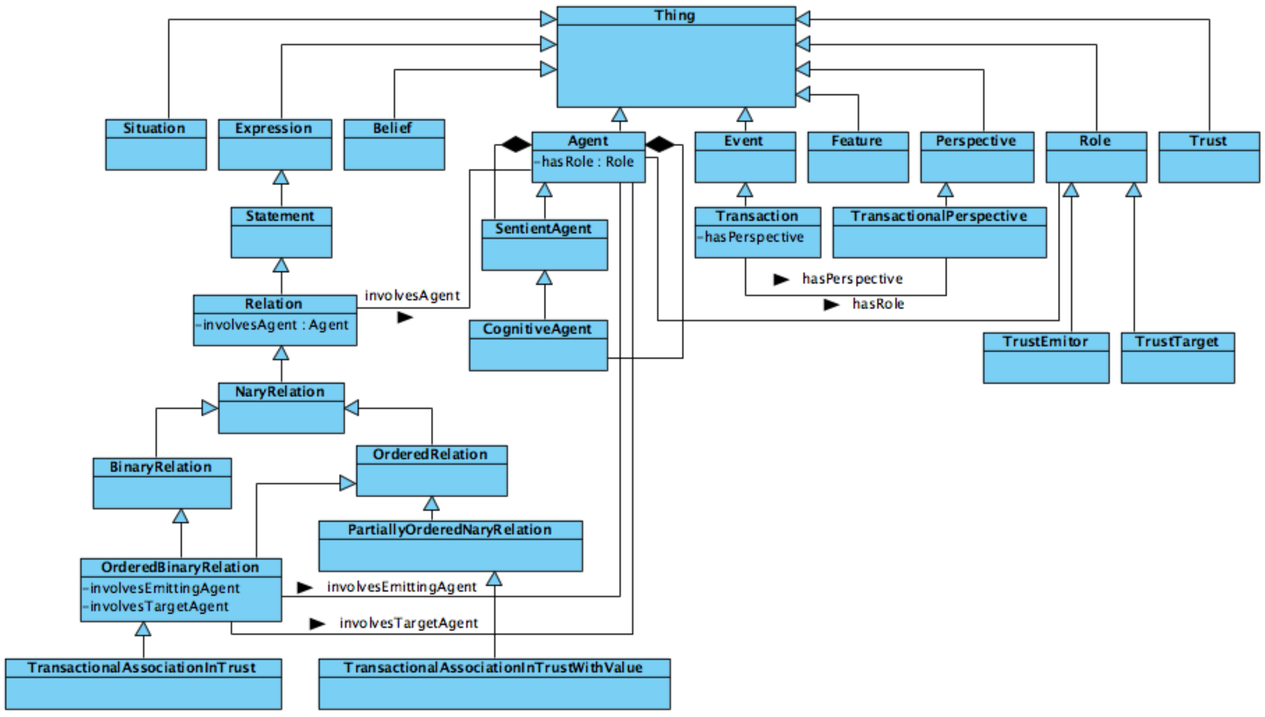

Figure 4.5: UML summary of the transactional approach to modelling trust.

$$Relation \sqsubseteq \exists\ involvesAgent\ \ Agent$$

$$\begin{aligned} NaryRelation &\sqsubseteq Relation \\ NaryRelation &\sqsubseteq\ \geq 2\ \ involvesAgent\ \ Agent \end{aligned}$$

$$\begin{aligned} BinaryRelation &\sqsubseteq NaryRelation \\ BinaryRelation &\sqsubseteq\ = 2\ \ involvesAgent\ \ Agent \end{aligned}$$

$$\begin{aligned} OrderedRelation &\sqsubseteq NaryRelation \\ OrderedRelation &\sqsubseteq\ \geq 1\ \ involvesEmittingAgent\ \ Agent \\ OrderedRelation &\sqsubseteq\ \geq 1\ \ involvesETargetAgent\ \ Agent \end{aligned}$$

$$\begin{aligned} PartiallyOrderedNaryRelation &\sqsubseteq OrderedRelation \\ TransactionalAssociationInTrust &\sqsubseteq PartiallyOrderedNaryRelation \\ TransactionalAssociationInTrustWithValue &\sqsubseteq PartiallyOrderedNaryRelation \end{aligned}$$

$$\begin{aligned} TrustEmitor &\sqsubseteq Role \\ TrustTarget &\sqsubseteq Role \end{aligned}$$

4.6 Modelling trust as systemic associations

The motivating idea behind developing this complementary perspective on trust comes from the work of NeOn partners on context. In particular, context has been broadly proposed as a modifier of semantics –

abstractly, an additional chunk of information that may be attached to a datum and as such is capable of enriching, canceling, circumscribing, or generally *altering the meaning* of the datum.

In this work on trust we took a similar stance and essentially view a situation as a potentially complex system arising around the two participants in a trusting relation. In order to ground the transactional aspect to a specific feature of a situation, we are introducing a notion of 'context' as a partial model of a situation. In line with the system theory, a model cam be seen as a schematic description of a system, theory or phenomenon that accounts for *a selection of known properties*, features and descriptive factors. Models are typically used to *partially characterize* (a certain aspect of) a system.

Our decision of introducing this systemic view is to support, in principle, the declarations about trust being explained, justified with some rationale, or being simply traced to some aspect of the situation, which the asserting agent believes may be useful for other agents to assess the validity (and trustworthiness) of the statement about trust. As we illustrated in the earlier chapters, the main point of this module is to express a range of "contexts" linking the trust transaction to concrete features of the situation. This "*context of trust*" complies with the NeOn's abstract definition of a modifier of semantics, i.e., a form of projection or mapping that assigns elements from one set to the appropriate elements (and their meanings) from another set. In our approach, context is seen as one of the models that may describe a given situation (usually from a specific perspective, e.g., 'duration of trust relation' or 'provenance of trust relation').

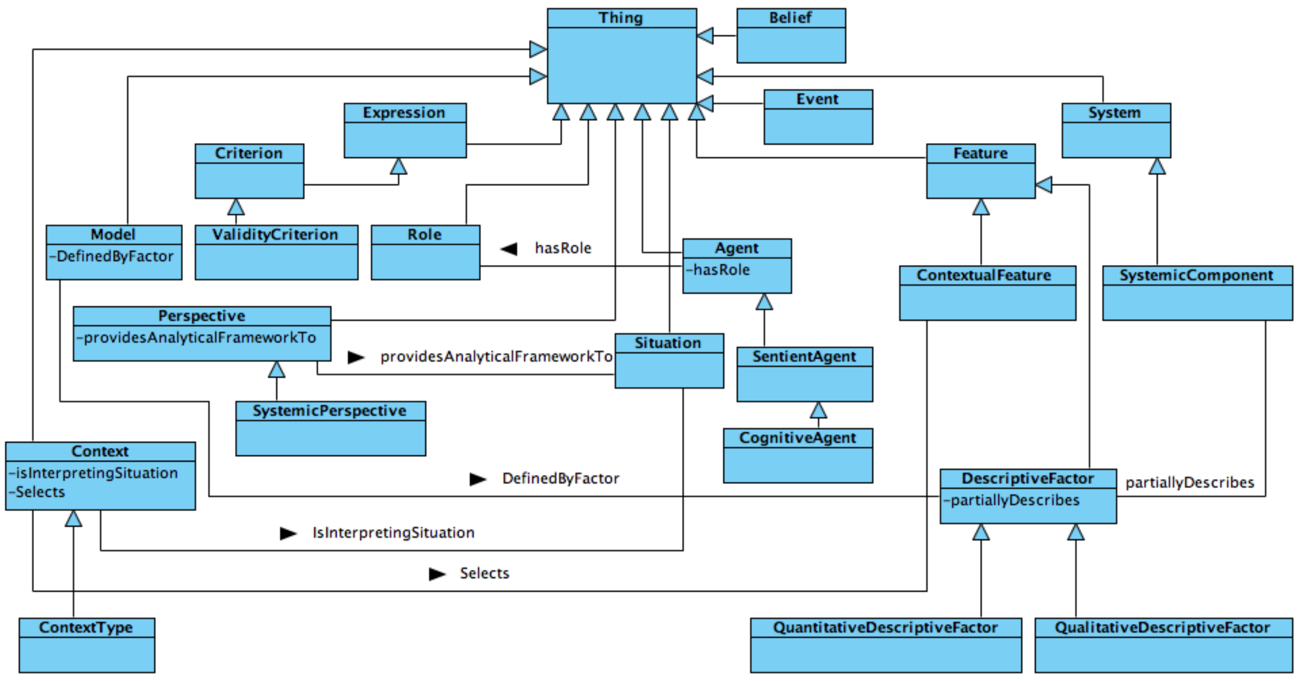

Figure 4.6: UML summary of the systemic approach to modelling trust context (intended meaning of the asserting agent).

Overall structure of the systemic module is depicted by the UML diagram shown in Figure 4.6, and below are the key definitions:

$SystemicPerspective \sqsubseteq Perspective$
$ContextType \sqsubseteq SystemicPerspective$

$System \sqsubseteq Thing$
$SystemicComponent \sqsubseteq System$

$Model \sqsubseteq Thing$

$$
\begin{aligned}
&Model \sqsubseteq \forall\ definedByFactor\ DescriptiveFactor\\
&Context \sqsubseteq Model\\
&Context \sqsubseteq \exists\ isInterpreting\ Situation\\
&Context \sqsubseteq \forall\ selects\ ContextualFeature\\
&Context \sqsubseteq\ \geq 1\ selects\ ContextualFeature
\end{aligned}
$$

$$
\begin{aligned}
&ContextualFeature \sqsubseteq Feature\\
&ContextualFeature \sqsubseteq\ \geq 1\ selectedIn\ Context\\
&ContextualFeature \sqsubseteq\ \geq 1\ definedByFactor\ DescriptiveFactor
\end{aligned}
$$

$$
\begin{aligned}
&DescriptiveFactor \sqsubseteq Feature\\
&DescriptiveFactor \sqsubseteq \exists\ partiallyDescribes\ SystemicComponent\\
&QualitativeDescriptiveFactor \sqsubseteq DescriptiveFactor\\
&QuantitativeDescriptiveFactor \sqsubseteq DescriptiveFactor
\end{aligned}
$$

4.7 Linking transactional and contextual aspects

Having introduced the two modules for understanding trust as a transaction within a certain, partially describable situation, we proceed to bringing it all together. As we mentioned earlier, the core linking concept is that of a TrustAssertion, which is defined as a specification of an assertion, i.e., of a declaration made by a cognitive agent with/within a certain belief comprising an expression that can be, in principle, evaluated as true or false. Generally assertions are made as ‘true’ with no objcetive and explicit supporting evidence required. Formally, assertions are defined as statements that may only be asserted by a cognitive agent (i.e., an agent with a belief system). Bearing the concept of assertion in mind, TrustAssertion is an assertion declaring a relationship between the source and target entity involving the notion of trust, which is generally hold true by its author.

We see the transactional aspects as being essential clauses of a trust assertion. Thus, formally our model defines a covering class labelled TransactionalTrustAssertion and states that all transactional trust assertions are considered to be trust assertions. The *transactional trust assertion* comprises both transactional association in trust with value and without value. In other words, a minimal trust assertion that is allowed by this modelling decision has the following form(s):

Direct assertion about a cognitive agent (*John: “John* $\mapsto$ *Jane”*): John believes he is trusting Jane.

Direct assertion with a value (*John: “John* $\mapsto_{v=80\%}$ *Jane”*): John believes he is trusting Jane with confidence value 80%.

Direct assertion about a generic agent (*John: “John* $\mapsto$ *TimeOntologyURI”*): John believes he is trusting time ontology on a give URI.

Indirect assertion with a value (*FlinkAlgorithm: “John* $\mapsto_{v=60\%}$ *Jim”*): Flink algorithm believes John is trusting Jim with (confidence) value 60% (in reality, based on its author’s belief, Flink algorithm “interprets relation R” among John and Jim as trust with a given confidence).

Referential assertion with a value (*John: “SumoOntologyURI* $\mapsto_{v=75\%}$ *TimeOntologyURI”*): John believes that SUMO ontology on a given URI is trusting the time ontology on a given URI with confidence value 75%.

All above examples are seen as *declarations from a cognitive agent, in which an explicit ‘trust direction’ is distinguished*; i.e., as assertions by means of which one party (emitor) can be said to trust another party (target). As shown in some examples, the *Trustee–Trusted relationship may be associated with a value* (e.g., the level of emitor's confidence in the target). According to the formal model, all trust assertions *may be contextualized* by some contextual association. This association is fully optional for a mere transactional trust assertion! However, it is usually not a good practice to omit it, as the assertion often loses its meaning, the reason why it was made by a particular cognitive agent. Informally speaking, assertions as those above are made as "take or leave" statements – since no additional information is given, it may be difficult for a rational agent to adopt such unsupported assertions of a different agent.

In order to capture at least some rationale leading a cognitive agent to make a particular trust assertion, we define so-called *contextualized trust assertion* – simply as a transactional trust assertion that is assigned at least one contextual association statement. A contextual association is a simple N-ary relation that is explicitly assigned one of the recognized context types (e.g., target authorship as a context). In other words, it is a formulation of a relationship among systemic components or within systems that has been made from a certain recognized contextual perspective. Thus, a contextual association allows the link between contexts (and their descriptive features) and generic context types. In turn, context types enable an agent to interpret a concrete context and its contextual features.

Contextual trust assertion follows the same definition as the transactional one mentioned above, with a caveat "*Trustee–Trusted relationship may be contextualized in a specific context; i.e., additional information may be represented about the situation involving trust (e.g., in order to justify, explain or support a particular transactional association between an emitor and target).* In other words, the following form(s) become possible using the contextualization add-on:

Direct contextualized assertion (*John: "John* $\mapsto$ *Jane* $|_{\forall sg \in ShoppingGoods: Jane\ isBuying\ sg \land sg \approx Food}$*"*):
John believes he is trusting Jane, when it comes to buying food.

Direct assertion with a value (*John: "John* $\mapsto_{v=80\%}$ *Jane'* $|_{\forall v \in ValueFunction: v = Accident-free\ car\ trips}$*'*):
John believes he is trusting Jane with confidence value 80%, which is measured as a ratio of all trips made by the target in which no accident has occurred.

Contextualized assertion (*John: "John* $\mapsto$ *TimeOnto"* $|_{\exists a \in Agent: TimeOnto\ authoredBy\ a\ \land\ a = Deborah}$):
John believes he is trusting time ontology, because it was authored by an agent known as Deborah.

Indirect assertion with a value (*FlinkAlg: "John* $\mapsto_{v=60\%}$ *Jim* $|_{\forall v \in ValueFunction: v = Coauthored\ books}$*"*):
Flink algorithm believes John is trusting Jim with (confidence) value 60% that is derived from a ratio of co-authored books (in reality, based on its author's belief, Flink algorithm "interprets relation R" among John and Jim as trust with a given confidence).

Referential assertion(*John: "Sumo* $\mapsto_{v=75\%}$ *Time* $|_{Sumo \sqsupseteq Time\ \land\ \exists v \in ValueFunc:\ v = Unmodified\ concepts}$*"*):
John believes that SUMO ontology must be trusting the time ontology because it imports it and reuses 75% of concepts without any modification.

The above statements formally drawing on the following definitions:

$$Assertion \sqsubseteq Statement$$
$$Assertion \sqsubseteq \forall\ assertedBy\ CognitiveAgent$$

$$TrustAssertion \sqsubseteq Assertion$$
$$TrustAssertion \sqsubseteq \forall\ contextualizedBy\ ContextualAssociation$$
$$TrustAssertion \sqsubseteq \exists\ validatedBy\ ValidityCriterion$$
$$TrustAssertion \sqsubseteq \exists\ isInterpreting\ TrustSituation$$

$$ContextualAssociation \sqsubseteq NaryRelation$$
$$ContextualAssociation \sqsubseteq \exists\ hasPerspective\ ContextType$$

$$ContextualAssociation \sqsubseteq \geq 1 \ hasPerspective \ ContextType$$
$$ContextualAssociation \sqsubseteq = 1 \ partiallyDescribesContext \ Context$$

$$TransactionalTrustAssertion \sqsubseteq$$
$$(TransactionalAssociationInTrust \bigcup TransactionalAssociationInTrustWithValue)$$

$$ContextualizedTrustAssertion \sqsubseteq TransactionalTrustAssertion$$
$$ContextualizedTrustAssertion \sqsubseteq \geq 1 \ contextalizedBy \ ContextualAssociation$$

Chapter 5

Instantiations of the Metamodel for Sample Contexts

In this chapter we provide several partial examples instantiated our generic and abstract metamodel. In particular, we show how the model proposed in the previous chapters can be applied to express trust assertions targeting ontological elements, trust assertions were ontological elements are in emitting role, and trust assertions that can be contextualized using element provenance. There are obviously many more contexts one may think of in terms of supporting trust assertions; however, it is not the point of this deliverable to exhaustively cover all possible contexts.

The whole chapter is grounded in the NeOn scenario assuming the use of multiple ontologies that come from an open and often distributed environment. In such a distributed setting, trust assertions need to be not only explicit, but should also be open to scrutiny by the users and applications. In other words, a third party wanting to reuse ontological element X will need to decide whether or s/he considers a particular trust assertion valid or not – for his or her specific purposes.

Overall, trust is essentially one of the contexts applicable to networked ontologies – a context that may provide a viewpoint on a particular ontological model. This viewpoint, in turn, may help the user to decide whether or not two ontologies go together, whether or not an axiom may be reused, etc. All this additional information may be paramount since the open world assumed by NeOn will be fraught with uncertainties and incomplete information. Thus, any additional assertions supporting a vague or incomplete model may be considered as beneficial for the user.

5.1 Trust and ontological elements

In principle we can distinguish the following situations in which some form of trust assertion may be expressed:

- Trust assertions targeted toward ontologies or ontology elements
  - Direct trust assertions made by a cognitive agent on its own behalf ("(I believe) I am trusting X")
  - Indirect trust assertions made by a cognitive agent on behalf of other agents, e.g., by observing their behaviour ("I believe John is trusting X")
- Trust assertions made on behalf of ontologies or ontology elements – these would be always indirect and, in principle, referential, as ontologies cannot be assumed to have emotional states allowing them to declare trust.
- Trust assertions made by agents and targeted toward other agent – these would be usual trust statements as made in a social situation. While these are formally possible in our metamodel, they are not discussed to any greater length in this report.

5.1.1 Trusting in ontological elements

This is the simplest form of trust assertion allowed in the proposed metamodel. From the perspective of ontologies it can also be labelled as “passive trust”, as ontologies and ontological elements only appear in a role of *TrustTarget*. Now consider the following example:

> Assume a user of an ontology developing toolkit named *alice* uses Watson search engine to discover a new OWL axiom stating that “*agent is a sub-class of an object*”. Alice decides to reuse this axiom and wants to record the fact that she trusts it.

This direct trust assertion by a cognitive agent name *alice* would therefore look as follows (using the OWL functional syntax). First, we note what is the trust assertion related to (axiom known as *ax1*, which is an instance of the *Axiom* concept defined in the OWL-ODM ontology[1]:

```
Declaration(odm:OWLClass Object)
ClassAssertion( Declaration(SubClassOf Agent Object) odm:Axiom ax1)
```

Next, the system initializes the definition of a trust assertion *ta1* on behalf of Alice – it considers Alice to be both the author of the assertion as well as the trust emitor. The target of the trust assertion is set to the axiom in question, *ax1*:

```
ClassAssertion(TrustAssertion ta1)
ObjectPropertyAssertion(assertedBy ta1 alice)
ObjectPropertyAssertion(involvesEmittingAgent ta1 alice)
ObjectPropertyAssertion(involvesTargetAgent ta1 ax1)
```

In addition to declaring her personal and direct trust, Alice interprets Watson’s selection of a particular axiom as a form of endorsement – hence, she may wish to record an indirect trust assertion for an agent named *watson*:

```
ClassAssertion(SentientAgent watson)
ClassAssertion(TrustAssertion ta2)
ObjectPropertyAssertion(assertedBy ta2 alice)
ObjectPropertyAssertion(involvesEmittingAgent ta2 watson)
ObjectPropertyAssertion(involvesTargetAgent ta2 ax1)
```

Note that although the above example mentions Alice as an explicit cognitive agent that is capable of asserting something about trust, the implementation of this in a system like NeOn Toolkit may be more straightforward. For example, by default, the toolkit may be set up so that it records any reuse of an ontological axiom or entity by a specific user as a trust assertion along the lines described above. Thus, this simple form of direct trust assertions can be achieved with a minimal involvement and cognitive overload of the user/developer!

5.1.2 Mutual trust of the ontological elements

Another scenario observed in trust assertions is so-called ‘referential’ trust; i.e., the situation when a cognitive agent interprets a relationship between two ontologies (or ontological elements) as something denoting a form of trust. Thus, consider the following example:

> Assume an ontology developer named *Alice* uses Watson search engine and finds out that SUMO ontology imports W3C_Time ontology. Alice decides to interpret this relationship between the two ontologies in terms of trust.

[1] http://owlodm.ontoware.org/OWL1.1

This situation is slightly different from the previous ones – here the agent expressing their belief, *Alice*, uses an explicit reference in one ontology to another ontology as a basis for asserting a trusting relationship. Since *SUMO* ontology cannot hold a belief toward anything, it is not possible to make a direct trust assertion. Instead, an indirect, referential trust assertion is made:

```
ClassAssertion(odm:Ontology sumo)
ClassAssertion(odm:Ontology w3ctime)
ObjectPropertyAssertion(odm:importedOntology sumo w3ctime)

ClassAssertion(SentientAgent sumo)

ClassAssertion(TrustAssertion ta3)
ObjectPropertyAssertion(assertedBy ta3 alice)
ObjectPropertyAssertion(involvesEmittingAgent ta3 sumo)
ObjectPropertyAssertion(involvesTargetAgent ta3 w3ctime)
```

This type of trust assertion made by a cognitive agent in reference to otherwise sentient (non-cognitive) objects can be extended to the core of the NeOn scenario: modularization. Imagine a module O_M is extracted from ontology *PO* (i.e., $O_M \subseteq PO$). Later, an ontology search engine may find that the same module is present in another ontology, say *X*; i.e., $\forall ent \in O_M : ent \in X$. Watson may contain an evaluating technique that would be able to spot this overlap in entity definitions and assert an indirect trust relationship between module O_M and ontology X:

```
ClassAssertion(odm:Ontology po)
ClassAssertion(odm:Ontology x)
ClassAssertion(odm:OntologyModule om)

ClassAssertion(TrustAssertion ta4)
ObjectPropertyAssertion(assertedBy ta4 watson)
ObjectPropertyAssertion(involvesEmittingAgent ta4 x)
ObjectPropertyAssertion(involvesTargetAgent ta4 om)
```

In practice, the above trust assertion would be admissible, as it contains the three minimal features (author, emitor and target). However, such an assertion looks very opaque, there is no mention in it that Alice's trust originated in or relied on the fact that SUMO imported W3C_Time ontology. In order to enrich (and actually justify) the assertion, Alice may want to *contextualize it* by means of pointing to a particular contextual association and, in turn, to a specific context. Further details on coping with contextualization is given in the next section; however, let us show the principle of a simple referential context before delving into details.

First, we need to create an instance for the axiom supporting Alice's belief:

```
ClassAssertion(
      ObjectPropertyAssertion(odm:importedOntology sumo w3ctime)
      odm:Axiom ax1
)
```

Now, all the above assertions hold as before, only a caveat would be added grounding Alice's decision in a particular context. Trust assertion is linked to a contextual association *rc1*, which is related to the whole Alice's statement *ta3*. The contextual association, in turn, enables Alice to describe an instance of class *Context*, here named as *con1* that would simply refer to our axiom *ax1* as the source of Alice's trust belief:

```
ObjectPropertyAssertion(contextualizedBy ta3 rc1)
ClassAssertion(ReferentialContextualAssociation rc1)
```

```
ObjectPropertyAssertion(involvesAgent rc1 ta3)
ObjectPropertyAssertion(partiallyDescribesContext rc1 con1)
ClassAssertion(OntologyReferentialContext con1)
```

Assume, *OntologyReferentialContext* is a simple declaration of an axiom existing in a given ontology that makes the cognitive agent to assert his or her belief. In our case, referential context *con1* would be sufficiently described by axiom *ax1* as follows:

```
DataPropertyAssertion(hasConfidence "1.0")
ObjectPropertyAssertion(selectsSource con1 ax1)
```

In principle, what Alice expressed here is her belief (or assumption) that importing W3C_Time ontology into SUMO ontology conceptually means the latter ontology ‘inherits’ all the axioms made by W3C_Time and thus, has to ‘trust’ they are correct, appropriate, etc. This is probably the simplest form of contextualizing one’s trust assertion. More advanced and more expressive forms are mentioned in the subsequent section.

5.2 Trust contextualized by provenance

As suggested in report D3.1.3 [QH08], provenance is actually a form of context that is capable of showing where different ontological statements originate on the Web and what are their sources. As the report argues, the simplest form of provenance is *data provenance* that is usually attached to the instances of an ontology. However, in the scenario of building ontologies by reusing axioms from other ontologies, provenance may prove to be a useful mechanism to express the fact that an agent’s trust for the target is grounded in the target’s origin – its provenance.

The principles of our “provenance as a trust context” instantiations are visually shown in Figure **??**. This instantiation is formally developed as extensions and specializations of generic classes in our metamodel: *Context*, *ContextualAssociation*, *DescriptiveFactor*, etc. Drawing on the OWL-ODM (as mentioned in the previous instantiations), the main class enabling the link to the ontology is *OntologyElement*. Ontology element is involved in a trust assertion, let us say in the target role.

Therefore, a specific instance of *TrustAssertion* can be created in our model, and it may be associated by an object property *contextualizedBy* to a concrete instance of a newly defined class *ProvenanceContextualAssociation*. According to the definition, contextual association must (partially) describe exactly one instance of class *Context*. In our case, we define a new class called *ProvenanceContext* (or better “provenance as a context”), which essentially corresponds to the definition made in D3.1.3, Figure 4.2 [QH08]. Hence, our provenance context class is restricted by a sub-property called *selectsSource* that points to an instance of an abstract class *Source*. For sake of simplicity, let as assume that *Source* is equivalent to class *Object* (note that in D3.1.3 the source is restricted to InformationObject).

This treatment of provenance would easily allow us to define e.g., an *AuthorshipContext* as a sub-class of *ProvenanceContext* with a single additional restriction demanding that *selectSource* object property is restricted to instances of class *Author* (which we defined as equivalent to *CognitiveAgent*) rather than to any object.

Data property *hasNumericValue* has been specialized into *hasConfidence* and *hasRelevance*, which both can be assigned to an instance of class *ProvenanceContext* (or *AuthorshipContext*) – precisely in line with the proposal in D3.1.3.

Now consider the following example:

> Assume a user of ontology developing toolkit named *alice* used Watson search engine to discover a new OWL axiom stating that “*agent is a sub-class of an object*”. Alice decides to reuse this axiom and wants to record the fact that she trusts it, because it originates in the SUMO ontology with *uri* (URI of this source has been provided by Watson).

Contextualized trust assertion would therefore look as follows (using the OWL functional syntax):
First, we note what is the trust assertion related to:

```
Declaration(odm:OWLClass Object)
ClassAssertion( Declaration(SubClassOf Agent Object) odm:Axiom ax1)
```

Next, the system initializes the definition of a trust assertion *ta1* on behalf of Alice – it considers Alice to be the author of the assertion as well as the trust emitor. The target of the trust assertion is set to axiom *ax1*:

```
ClassAssertion(TrustAssertion ta1)
ObjectPropertyAssertion(assertedBy ta1 alice)
ObjectPropertyAssertion(involvesEmittingAgent ta1 alice)
ObjectPropertyAssertion(involvesTargetAgent ta1 ax1)
```

Trust assertion *ta1* is contextualized by means of creating *pc1*, an instance of provenance contextual association:

```
ObjectPropertyAssertion(contextualizedBy ta1 pc1)
ClassAssertion(ProvenanceContextualAssociation pc1)
```

Next, the system needs to refer in the contextual association to the object/agent involved in a given statement (in our case, it is axiom *ax1*) and asserts that this contextual association partially describes provenance context, say *con1*:

```
ObjectPropertyAssertion(involvesAgent pc1 ax1)
ObjectPropertyAssertion(partiallyDescribesContext pc1 con1)
ClassAssertion(ProvenanceContext con1)
```

Finally, the actual provenance is defined using a specific confidence value ("*0.8*") and pointing to a specific instance of a source, in our case *uri1*:

```
DataPropertyAssertion(hasProvenanceConfidence "0.8")
ObjectPropertyAssertion(selectsSource con1 uri1)
ClassAssertion(Object uri1)
```

Should another user, Bob, later decide to support the existence of the above axiom *ax1* e.g., by endorsing it and supporting it with his knowledge of an author, a new trust assertion, say *ta2* would be made, which would follow the same steps as described above, only using a different grounding for object property *selectsSource*.

```
ClassAssertion(TrustAssertion ta2)
ObjectPropertyAssertion(assertedBy ta2 bob)
etc.
```

If, however, Alice decided to extend her endorsement, e.g., by point to *universityZ* as an author of the above axiomatization, it would be sufficient to assert another contextual association:

```
ObjectPropertyAssertion(contextualizedBy ta1 ac2)
ClassAssertion(ProvenanceContextualAssociation ac2)
ObjectPropertyAssertion(involvesAgent ac2 ax1)
ObjectPropertyAssertion(partiallyDescribesContext ac2 con2)
ClassAssertion(AuthorshipContext con2)
etc.
```

One side effect of the formalization as shown in our walkthrough is the possibility to also express trust assertion targeting a statement made by another user. For example, user Carol may easily (and explicitly) declare her trust in any of Alice's statements (and even in Alice herself). In other words, Carol may state she trusts Alice's judgment on trustworthiness of the axiom *ax1*, but as easily Carol may state that she also trusts (endorses) the provenance context *con1* mentioned by Alice, which points to a particular source. Obviously, each of these assertions may be further contextualized:

```
ClassAssertion(TrustAssertion ta3)
ObjectPropertyAssertion(assertedBy ta3 carol)
ObjectPropertyAssertion(involvesTargetAgent ta3 ta1)
```

or

```
ObjectPropertyAssertion(involvesTargetAgent ta3 con1)
```

or

```
ObjectPropertyAssertion(involvesTargetAgent ta3 alice)
etc.
```

What can all this be used for? Ontologies and conceptual modelling is typically a domain of specialist engineers. Thus, declaring whether or not a particular ontological entity can be trusted based on the conceptual grounds is likely to be left to those who understand the syntax, the domain, the perspective, etc. Nonetheless, a declaration from knowledge engineer Joe Bloggs may not be very helpful for an ordinary user. Hence, providing e.g., a provenance context for a given entity may allow the user to draw upon a form of contextual reasoning to infer something about trustworthiness of the entity based on its origins.

One can think of this scenario as a validation performed by agent *C* (consumer) of a trust assertion authored by agent *P* (provider). Whereas our consumer *C* may not be able to assess the validity of trust assertion in a form "$P \mapsto ClassX$" based on its face value, s/he is usually able to declare his or her trust relation toward a particular source, say $C \mapsto uri_1$ or $C \mapsto white\ paper_2$. Hence, if consumer *C* decides to trust particular sources, contextual reasoner may allow for 'transitively inheriting' his/her trust to all those object that claim a given source as their provenance:

$$C \mapsto uri_1 \ \ \wedge \ \ P \mapsto X \mid hasProvenance(X, uri_1) \implies {}^{\dagger}\, C \mapsto X$$

† by virtue of C trusting *all* entities in a given source

5.3 Other types of context

Another context mentioned in D3.1.3 was argumentation. One can immediately think of a scenario where a trust assertion about an ontological entity would not be supported merely by its provenance or authorship, but a more expressive mechanism would be required. Arguments are one such expressive framework that may be able to capture not only origins or outcomes of a particular decision, but also the entire process leading to the decision. This argumentation process may, in turn, contain may statements in support or against a particular course of action, and as such may be much more informative for a user who wants to validate a trust assertion contextualized by argumentative structures.

Reusing the model of argumentation proposed in D3.1.3, Figure 4.3, one may actually see ArgumentationContext as a super-class of simpler contexts, such as Provenance or Authorship mentioned in the previous section. For example, by selecting the *User* entity of the argumentation model, one is emulating the *AuthorshipContext*. By selecting an entity e.g., of type *Example*, we may replicate features of the *ProvenanceContext*. Fully fledged *ArgumentationContext* may then be achieved by associating it with the *SolutionProposal* node of the model.

2006–2008 © Copyright lies with the respective authors and their institutions.

Chapter 6

Modelling Access Control

As more and more semantic web applications contain and produce sensible data, most of them are based on some collaborative platform. Hence, it is often essential to control to whom, to what degree and when data are disclosed. In this chapter we propose an access control model to answer the needa of access control identified in earlier reports [DGA$^+$07]. The first section introduces a brief use case which will support the explanation of the model. This use case comes from the NeOn work package 8 [GPALC07]. The section 6.4 will present the requirements justified by this concrete use case. In the following sections, the conceptual model for access rights and their representation will be described. We will conclude this chapter with explaining how our use case features in the proposed access right model.

6.1 Motivation 1: content sensitivity

A traditional situation that has originally led to investigating access rights and authorities comes from the requirement of individuals or organizations to preserve certain *confidentiality* due to sensitivity of data content. Confidentiality can be seen here as a doubt about someone's trustworthiness, or perhaps a decision to trust someone only with certain tasks, objects, etc. In a single-user, isolated environment, any concern for confidentiality is somewhat paranoid – one usually does not need to prevent him- or herself from accessing what they created. However, such single-user, fully isolated environments are become increasingly rare. In reality, many of us share resources with others and/or operate within a networked, connected environment, which enables physical access to both, known and unknown subjects.

Content sensitivity issue relates not only to such extreme cases as preventing any internal communication from leaking outside of corporate Intranet. In fact, it is usually far more important to limit access *inside* Intranets, often as much as toward the outside world. In today's business we can easily picture a situation where several stakeholders contribute e.g. to different facets of an Enterprise Information Management System with databases and possibly knowledge bases specialized in financial reporting, procuring, customer management, planning, production management, design, etc.

Often, individual employees from different units in an organization need to access the same shared dataset (e.g. a budget or a strategic plan). However, one should only acquire as much information, data, resources as they truly need to carry out their duties. Thus, chassis production line supervisor needs to be aware of only that part of budget that relates to his or her area of responsibility. Thus, with the exception of budgeting for chassis production, s/he should not even be aware of how other budget chapters are organized, distributed, and obviously instantiated with real numbers.

Such a situation can be transposed onto ontologies and their content. Organizational information systems, such as those discussed in [**?**] in the context of managing the flow of financial and invoicing information, are often modular. Modularity is achieved by introducing blocks responsible for different parts of the business process. Now, we can imagine these different business processes use different languages, concepts, and relationships, and as such could be, in principle, conceptualized as a series of ontologies and/or knowledge bases (KB) of varying size and complexity. As is common with business processes, certain information is restricted to selected individuals. Even if several people access the same information, they may see different versions of it, different level of detail, sensitivity, and similarly. Hence, it makes sense to replicate this situation and to create a number of partitions on the ontological model describing such an organizational information system. This can be achieved by expressing access authorities to the individual ‘modules’. One advantage of not just *denying access at the request stage* but rather *removing certain modules from the large system/ontology/KB* is in the content confinement. The difference between the two is as follows:

denying access at the request : If our line supervisor requests the system to show him/her next year’s budget, the system may come back with a list of chapters and sections (folders, files,. . .). Then upon attempting to access a given node, the system would ask for authentication and possibly deny the access. However, our line supervisor had already learned a lot about the way the budget is structured; s/he saw at least a partial schema, which is not what s/he really needs to do his/her job. Hence, we have got here an information and security leak.

removing content module(s) : In the same situation of requesting next year’s budget, our line supervisor would obtain only those parts of the budget that s/he has access rights to see or work with. In other words, in this case ‘*next year’s budget*’ is equivalent to the ‘*part of the next year’s budget accessible to me*’.

As the two strategies above show, it is rather difficult to prevent subjects from requesting an entity they believe has to be in the system. Although they only truly see what they are entitled to, there is much more information provided to them than necessary. In the latter case the request for the whole budget still took place, but our subject did not obtain any more information from the system than what s/he was entitled to.

A similar line of thought might be elaborated for ontologies and instances of the concepts within ontologies. On the level of company, there may exist one large organizational ontology (or perhaps a network of smaller organizationally-focused ontologies), whose purpose is to conceptualize the domain. However, on the level of individual subjects the conceptualizations and conceptual views are contingent on their *authority*, on *data sensitivity*, on the *confidentiality* requirement, on the organizational *policies*, etc.

Hence, subject called (say) *Researcher* would be entitled to see in the conceptualization of the organization *KMi* its projects, its staff members, its publications, and the relationships among these entities (e.g. person X leading project Y). However, this subject may not be permitted to see what vocabulary and structures are used in *KMi* to manage salaries and contracts. On the other hand, subject (say) *Administrator* would be entitled to see but not edit the research aspects, and in his or her conceptualization there would be additional modules to express statements about people’s salaries, contracts, etc.

Thus, the implication of this scenario on trust modelling can be summarized as follows: The notion of access as described in the scenario carries a certain relational component (between the accessing object and subject being accessed). In addition to this core component there is also an important contextual or explanatory component, whose role is to capture (i) how a particular relationship started, (ii) what are operational consequences of working with such a relationship, and (iii) how does the relationship reflect the actual world or system being modelled. In other words, such an access control statement can be seen as a specific type of a trust assertion declared in a specific (organizational) situation.

6.2 Motivation 2: collaborative ontology (re-)engineering

In contrast with the first motivational example, where we had top-down imposed restrictions on resource due to their sensitivity, this motivation considers another fairly common pattern of organizational life. Imagine a situation where an organization manages an ontology (or a network of ontologies) on a particular aspect related to the organization’s interest (say fisheries or medical drugs) – let us label this ontology as O_{shared}. Due to latest developments in the field, subject Alice starts conceptualizing a new chunk of knowledge that leads her to introducing and/or amending certain entities in that existing ontology. Let us denote this chunk as M_{alice}.

Obviously, Alice may want to see her changes as if they were already part of the official ontology; e.g. to do some tests. She wants to work with the O_{shared}, however. What she may thus do is committing her amendments in M_{alice} into the official O_{shared} ontology. Now there are two things that can happen:

versioning strategy : Alice spins off a branch of the official O_{shared}, which only she can see, and commit her new module M_{alice} to that branch. Other colleagues continue working with the main branch of O_{shared}.

access control strategy : Alice does her amendments directly in the official O_{shared} because she is normally permitted to do such changes, but she is not yet happy with the current state of her module M_{alice}. Hence, she defines ontology O_{shared} as including module M_{alice}, but she would simultaneously restrict the access to M_{alice} to herself.

Since this is a work in progress and perhaps a minor conceptual amendment (e.g. translation of term), Alice feels she may work on the official branch/version of O_{shared}. She only needs that 'lock' on her new additions temporarily – imagine she only wants to consult with Don whether a particular translation is correct or not.

What we are getting here, is a situation where there is a shared ontology O_{shared}, which is used by Alice and many of her co-workers for very specific purposes and tasks. For sake of this example, assume that Bob is among those co-workers who use O_{shared} and Don is among those who cannot access the ontology in question. We now have one ontology (which includes Alice's new module M_{alice} among other aspects), but our three users see three different things:

Alice sees $O_{shared} \equiv O_{shared} : M_{alice} \in O_{shared}$;

Bob sees $O_{shared} \approx O_{shared} : M_{alice} \notin O_{shared}$;

Don sees $O_{shared} \approx \oslash$ (i.e. empty set)

Now Alice needs Don's advice on some issues in her module. In a physical collaborative setting, Alice would simply visit Don in his office and ask for his opinion. In the virtual setting, this can be emulated by Alice granting Don a temporary authority to access her module M_{alice} (or possibly $M_{smaller} \subset M_{alice}$ or even M_{larger}: $M_{alice} \subset M_{larger} \subset O_{shared}$). As a result, Don will be able to access a particular module (say, M_{larger}), but not the whole ontology – so there is no major breach of confidentiality here. Alice may even restrict Don's actions to commenting and annotating the content in M_{larger} but not add or delete anything.

In the meantime, nothing whatsoever changed for Bob (and any other co-worker) – they still interact with the original, sanitized O_{shared} which hides any of Alice's changes and Don's annotations. At some point Alice and Don reach agreement, and Alice feels confident that now she may make an official proposal for the *extension of* O_{shared}. She simply 'cuts' Don's temporary authorities, and enables her co-workers to obtain authorities appropriate to a specific extension protocol, workflow, or business process. Only at this stage would Bob and other become aware of a change; however, this would already be presented as an officially different ontology – say, $O_{shared}^{extensionproposal} \equiv O_{shared} \oplus M_{alice}$.

How we can realize such a scenario is shown later, in section **??** using an example of wiki as a factory for collaborative engineering of ontologies. The examples include screenshots that would make the abstract notation used in this section easier to understand.

Again, as above there is a certain implicit notion of trust in this scenario too. Unlike the first scenario that explicitly tried to maintain confidentiality and secrecy, this scenario attaches the notion of trust to the agent's role in the target lifecycle. Agents are trusted to make useful comments, suggestions or modifications, and based on this interpretation they are or are not allowed to access a particular resource.

6.3 Use case: Managing invoice work flows

The use case described in this section comes from a NeOn scenario partner and draws on the work described in [GPALC07]. We pick the scenario named “invoice work flow scenario”, which needs access rights only for the ABox part of sharedontologies. The choice of this scenario to illustrate our model was made because of the easy translation of the access right needs from the ABox to the TBox.

In the invoice workflow scenario workers participate in the role of “Users” of the invoicing application. This role is divided in three different sub-roles: “Receiver” of the invoices, “Emitter” of the invoices, and “Decision Taker”. In this scenario no member of any of these roles may have the right to modify any part of the ontology schema (TBox). On the other hand, workers with the role of “Administrator”, “Emitter” and “Decision Taker” shall be able to create and modify instances of the ontologies (i.e., the ABox). Furthermore, a worker in the role of “Receiver” has no rights to modify any instance but has reading and viewing rights.

In the invoicing workflow these different access authorities need to be appropriately combined. First, the “Emitter” of invoices (and possibly an “Administrator”) creates an invoice (i.e., an instance of concept *Invoice*. To transfer this task to another person would usually not be advisable due to a sensitive nature of information that any invoice contains. Only a few people in the company are normally permitted to create and emit invoices, as these are legal, contractual documents. Moreover, in the case of larger companies, emitters may be assigned to particular clients; hence emitter Alice can raise invoices for (say) all pharmacies, but not for wholesale intermediaries and warehouses. This organizational policy shall be reflected in the access rights, whereby emitters are authorized to amend only certain branches or parts of the ontology.

The reception of invoices can be performed by two types of users. First, a user in the “Decision Taker” role usually has rights to receive and accept/reject an invoice. The user in the “Receiver” role may also have the capability to receive invoices, but (in some cases) such a user may not be permitted to accept/reject them. This acceptance may be performed by users with the “Decision Taker” role (or possibly “Administrators”). In the ontological language, different capabilities may be reflected by stating what particular *operations on resources* are assigned to specific authority keys.

The role “Decision Taker” may be able to transfer their invoice approval capability to other workers (i.e., receivers). Obviously, in some cases the *authority delegation* may involve only a part of their acceptance rights, in order to satisfy the principle of minimal necessary authority (as mentioned in [DGA$^+$07]). In other words, the receivers may be permitted (by means of using delegated authority keys) to receive and automatically approve invoices, e.g., for a certain client, up to a certain monetary value, for a particular product, etc. Also, an administrator of invoices may delegate some responsibility to other users for receiving the invoices and perform the actions needed – whether temporarily or in a longer term.

6.4 Requirements for the NeOn access control model

In the deliverable [DGA$^+$07] we determined what must be the key point of your model. Here we summarize the core of those requirements as follows:

1. *Access rights models in a role of collaboration enabler* – the core of this issue is that a subject should be able to grant a selective access to any particular object resource to other collaborating subjects; then revoke this access and manage the whole process of delegation/revocation on an appropriate level of granularity;

2. *Access control models in a distributed environment* – the rationale for this requirement stems from the fact that it is usually easier to manage access to a centralized repository of object resources; however, an access control model supporting ontologies shall be able to cope with distributed information and not necessarily enforce a central control;

3. *Access control models as accountability mechanisms* – in addition to managing the actual access to the shared object resources there are other issues the users of an information system may be

interested in. For instance, in case of ontologies, two features that often have significant role in using and reusing the ontological commitments include the provenance and trust in the shared resource. This is particularly important problem to be addressed, because ontologies act as referential conceptual frameworks, models that facilitate collaborative work. Hence, such issues as audit trails, logs and traceability become more important (e.g., to ensure conceptual soundness of the resources, as well as to detect or re-create access violations);

4. *Access rights control not disrupting the functionality of common tools* – the core rationale for including this issue into our discussion of access rights to ontologies relates to the nature of ontologies; they are primarily referential frameworks that may be used in a wide range of tools - hence, it is desirable to achieve managed access without necessarily demanding alterations to the existing tools.

Further details, including explanation and rationale for each of these issues can be found in our earlier report [DGA$^+$07]. This previous report also contains rather extensive review of various common access control models, such as MAC [1], DAC [2] or RBAC [3], hence, we will not review them again.

6.5 Conceptual model for access rights

The conceptual model of access control that we propose in this chapter extends the OMV [4] presented in [PHGP$^+$06]. It takes the form of an ontology which will be described in section 6.5.1. Later on, we will explain how granting, revoking and delegating of access rights can be achieved in our model.

6.5.1 Description of the model

To make the comprehension easier to the reader we propose a translation of this ontology in UML format, presented in Figure 6.1. An OWL file is accessible online at http://isweb.uni-koblenz.de/ontologies/2008/10/accessRight.owl. To help with the model behaviour representation we use rules in addition to the ontology:

Rule 1 : Every agent starts with no rights except the agent who has all the rights for his own data (we will call such an agent the owner of the rights).

Rule 2 : If there are two rights contradictory given by two agents who are mutually equal, the most recent assertion will be considered;

Rule 3 : No agent can delegate a right to him/herself.

Next, we will present the different concepts that comprise the ontology and explain them.

Class Entities

Access

Access is a specific *Action* which describes how an *Agent* may access a particular *Resource*.

$$Access \sqsubseteq \forall\ actionOn\ \ Entity$$
$$Access \sqsubseteq Action$$

[1]Manadatory Access Control
[2]Discretionary Access Control
[3]Role-Bases Access Control
[4]OMV stands for Ontology Metadata Vocabulary

AccessRight

AccessRight is the capability to either deny or allow an *Action* on a specific *Resource* to a certain *Agent*. Each *AccessRight* contains an emission date of this right. If there are contradictory rights in the system for a specific *Agent* and the same *Resource*, then the latest right is consider as the valid one (i.e., the one with the latest emission date):

$AccessRight \sqsubseteq \geq 1\ hasAction\ Access$

$AccessRight \sqsubseteq Right$

AccessRightSituation

AccessRightSituation is a *Situation* in which an *Agent* (i.e., a situationActor) gets a *Right* to access something (the accessedEntity) for a certain *Action* (hasAction). The *rightSituationTime* represents the time of emission of this *Situation*. The actor, accessedEntity, and rightSituationTime must be always determined:

$AccessRightSituation \sqsubseteq RightSituation$

$AccessRightSituation \sqsubseteq =\ hasAction\ Access$

$AccessRightSituation \sqsubseteq \geq 0\ rights\ AccessRight$

Action

An action represents the different possibility for an *Agent* to interact with the system.

Add

Add is a fundamental access which can be applied only to the *Entity* of the type *Container*.

$Add \sqsubseteq Access$

$Add \sqsubseteq \forall\ actionOn\ Container$

Agent

$Agent \sqsubseteq Entity$

Container

Delegation

A delegation is an *Action* involving an *Agent* and another *Action*.

$Delegation \sqsubseteq Action$

DelegationRight

It is a right to delegate a specific *Right* to another agent. The *Right* can be another *Delegation*- or *AccessRight*. We will propose a better overview of this mechanism later in this chapter (see section 6.5.2).

$DelegationRight \sqsubseteq =\ hasAction\ Delegation$

$DelegationRight \sqsubseteq Right$

DelegationRightSituation

A *DelegationRightSituation* is a *Situation* involving two *Agent*s (the actor and the delegate). This *Situation* is instantiated when a delegate receives a right from an *Agent*.

$DelegationRightSituation \sqsubseteq \; = hasAction \; Delegation$

$DelegationRightSituation \sqsubseteq \; = rights \; DelegationRight$

$DelegationRightSituation \sqsubseteq RightSituation$

DenyAccessRight

It denies a certain *Action* for a specific *Resource* to a certain *Agent*. Each *DenyAccessRight* must have exactly one resource for one agent and at least one resource.

$DenyAccessRight \sqsubseteq AccessRight$

$DenyAccessRight \sqsubseteq \neg \; GrantAccessRight$

Entity

GrantAccessRight

It allows a certain *Action* for a specific *Resource* to a certain *Agent*. Each *AllowAccessRight* must have exactly one *Resource* for one *Agent* and at least one *Resource*.

$GrantAccessRight \sqsubseteq AccessRight$

$GrantAccessRight \sqsubseteq \neg \; DenyAccessRight$

Read

Read is a fundamental access that results only in the flow of information from an *Entity* to an *Agent*.

$Read \sqsubseteq Access$

Remove

Remove is a fundamental access which is defined as the opposite action of “Add”, it can be applied to the types of *Entity, Container*.

$Remove \sqsubseteq Access$

$Remove \sqsubseteq \forall \; actionOn \; Container$

Resource

Right

$Right \sqsubseteq \; = rightActor \; Agent$

$Right \sqsubseteq Thing$

$Right \sqsubseteq \; = creationTime$

$Right \sqsubseteq \; \geq 1 \; regardingResource \; Resource$

RightSituation

RightSituation is a *Situation* concerning *Right* which really happens. This *Situation* can be either a *Delegation-* or an *AccessRight-Situation*.

$$
\begin{aligned}
RightSituation &\sqsubseteq\ = accessedResource\ \ Resource \\
RightSituation &\sqsubseteq\ = situationActor\ \ Agent \\
RightSituation &\sqsubseteq\ \geq 1\ \ rightSituationTime \\
RightSituation &\sqsubseteq\ Situation
\end{aligned}
$$

Situation

It is a view on a set of entities. It can be seen as a ‘relational context’, reifying a relation. For example, a PlanExecution is a context including some actions executed by agents according to certain parameters and expected tasks to be achieved from a Plan; a DiagnosedSituation is a context of observed entities that is interpreted on the basis of a Diagnosis, etc.

Update

An update is a fundamental access that modifies the state of a *Resource*.
$Update \ \sqsubseteq \ Access$

Annotate

Annotate is a fundamental access that creates an annotation of a *Resource*.
$Annotate \ \sqsubseteq \ Access$

Instantiate

Instantiate is a fundamental access that result of the creation of a new instance of a concept.
$Instantiate \ \sqsubseteq \ Access$

6.5.2 Granting, Revoking and Delegate rights in our model

How to grant a right

We assume in the initial state of the system that nobody has any rights on any resources. As shown in the previous figure, there are two kinds of right – an *AllowAccessRight* and a *DenyAccessRight*. To grant a right the user has to instantiate one of these two concepts and make sure that one *Resource*, one *Agent* and at least one *Action* is defined.

How to revoke a right

The right forms a chain of delegation from the owner of a right (*Agent*) to the user (the revoked *Agent*). Only the *Agents* which are positioned higher in the chain can revokes his right. For example if Alice delegate rights to Bob for a resource A. He delegates some other rights for A to Carol and Carol did the same to David. Today Carol leave the organisation and she lost all her right. In this case only Alice and Bob may revoke the right of Carol, David cannot because he get his right for the resource from Carol. For revoking her rights either Alice or Bob create an instance of *DenyAccessRight* with the actor Carol for the resource A with the current time. We already mention if two contradirectory rights co-exist in the system only the more recent can be apply.

2006–2008 © Copyright lies with the respective authors and their institutions.

How to delegate a right

The delegation is a fundamental mechanism which grants the controlled propagation of the access right. This mechanism is controlled because each time the user does a delegation, s/he can specifiy which resource, by whom, and how to access it. As shown in Figure 6.2, Alice who is the owner of a right to the resource Apple wants to cut it (i.e., action = update). Unfortunately she broke her arm a few day ago. She need some help and therefore she delegates to Bob the right to delegate the right to *update* her *Apple* to *Carol*. At the same time Alice does not want that Bob can see (read) or do something else to the Apple. This simple example shows how our model helps the owner of the right to a resource to keep the control of it.

6.6 Model instantiation with the use cases

Let us now extend the use case we presented in section 6.3. Alice and Bob are two emitters of invoices. They are in charge of two different customer branches, respectively for pharmacies and for wholesale warehouses. Alice and Bob do not have any access to the work of each other, but they both have the right to delegate any access right to any person who has the role of emitter for the resources for their respective customers. Assume now that Alice falls ill for two weeks and when she finally comes back to work, she finds many messages from her customers asking to get their invoices as soon as possible. Alice needs help to achieve that work and decides to give temporarily the right to read and update her customer resources to Bob. Thus, Bob can access those resources, he can do a part of Alice's work. After a while the work of Alice is updated and she may therefore revoke the right that she has given to Bob earlier. Figure 6.3 shows how the example works in our model.

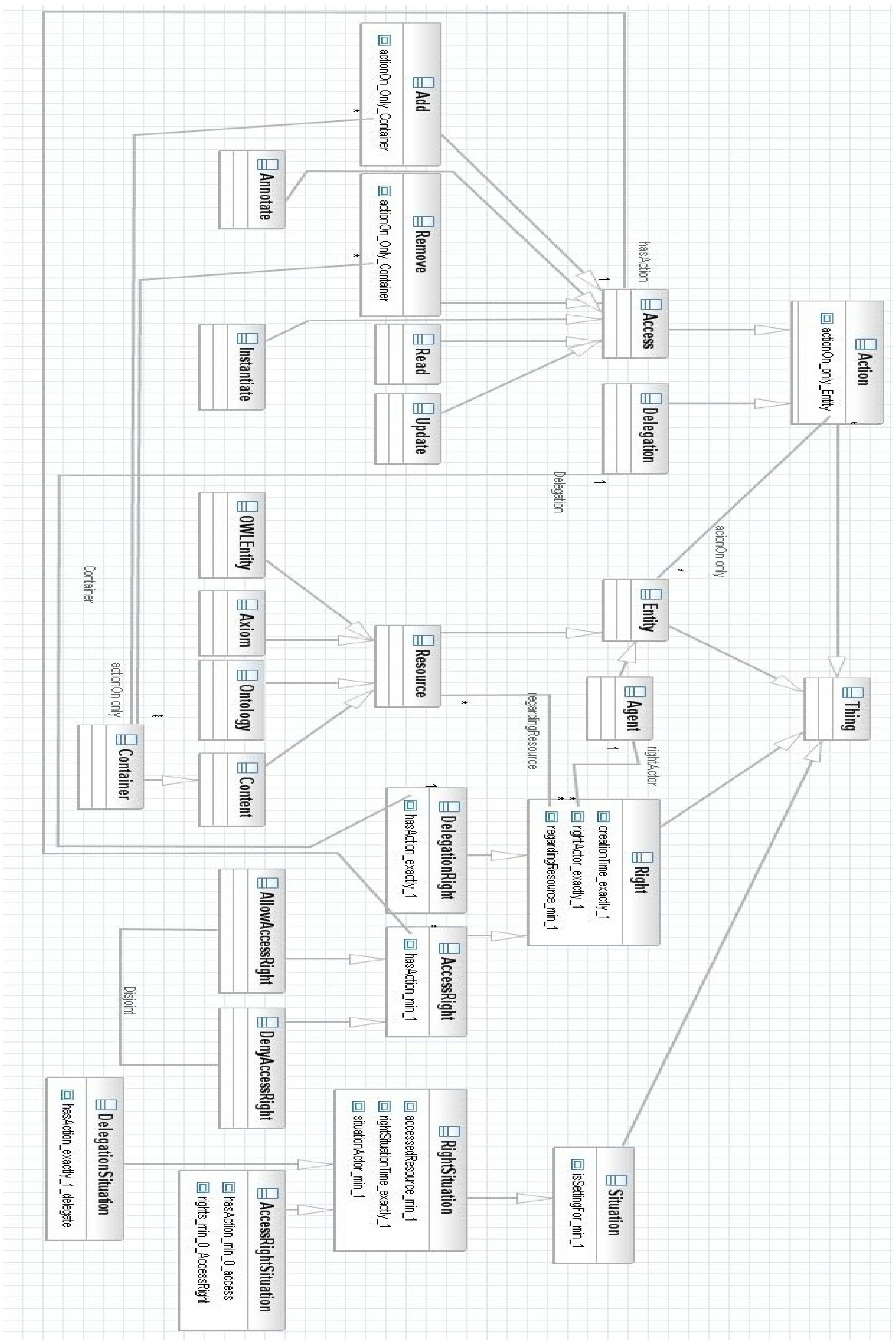

Figure 6.1: The conceptual model of access control for the “NeOn scenario”

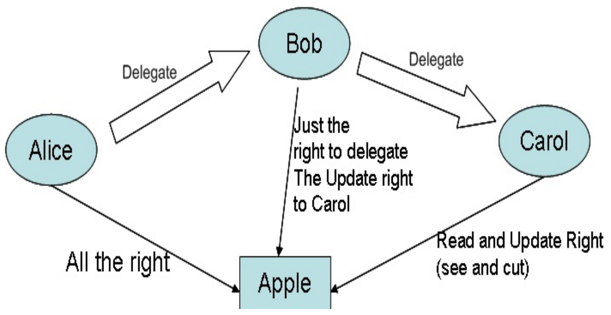

Figure 6.2: The mechanism of right delegation

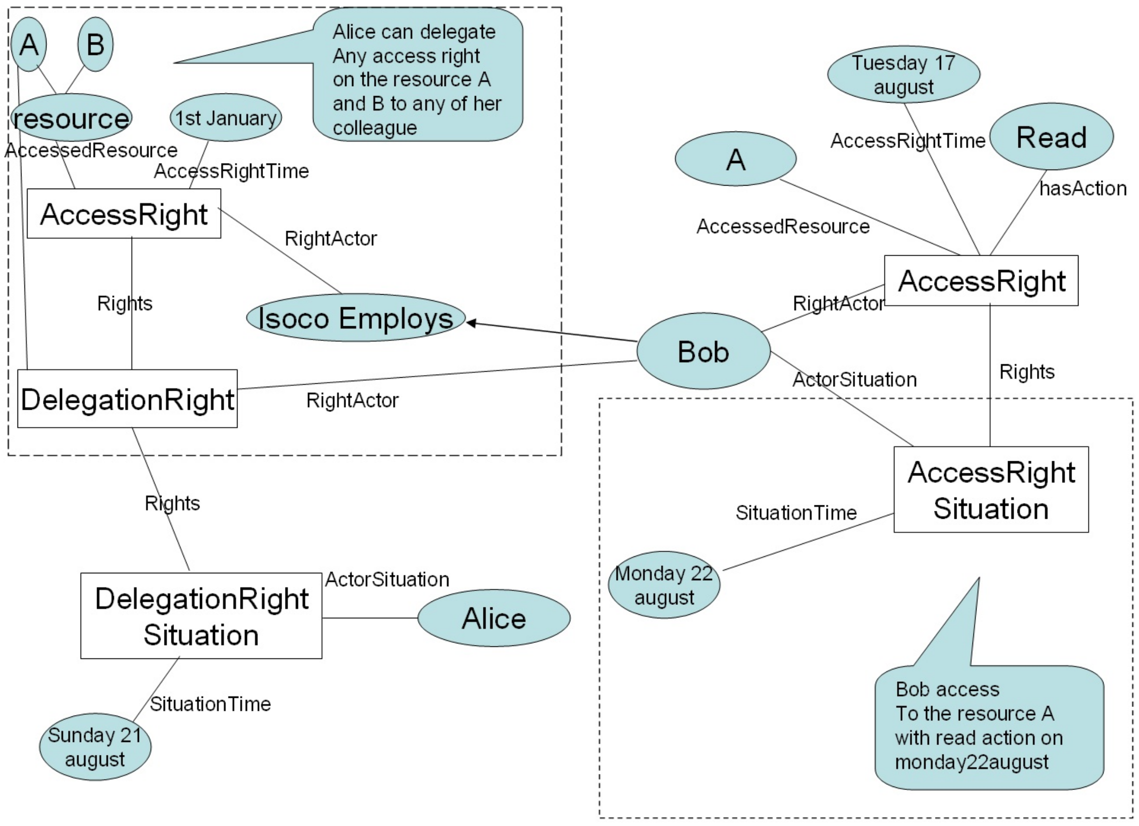

Figure 6.3: The use case instantiation

Chapter 7

Conclusions

In this deliverable we proposed an approach to modelling various situational aspects that may lead a cognitive agent (usually, a human being) to make an assertion about the trust relationship(s) in a given situation. The deliverable perceives trust as one of the contextual modifiers that may be attached to ontologies and ontological elements; essentially, with a purpose of explicitly disclosing some emotional or rational processes inside an agent that are ‘manifested’ as trust.

The approach we emphasized here is agent-driven – i.e., trust assertions are made, declared by cognitive agents, often ontology users or developers. In our opinion, this is actually a very natural way of treating trust, since it complies with the philosophical assumptions of trust being attributed to a system of beliefs. However, we should note that the fact that a declaration is made by a cognitive agent does not prevent our model from expressing multi-way trust relationships. These include agent – agent, agent – object, and even object – object assertions.

Obviously, with such a multitude of trust party pairings, it may become difficult to reason with trust. To this extent we proposed the notion of “contextualized trust assertion”, i.e., an assertion that is linked to additional information that may alter (usually only clarify and justify) the meaning of the trust assertion. These contextual features may actually serve as validation points enabling a third party to decide on whether a particular assertion s/he encounters in the open world environment shall be considered valid (and thus adopted by that third party).

The expressive richness of the proposed model will also enable a contextual reasoner (not yet developed for trust) to decide on when and how different reasoning rules may be applied – e.g., when two trust statements are transitively linked and when transitivity would lead to inconsistent or incoherent inferences.

In a similar tune, we were not attempting to formally include the access control model fully into the formal NeOn model of trusting in networked ontologies. Nevertheless, initial discussions took place and also the work on generic modularization has already been included into the NeOn Model. All this forms a good groundwork for further elaboration, formalization, and ultimately creation of an extension of the proposed metamodel of trust that would handle at least the most common aspects related to access management. The core idea here, mentioned in the deliverable, is *treating access control as a contextual side-effect of a trust assertion*, which allows us to merge the two approaches and two tasks into one broader framework.

One frequent comment we received from the use case partners in the past was regarding the operational implementation of our proposals. While the actual extensions to the NeOn Toolkit are subject to some future work, the bulk of the trust management, acquisition and representation has been proposed with a NeOn Toolkit and NeOn Infrastructure (in particular, Watson) in mind. This is documented in our walkthrough examples that substantially rely e.g., on ontology discovery by Watson or partial reuse of an existing axiom in developing new ontologies. Individual sections in chapter 5 made a conscious attempt to also suggest ways to automate and implement a particular form of trust assertion in an environment like NeOn Toolkit.

The future work can be done on many levels. First, one may actually operationally described additional contexts – here of a particular interest is the argumentation and full-scale access control. Second, one may look more closely on reasoning and propagation issues, tools and techniques that would be able to consume

2006–2008 © Copyright lies with the respective authors and their institutions.

existing trust assertions and to automatically produce new implications. Third, one may obviously move more toward a functioning framework for trust representation and acquisition by linking our metamodel to specific actions performed in the NeOn Toolkit. Fourth, one may want to use the proposed metamodel as a basis to expose trust-related data in other existing applications developed in the scope of NeOn project, in particular, ORS [SAd+07].

Bibliography

[AG07] Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semant.*, 5(2):58–71, 2007.

[ARH97] A. Abdul-Rehman and S. Hailes. A distributed trust model. In *Proc. of the New Security Paradigms Workshop*, 1997.

[BFK98] M. Blaze, J. Figenbaum, and A. D. Keromytis. Keynote: Trust management for public-key infrastructure (position paper). In *Proc. of the 6th International Workshop on Security Protocols*, pages 59–63, 1998.

[BFL96] M. Blaze, J. Figenbaum, and J. Lacy. Decentralized trust management. In *Proc. of the IEEE Symposium on Security and Privacy*, 1996.

[BZ97] B. Bhargava and Y. Zhong. A distributed trust model. In *Proc. of the Data Warehouse and Knowledge Management Conf.*, 1997.

[CDH07] E. Chang, T. S. Dillon, and F. Hussain. Trust ontologies in e-service environments. *International Journal of Intelligent Systems*, 22:519–545, 2007.

[CFL$^+$97] Y.-H. Chu, J. Figenbaum, B. LaMacchia, P. Resnik, and M. Strauss. REFEREE: Trust management for Web applications. *Computer Networks and ISDN Systems*, 29:953–964, 1997.

[CNS03] M. Carbone, M. Nielsen, and V. Sassone. A formal model for trust in dynamic networks. In *Proc. of the First International Conference on Software Engineering and Formal Methods*, pages 54–61, 2003.

[CY00] R. Chen and W. Yeager. Poblano: a distributed trust model for peer-to-peer networks. Technical report, Sun Microsystems, 2000.

[DGA$^+$07] M. Dzbor, Laurian Gridinoc, C. Buil Aranda, A. Lopez-Cima, and A. Kubias. The role of access rights in ontology customization. Deliverable D4.4.1, NeOn Project, 2007.

[Din00] R. Dingledine. The Free Haven project: Design and deployment of an anonymous secure data heaven. Technical report, MIT, Master’s Thesis, 2000.

[DKG$^+$07] Martin Dzbor, Alexander Kubias, Laurian Gridinoc, Angel Lopez Cima, and Jose Manuel Gomez. The role of access rights in ontology customization. Deliverable D4.4.1, NeOn Project, 2007.

[DRS$^+$06] Nicholas Drummond, Alan Rector, Robert Stevens, Georgina Moulton, Matthew Horridge, Hai Wang, and Julian Sedenberg. Putting owl in order: Patterns for sequences in owl. In *OWL Experiences and Directions (OWLEd 2006)*, Athens Georgia, 2006.

[Ess97] D.J. Essin. Patterns of trust and policy. In *Proc. of the New Security Paradigms Workshop*, 1997.

2006–2008 © Copyright lies with the respective authors and their institutions.

[FC04] R. Falcone and C. Castelfranchi. Trust dynamics: How trust is influenced by direct experiences and by trust itself. In *AAMASÕ04*. Springer, 2004.

[Gan08] A. Gangemi. Norms and Plans as Unification Criteria for Social Collectives. *Journal of Autonomous Agents and Multi-Agent Systems*, 16(3), 2008.

[GM08] Bernardo Cuenca Grau and Boris Motik. OWL 2 Web Ontology Language: Model-Theoretic Semantics. http://www.w3.org/TR/owl2-semantics/ [2008-05], 2008.

[GPALC07] Jose Manuel Gmez-Prez, Carlos Buil Aranda, Toms Pariente Lobo, and Germn Herrero Crcel. Software architecture for the neon pharmaceutical case studies. Deliverable D8.2.1, NeOn Project, 2007.

[GPH03] J. Golbeck, B. Parsia, and J. Hendler. Trust networks on the semantic web. In *Proc. of the 7th International Workshop on Cooperative Intelligent Agents*, 2003.

[GS03] T. Grandison and M. Sloman. Specifying and analysis trust for internet applications. In *Proc. of the 7th International Workshop on Cooperative Intelligent Agents*, 2003.

[HF06] Jingwei Huang and Mark S. Fox. An Ontology of Trust. Formal Semantics and Transitivity. In *ICECÕ06, August 14-16, 2006, Fredericton, Canada*. ACM, 2006.

[HRW+06] Peter Haase, Sebastian Rudolph, Yimin Wang, Sartje Brockmans, Raul Palma, Jerome Euzenat, and Mathieu d'Aquin. The networked ontology model. Deliverable D1.1.1, NeOn Project, 2006.

[Jos96] A. Josang. The right type of trust for computer networks. In *Proc. of the ACM New Security Paradigms Workshop*, 1996.

[LSNM06] H. Lewen, K. Supekar, N.F. Noy, and M.A. Musen. Topic-Specific Trust and Open Rating Systems: An Approach for Ontology Evaluation. In *Proc. of the 4th International Workshop on Evaluation of Ontologies for theWeb (EON2006) at the 15th International WorldWideWeb Conference (WWW 2006)*, 2006.

[Mar94] S. Marsh. Formalizing trust as a computational concept. Technical report, Dept. of Mathematics and Computer Science, University of Sterling, PhD Thesis, 1994.

[ODP] Content Patterns Portal. http://www.ontologydesignpatterns.org.

[PGD+08] V. Presutti, A. Gangemi, S. David, G. Aguado de Cea, M.C. Suarez-Figueroa, E. Montiel-Ponsoda, and M. Poveda. Library of design patterns for collaborative development of networked ontologies. Deliverable D2.5.1, NeOn project, 2008. http://www.neon-project.org/.

[PHGP+06] Raúl Palma, Jens Hartmann, Asunción Gómez-Pérez, York Sure, Peter Haase, Mari del Carmen Suárez-Figueroa, and Rudi Studer. Towards an ontology metadata standard. In *Poster at 3rd European Semantic Web Conference, ESWC 2006*, 2006.

[PHWd08] Raul Palma, Peter Haase, Yimin Wang, and Mathieu d'Aquin. Propagation models and strategies. Deliverable D1.3.1, NeOn Project, 2008.

[QH08] Guilin Qi and Peter Haase. D3.1.3 Improved NeOn formalism for context representation. Technical report, The NeOn Project, deliverable report, 2008.

[QP08] Guilin Qi and Peter. Improved neon formalism for context representation. Deliverable D3.1.3, NeOn Project, 2008.

[REG] Region CP. http://www.ontologydesignpatterns.org/cp/owl/region.owl.

[S. 04] S. Bechhofer et al. Owl web ontology language reference. http://www.w3.org/TR/owl-ref/, 2004.

[SAd+07] Marta Sabou, Sofia Angeletou, Mathieu d’Aquin, Jesús Barrasa, Klaas Dellschaft, Aldo Gangemi, Jos Lehmann, Holger Lewen, Diana Maynard, Dunja Mladenic, Malvina Nissim, Wim Peters, Valentina Presutti, and Boris Villazń. D2.2.1 Methods for Selection and Integration of Reusable Components from Formal or Informal User Specifications. Technical report, The NeOn Project, 2007.

[Sch08] Simon Schenk. On the semantics of trust and caching in the semantic web. In *ISWC2008: 7th International Semantic Web Conference*, 2008.

[SIT] Situation CP. http://www.ontologydesignpatterns.org/cp/owl/situation.owl.

[Vil05] L. Viljanen. Towards an Ontology of Trust: Trust, Privacy and Security in Digital Business. In *Lecture Notes in Computer Science*, pages 175–184, 2005.

[ZJ07] M. Zhu and Z. Jin. Trust Analysis of Web Services Based on a Trust Ontology. In *Proc. of the Conf. on Knowledge Science, Engineering and Management*, 2007.

2006–2008 © Copyright lies with the respective authors and their institutions.