Karlsruhe Institute of Technology

Social Sense-Making with Linked Open Data

Diploma Thesis of

Andreas Kreidler

At the faculty of Economics and Business Engineering
Institute of Applied Informatics
and Formal Description Methods (AIFB)

Reviewer: Prof. Dr. Rudi Studer
Advisor: Dr. Max Völkel

Submission date: 15.08.2011

”Ich versichere hiermit wahrheitsgemäß, die Arbeit selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.”

’Hereby I declare that I have written this thesis by my own. Furthermore, I confirm that no other sources have been used than those specified in the thesis itself’

Karlsruhe, 15.08.2011

Andreas Kreidler

Abstract

Decision making is an omnipresent topic. Apart from business and law where the decision making process is part of day-to-day business and computer supported argumentation has already been established, personal decision making is an increasingly interesting topic. This is the case especially when thinking about the ubiquity of information technology in our modern times. Everywhere on the internet people are deliberating about current topics and want to make up their own opinion independent of print media or television even if they are no experts. This is where social networks join in the game. Decision making becomes a crowd sourcing activity. For aspects of decisions people aren't versed in or even don't have a dedicated opinion about, they rely on friends or friends of friends et cetera connected by implicit social trust frameworks. There are three tasks that are crucial in this context. A huge amount of facts has to be (1) collected (2) combined meaningfully and (3) mapped to one's subjective point of view (personalized). To offer a whole approach assisting people in this activity of collaborative decision making kNet has been developed. From a more technical standpoint this can be reduced to one main purpose namely bringing semantics into discussion threads in a way that a computer can assist people in building their personal opinion. kNet contributions are (1) a well elaborated data model which enables people to bring forward their arguments in a manner that they are understandable by machines i.e. reasoning can be applied, (2) a rating model used to build a concise and reliable trust network between subjects and to rate facts and relations to evaluate arguments in a subjective manner and (3) an application architecture with focus on layering which is different from current existing approaches e.g. cohere. Evaluations show that kNet can help people to make less intuitive and more substantiated decisions. This is possible because it combines trust networks directly with arguments and therefore makes arguments more traceable.

Contents

1. Introduction

Problem

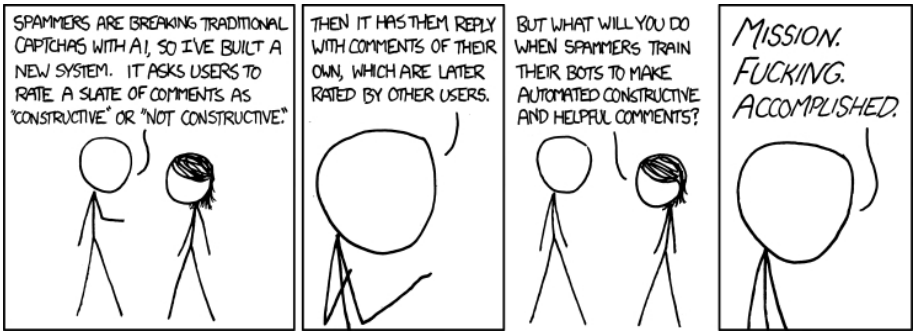

Figure 1.1: http://xkcd.com/810/

Every human action is a result of a decision process. People decide all the time: they decide what to buy, what to eat, which good to produce, whom to vote for and so on. The main determinant of their decisions is their experience. They remember how they acted in a similar situation and what was the positive or negative result of their action. Good decisions take many facts into account. The difficulty of a decision is to retrieve all the relevant facts, combine them meaningfully and choose the action which probably leads to the desired result. Therefore, it is quite easy for people to decide simple questions. They just remember the few facts needed or look them up and combine them. However, when it comes to difficult questions with a very huge amount of facts to be combined this is no longer an option. Even if all the facts can be retrieved, it is often hardly possible to combine them and make a well-founded decision. So people tend to simplify their decision by discarding facts which at the end leads to the aforesaid simple question which can be decided easily. With a non-systematic process in ignoring facts, the resulting decision often does not lead to the expected results. Additionally, decisions have to be taken on topics in which individual opinions and assumptions play an important role. There are few systematic approaches to deal with a large number of facts and opinions in a traceable and clear way.

To sum up, there are three problems that have to be overcome when making a decision: a huge amount of facts has to be (1) collected (2) combined meaningfully and (3) mapped to the subjective point of view (personalized).

Idea

As illustrated in Section 1.1 the first step of taking a decision is to collect the facts on which the subsequent decision will be based. The main idea is to solve this with collaborative modelling. People express their ideas in a discussion-thread-like environment which is semantically enriched on the basis of a common ontology. People can post simple facts as well as relations between these facts. A first result is a description with a common vocabulary of a specific decision problem or topic area in a machine readable form. Due to the machine readable semantics of some parts of the vocabulary, reasoning engines can augment the discourse by supplying logical deductions and hence help to find contradictions. User ratings are introduced to overcome the problem of subjectiveness. Every element in the elementary ontology can be rated. Users can rate other users, facts and relations between facts. The basic concepts can be seen in figure 1.2. With the help of ratings it is possible to map certain questions to people's personal point of view only with this collected data and without their further contribution. To enable other researchers to reuse the collected data or build their own applications on top of the developed software, the focus is set on openness. This includes the possibility to reuse parts of the UI, to execute high level queries on the logic layer and to access the collected raw data as linked open data [Bizer, 2009] sets. Therefore, the application is divided into well separated layers which are independently accessible via web interfaces (N-tier architecture).

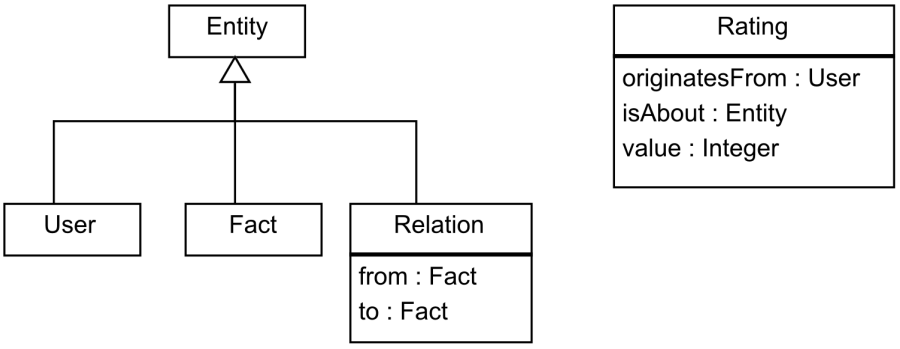

Figure 1.2: Basic concepts

2. Background

This chapter gives a short overview of the topic. At first, Sec. 2.1 introduces several terms concerning the topic argumentation along with their historical foundations and gives a vision about the future of argumentation on the web which resembles the vision of the semantic web. In figure 2.1 it is depicted how this concepts are linked up. Finally, Sec. 2.2 takes a look at different argumentation software tools and their corresponding models.

2.1 Theoretical Foundations

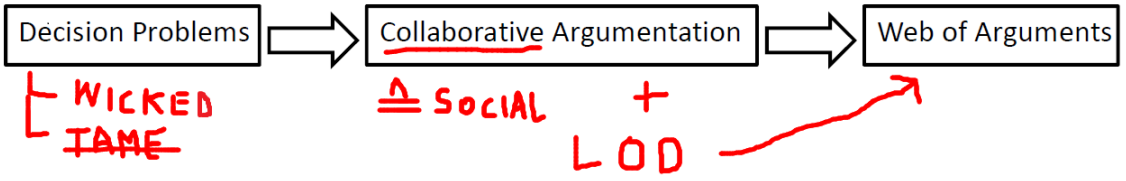

Figure 2.1: Long Way to the Web of Arguments

Decision Problems

As already described in the introduction, decision problems can be classified according to their complexity. An early work about the classification of problems is Rittel and Webber's "Dilemmas in a General Theory of Planning" [Rittel and Webber, 1973]. They divide problems into two classes of complexity, namely *wicked problems* and *tame problems*. This classification should not be confused with classification of computational problems in complexity theory dealing with the complexity of execution time and space of algorithms see [Hartmanis and Stearns, 1965]. In fact, almost all problems coming from mathematics or engineering are according to Rittel and Webber *tame problems* i.e. easy problems. They "...are definable and separable and may have solutions that are findable..". *Wicked problems* on the the other hand are mostly societal problems that "have neither of these clarifying traits". A *Wicked problem* for example is a question like "What causes global warming?" or "Should a public health system be established?". These are mostly political and social problems that neither can be formulated clearly nor have a sole solution. Rittel and Webber described this characteristic with the wordplay "...Social problems are never solved. At best they are only resolved - over and over again...". They also list ten properties

of *wicked problems*. Two of them that are almost sufficient to describe a *wicked problem* in this context are:

- "1. There is no definitive formulation of a wicked problem"
- "3. Solutions to wicked problems are not true-or-false, but good-or-bad"

Another characterization of a problem that's hard to solve is the *ill-structured problem* formulated by Simon [Simon, 1973]. It resembles Rittel and Webber's *wicked problem* but is written from a more technical point of view. It rather deals with the issue if a machine is capable to find a solution for a specific problem. Simon differentiates between *well-structured problems (WSPs)* and *ill-structured problems (ISPs)*. In contrast to Rittel and Webber, Simon defines six criteria for a *well-structured problem* and defines an *ill-structured problem* as a problem that is not a *well-structured problem*. Examples for the criteria are:

- "1. There is a definite criterion for testing any proposed solution..."
- "2. There is at least one problem space in which can be represented the initial problem state, the goal state, and all other states..."

To sum up, interesting problems in the context of decision making are problems that are very hard to solve and whose complexity goes back to (1) difficulty of its formulation, (2) the non-existence of a sole solution and (3) the fact that political and social considerations play an important role.

This definition can be applied to almost every problem in (1) **politics and macroeconomics**.

But also problems in (2) **science** and (3) **corporate decision making** can fall into that category.

With the spreading of the internet and thousands of online shops, rating websites and the huge amount of information about products on the web (4) purchasing a specific product can be considered as a wicked problem.

Problem solving

In [Roberts, 2000] three classes of strategies are outlined that could be used when dealing with *wicked problems*.

Authoritative Strategies: Reduce the complexity of a problem by reducing the number of people involved in the decision. A classical example for this kind of strategy is the supreme court which is the last resort when it comes to displeasing decisions in western democracies.

Competitive Strategies: Opponents compete for the best solution of a problem. A typical example for this kind of strategy is the free market economy where a lot of manufacturers compete for the product that is most suitable for the customer needs.

Collaborative Strategies: People cooperate to solve a problem. Divison of work is an example for this type of strategy. Due to the complexity of the modern working environment nobody can be an expert on every topic. That's why people concentrate on specific subjects and work together to solve problems.

Every class of strategies has its specific advantages and disadvantages. Whereas *authoritative strategies* fully rely on the judgement of a few experts which could be wrong, *competitive strategies* are wasting resources because opponents don't share information collected in the problem solving process. Finally, the shortcome of *collaborative strategies* are their

high transaction costs i.e. cost for coordinating the people that are involved in problem solving. According to the assumption that modern information technology in combination with the internet, especially the upcoming social networks, faciliate the coordination of many people i.e. decreases transaction costs, *collaborative strategies* would be the most promising ones.

Sense-Making

The term *sense-making* has been described by Weick see [Weick, 1995]. In his work from 1995 he characterizes *sense-making* as "a process that is" (1) "Grounded in identity construction", (2) "Retrospective", (3) "Enactive of sensible environments", (4) "Social", (5) "Ongoing", (6) "Focused on and by extracted cues", (7) "Driven by plausibility rather than accuracy". Besides the first three properties which are rather philosophical, the last four properties describe pretty concise what is important in the context of *sense-making*. First, *sense-making* is a social activity and it is ongoing i.e. has no specific start or ending. Furthermore, people gradually develop a sense of what is going on through a continuous process of assessing and extracting cues. Finally, focus is set on plausibility because the primary goal is the development of ideas and solutions whereas accuracy is playing a secondary role.

Argumentation

There are several historic works about argumentation that can be taken into account. Three important approaches are in chronological order: Wigmore's chart method [Wigmore, 1913] and Toulmin's and Walton's argumentation schemes [Toulmin, 1958] [Walton, 1996][Walton, 2010].

Whereas Wigmore sets its focus on logical deduction, Toulmin and Walton analyze the elements of argumentation. Both aspects should be considered when dealing with argumentation theory.

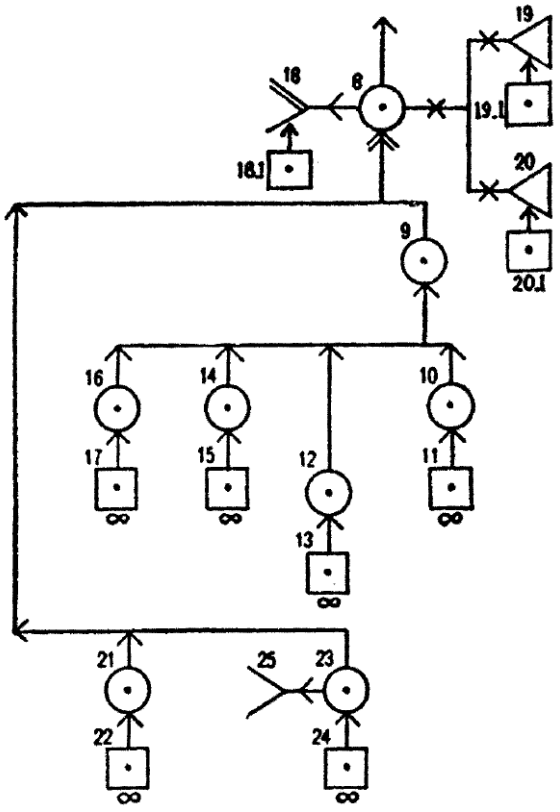

Figure 2.2: Sample Wigmore Chart taken from [Goodwin and Fisher, 2000]

Wigmore's elements are arrows that draw conclusions and nodes (squares, open triangles, closed triangles and circles) which represent facts. They're used to summarize information gathered so far for a specific law case. The shape of a fact is dependent on its usage (argument or additional information), type (impact or support), author (prosecutor or defendant) and strength. A square is an argument directly stated by a witness. A circle is an inferred argument. Whereas open triangles influence the impact of arguments, closed triangles stand for support of arguments. Nodes with a double line identify facts stated by the defendant. See figure 2.2 for an example of a *Wigmore diagram*. For a detailed look at the interpretation of Wigmore diagrams see [Rowe and Reed, 2006].

Toulmin concentrates on the structure of argumentation statements. As depicted in figure 2.3 it's a kind of annotated logical deduction in natural language. Harry was born in Bermuda. Therefore, he is a British subject. To justify this conclusion it's annotated with the corresponding deduction rule and its foundation. Although this conclusion may hold for almost any case there can be exceptions. These exceptions need to be formulated. Toulmin calls the left side of a deduction 'data', the right side 'claim', the annotation of the implication 'warrant', its foundation 'backing' and the exception 'rebuttal'.

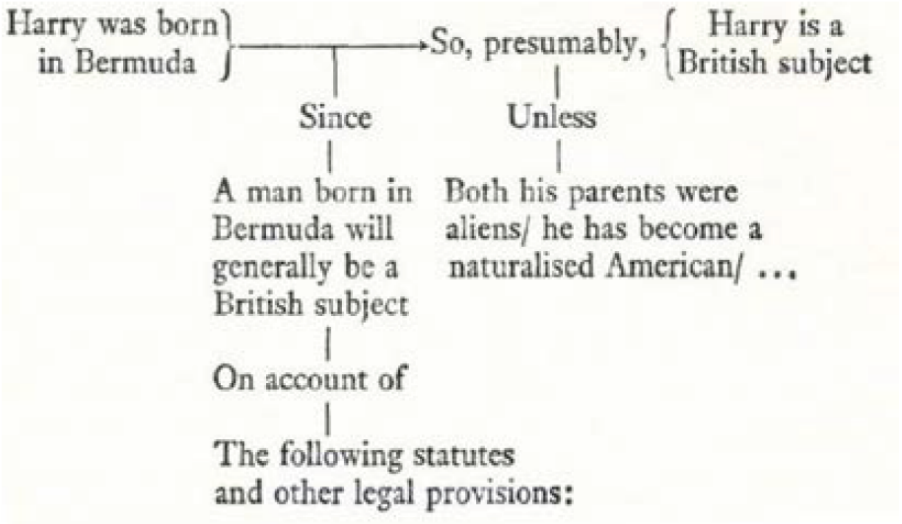

Figure 2.3: Example of Toulmin's argumentation scheme taken from [Toulmin, 1958]

As Toulmin concentrates on the structure of a single argument, Walton analyses the structure of the argumentation process. He classifies argumentation dialogues on the basis of their goal and elobarates at first six [Walton, 1996] and later adds a seventh [Walton, 2010] generic dialogue type. Figure 2.4 shows a list of these seven dialogue types.

TYPE OF DIALOGUE	INITIAL SITUATION	PARTICIPANT'S GOAL	GOAL OF DIALOGUE
Persuasion	Conflict of Opinions	Persuade Other Party	Resolve or Clarify Issue
Inquiry	Need to Have Proof	Find and Verify Evidence	Prove (Disprove) Hypothesis
Discovery	Need to Find an Explanation of Facts	Find and Defend a Suitable Hypothesis	Choose Best Hypothesis for Testing
Negotiation	Conflict of Interests	Get What You Most Want	Reasonable Settlement Both Can Live With
Information-Seeking	Need Information	Acquire or Give Information	Exchange Information
Deliberation	Dilemma or Practical Choice	Co-ordinate Goals and Actions	Decide Best Available Course of Action
Eristic	Personal Conflict	Verbally Hit Out at Opponent	Reveal Deeper Basis of Conflict

Figure 2.4: Walton's dialogue types taken from [Walton, 2010]

Linked Data

Linked Data is a concept primarily developed by Tim Berners-Lee. In [Berners-Lee, 2006] he states four basic rules Linked Data should satisfy. The quintessence of the rules is that every element should by addressable by a HTTP URI [1] and should "include links to other URIs". In 2010 he added a 5-star scheme to evaluate Linked Data:

- 1 star "Available on the web (whatever format), but with an open licence"
- 2 stars "Available as machine-readable structured data (e.g. excel instead of image scan of a table)"
- 3 stars "as (2) plus non-proprietary format (e.g. CSV instead of excel)"
- 4 stars "All the above plus, Use open standards from W3C (RDF and SPARQL) to identify things, so that people can point at your stuff"
- 5 stars "All the above, plus: Link your data to other people's data to provide context"

What is commonly meant when talking about *linked open data* has a rating of four stars and above. Expressed in simple terms it's just a bunch of linked RDF[2] data.

Web of Arguments

Whereas the classical world wide web mostly links data, the aim of the semantic web is to link information on the basis of its meaning. Therefore, the documents are no longer represented by HTML[3] which mainly describes the visualization of data but by RDF[4]. RDF describes the meaning of the information contained in the data. Figure 2.5 shows the layered architecture of the envisioned semantic web.

[1] [Berners-Lee, 1994]
[2] [Lassila et al., 1998]
[3] Hyper Text Markup Language
[4] [Lassila et al., 1998]

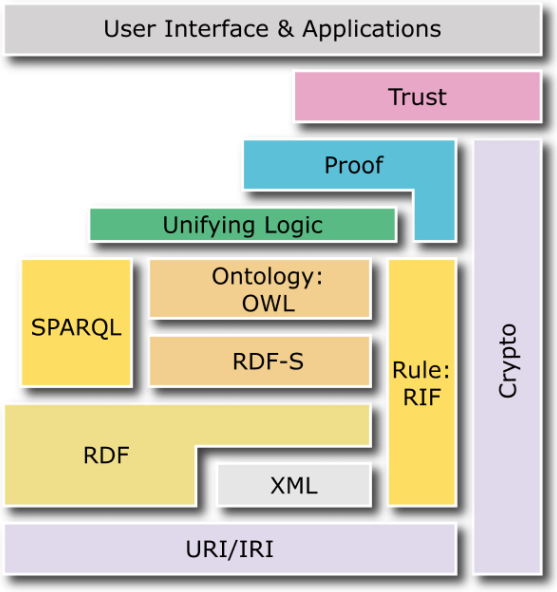

Figure 2.5: Semantic Web Stack taken from [Bratt, 2007]

To build the bridge from semantic web to argumentation, arguments are expressed in RDF and made available on the web as open linked data. In dependence on the semantic web this could be called "argument web" or "web of arguments". In [Schneider et al., 2010] which deals with overcoming the gap between web 2.0 and the semantic web in argumentation systems term "argumentation 3.0" is brought up according to "web 3.0" a term which is also used when talking about the semantic web.

2.2 Existing Software Tools

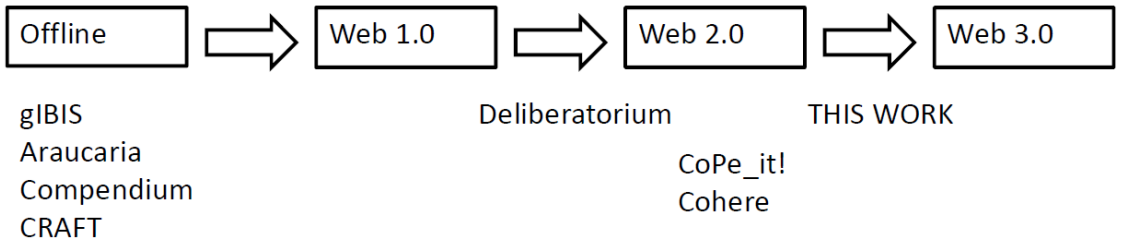

Figure 2.6: Progress of Software Tools

Besides the theoretical work there are a few software tools to mention which already have been implemented to enable computer supported argumentation. Figure 2.6 classifies existing software tools according to the used technologies.

A very often cited tool when talking about argumentation is gIBIS [Conklin and Begeman, 1988] which is based on a work of Kunz and Rittel [Kunz and Rittel, 1970].

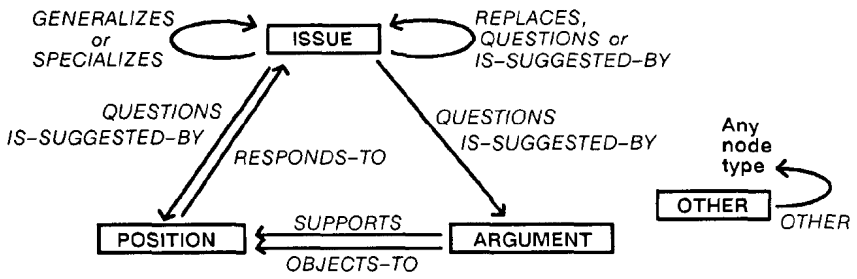

Figure 2.7: elements of gIBIS

gIBIS has a very strict structure with four different elements and a few fixed relations. These elements are illustrated in figure 2.7.

A recent ontology for argumentation which is also the basis for Cohere [Buckingham Shum, 2008] is Scholonto from Knowledge Media Institute. Scholonto's key concepts are claims and links, whereas the main focus is on the links as you can see by comparing the number of properties. The core concepts of Scholonto are shown in figure 2.8.

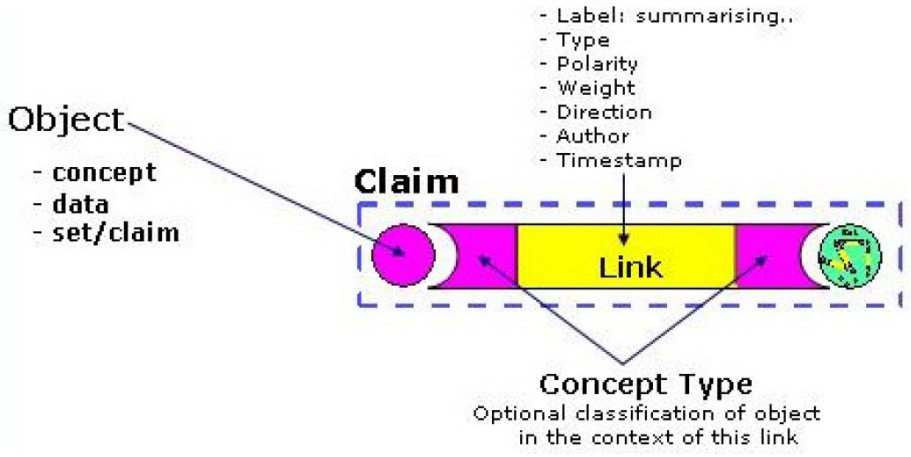

Figure 2.8: Scholonto

An example for an hierarchy of link types with different semantics is given in [Buckingham Shum, 2008] see figure 2.9.

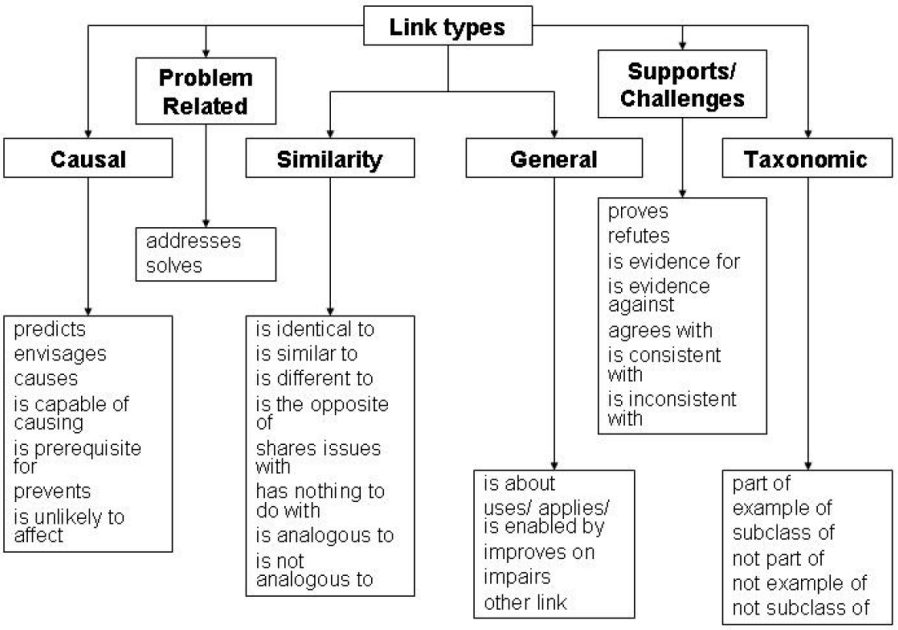

Figure 2.9: Scholonto

As an example for a software tool which takes argumentation schemes into account, Araucaria [Reed and Rowe, 2004] can be mentioned. It also includes a first approach to formulate a reusable ontolgy in XML. As argumentation schemes the classical schemes developed by Walton are used. They consist of a list of premises, a conclusion and several critical questions. Argumentation schemes are more a theoretical construct whose strength is accurate logical deduction rather than ease of usage. However, for personal decision making accurate logical deductions are playing a secondary role (see description of the term *Sense-Making*).

There also have been efforts to develop an argument interchange format (AIF). Due to its complex structure it's not suitable as a basis ontology. For more information about this topic see [Chesnevar and McGinnis, 2006].

The proposed argumentation schemes for the semantic web e.g. [Lange et al., 2008][Passant et al., 2010] are even more complex.

A lot of tools have been built regarding the topic "collaborative argumentation" support. E.g. Compendium[5][Buckingham Shum, 2008], Deliberatorium[6][Klein and Iandoli, 2008], cope it[7][Tzagarakis et al., 2009], CRAFT[Hupfer et al., 2009], Cohere[8] to mention a few. A high potential application out of this enumeration and therefore the one to compare with is Cohere. It's a PHP web application built at the Knowledge Media Institute[9]. For a closer look at Cohere see [De Liddo and Buckingham Shum, 2010].

[5]http://compendium.open.ac.uk/
[6]http://cci.mit.edu/research/deliberatorium.html
[7]http://copeit.cti.gr
[8]http://cohere.open.ac.uk/
[9]http://kmi.open.ac.uk/

3. Use Cases

A promising approach to select requirements for the argumentation model is to elaborate typical use cases. This will be running examples not only used for inferring requirements but also to illustrate model, implementation and at least its evaluation. As already outlined in Sec 2.1., complex decision problems especially arise in the areas (1) purchase advice, (2) science, (3) politics and macroeconomics and (4) corporate decision making. Therefore, from each of these areas two typical use cases were chosen. A use case is described by its area of application, a short but concise description of the subject that wants to make a decision, the decision problem itself and a non-exhaustive enumeration of questions that may appear when making such a decision.

Purchase Advice

UC1: [1] An ecology-minded girl wants to buy a yoghurt with natural ingredients.

UC1.1. Where does this yoghurt come from (emission)?

UC1.2. Which ingredients are natural according to her opinion?

UC1.3. What about the company selling this yoghurt (evil multinational company which doesn't care about fair trading)?

UC2: A patriotic man who likes playing video games wants to buy a new tv.

UC2.1. Which brands to buy?

- Which brands fulfil a minimum of quality requirements?
- Which brands are bestsellers ?

UC2.2. What about the manufacturer of the brand?

- What's the name of the manufacturer?
- Is the manufacturer a domestic company?
- Is this tv produced in a foreign country?
- Does this company reduce or maintain domestic jobs?

[1] Use cases are enumerated from UC1..UCn

- Does this company have at least a subsidiary his home country?
- Does this company pay taxes?
- Does this company suppress domestic companies?

UC2.3. Which features are required for gaming?

- Which response time is needed?
- Which image repeat rate is needed?
- Which aspect ratio is needed?

UC2.4. What about the display technology itself?

- Which display technologies exist?
- Which display techonology is the best for gaming?
- Which display technology is sustainable?

UC2.5. What about connectivity?

- Which connectors exist?
- Which connectors are needed for gaming?
- Which connectors are sustainable?

Science

UC3: A student wants to decide whether a diet is working.

UC3.1. What are the ingredients of the diet?

UC3.2. Does it work for all people or is it dependent on specific genes?

UC3.3. Is it just a calory thing?

UC3.4. Does it work in the longterm?

UC4: A canny housewife wants to know whether a electric car or a conventional car is the better choice for her family.

UC4.1. What about the price difference?

UC4.2. How will the gas price develop compared with the price of electricity?

UC4.3. What about carbon dioxide emission?

UC4.4. What about range and durability?

Politics and macroeconomics

UC5: The German government wants to decide whether to bail out Greece or not.

UC5.1. What economic theories are in favour of and which are against?

UC5.2. What about European partners?

UC5.3. What about elections in the future?

UC5.4. What about inflation?

UC6: A philosopher wants to know if a minimum wage is a good idea

UC6.1. What is its possible height?

UC6.2. What does it mean for unemployment?

UC6.3. How will it change society?

UC6.4. Will it boost the economy?

Corporate decision making

UC7: A product manager has to decide whether or not to launch a product.

UC7.1. Do people need this product?

UC7.2. Do they want it?

UC7.3. What about ethical or ecological aspects?

UC7.4. Is there a competitive product?

UC7.5. What are its shortcomings?

UC8: An energy provider wants to decide whether to abondon nuclear power voluntarily.

UC8.1. Is it definitely an image improvement?

UC8.2. Are customers willing to pay more?

UC8.3. What does it cost?

UC8.4. What are the alternatives to nuclear power?

4. Requirements

This chapter collects requirements for the different components of the model. As a starting point the use cases from the previous chapter are used. At first, the requirements for a layered design are elaborated. The remaining structure of the chapter follows these requirements whose core is a strict separation of data model, rating model and reasoning component. Sec. 4.2. describes requirements for a data model. In Sec. 4.3 requirements for a rating model are derived. Sec 4.4 elaborates requirements for a reasoning component. Finally, Sec. 4.5 takes a look at requirements for a user interface.

4.1 Layering

As it is intended that other researchers reuse this work, every component should be independent and reusable as itself (R1). In this way other researchers for example may build a completely different argumentation software build upon the data model or may reuse everything except for the reasoning engine which they replace with a different one. For a clean layer design, every layer should differentiate between API[1] and its corresponding data. Also the corresponding data should be reusable i.e. the data source used by the API should be accessible (R2). To make reusing easy, web standards like JSON[2] or RDF[3] should be used as an interchange format between layers (R3).

4.2 Data Model

When looking at the questions in chapter "Use Cases" it's remarkable that even supposedly simple questions, e.g. which yoghurt to buy, are very complex decision problems when paying attention to personal attitude. One's personal experience or knowledge is in almost all cases not sufficient to make a substantiated decision. People need the help of friends or experts they trust in to gain a better insight. That's why trust is probably the most important concept, especially when people are not able to verify the corresponding arguments because they are not acquainted with the corresponding matter. There are two aspects that appear in connection with trust. Trust between users and the subjective trust a user has in an argument.

[1] Application programming interface
[2] json2006
[3] rdf98

The latter can be the result of (1) a user's direct rating of an argument, (2) the trust in another user's rating of an argument, (3) the trust in the user that has brought forward the argument or (4) the conclusion of reasoning starting at (1), (2) or (3).

Derived requirements for the data model are:

- Data should at least differentiate between users, arguments and relations (R4)
- Data model should be able to express ratings between users (R5)
- Data model should be able to express rating of arguments by users (R6)
- Every argument should have an author because trust in arguments is directly related to trust in their authors (R7)
- There should be the possibility to attach semantics to relations to enable reasoning (R8)

Additional requirements for the Data Model can be derived from the requirements that make up Linked Data. The first requirement is derived from the four rules Tim Berners-Lee stated in 2006 [Berners-Lee, 2006]. The star scheme he added in 2010 is the basis for the second required dealing with linked open data. For a closer look at the rules and the star scheme see Sec. 2.1.

The additional requirements are:

- Every Element should have an HTTP URI (R9)
- Data should be stored in a form that can be transformed easily into RDF and SPARQL e.g. produce at least 4-star data (R10)

4.3 Rating Model

A simple model of trust rating that is widely used describes trust as a simple directed graph with weighted edges. The weight of an edge expresses the extent of trust between the entities represented by the nodes.

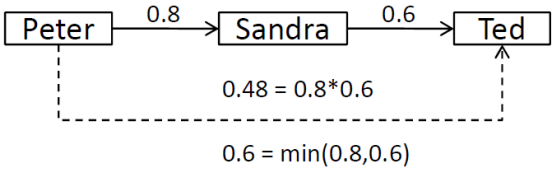

Figure 4.1: Simple Trust Model

See figure 4.1 for an example (trust between 0 and 1). Peter has deep trust in Sandra with a high value of 0.8 and Sandra trusts in Ted with a trust rate of 0.6. Because the graph is directed nothing is known about Sandra's trust in Peter or Ted's trust in Sandra.

Although this is a very simple model there are already a lot of design decisions to be taken when implementing it:

- How to measure the extent of trust? (Rating Representation)
- How to calculate trust? (Rating Computation)

When looking at a more complex subgraph there are additional questions to be considered:

- How to aggregate trust along edges of a path? (Trust Propagation)
- How to aggregate trust along different paths? (Aggregation of Trust)
- What about contradictory paths? What about cycles? (Calculation Bias)

Propagation of Trust

A very simple approach to propagate trust along one path i.e. a transitive edge between source and destination node e.g. Peter's trust in Ted in figure 4.1 is just taking the minimum of all edges. Another approach is to multiply weights along a path. It can easily be seen that the second approach is the more promising one because it additionally takes the path length into account. This is intuitive when looking at the game 'Chinese Whispers' [4]. (*Direct trust should have a higher rating value than indirect trust (R11)*)

A problem of using simple multiplication is the propagation of distrust. Taking a look at figure 4.2 it is easy to see that Peter's distrust in Ted should be higher than his distrust in Sarah because his trust in Sandra's rating is higher than in Tom's. But this fact is not reflected by the muliplied edges. Also the fact that Peter distrusts Sandra who trusts Ted does not automatically mean that Peter distrusts Ted which would be the result of multiplication. (*The relation of distrust should be preserved when propagating (R12) see figure 4.2*)

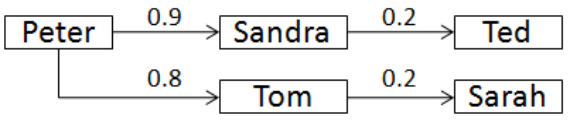

Figure 4.2: Propagation of Distrust

It is obvious that trust expressed by untrusted users will be disregarded (*From distrust nothing can be inferred (R13) see figure 4.3*)

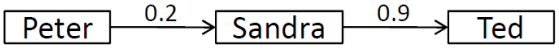

Figure 4.3: From distrust nothing can be inferred

Another unobvious property of these two approaches, especially for performance purposes, is that incremental updates of rating values are possible i.e. when inserting a new edge its trust value can easily be calculated using just its direct predecessors. This results from the self-maintainability of the used aggregate functions. For a closer look at properties and classification of aggregate functions see [Han, 2005]. (*Functions for trust aggregation should be self-maintainable (R14)*)

Aggregation of Trust

As far as now, a simple scalar mapped to interval [0,1] is sufficient to measure the trust between two entities. As soon as it comes to aggregation of multiple paths between the same source and destination this is no longer sufficient because it lacks expressibility.

An example is shown in figure 4.4.

[4] It's a children's game where a term is passed from person to person in a row whispering. For a brief example of this phenomenon see introduction of [Blackmore, 2000]

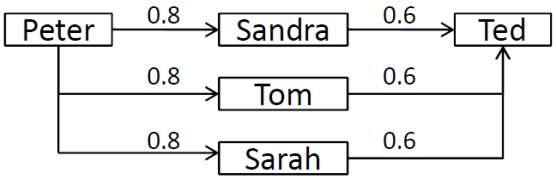

Figure 4.4: Multiple Paths

It's intuitive that when the trust graph looks like in figure 4.4 Ted is more trustworthy for Peter than for a graph like it is shown in figure 4.1. But when using a single scalar[5] for the weighting of edges this difference cannot be expressed. (*The aggregation of trust values from multiple paths should be considered higher than one path with the same value (R15)*)

Also the difference between Sandra fully distrusts Peter, i.e. there exists a path whether direct or transitive between them in the trust graph whose weighting is zero, and Sandra is uncertain about trusting Peter because there exists no connection between them, cannot be expressed. (*Trust metric should at least differentiate between trust and uncertainty (R16)*)

Jøsang proposes a trust metric which is more expressive than just a simple scalar [Jøsang, 1998]. It divides trust into three dimensions belief (b), disbelief (d) and uncertainty (u) see figure 4.5. A rating is expressed by the triple (b,d,u) where b,d,u in [0,1] and b + d + u = 1.

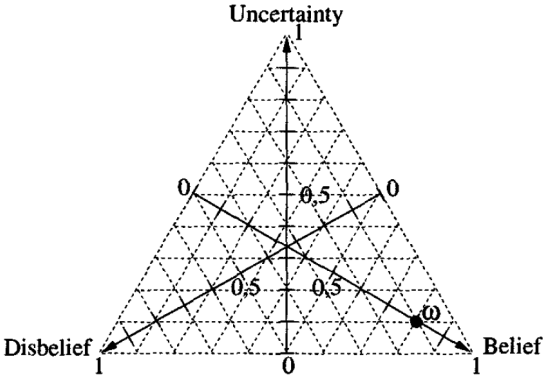

Figure 4.5: Opinion Triangle taken from [Jøsang, 1998]

As can be seen easily, this trust metric is superior because it has one more degree of freedom. Jøsang's work [Jøsang, 1998] along with more recent works based on it like e.g. [Hang et al., 2009] also show that when increasing the number of dimensions used for a trust metric special attention should be paid on formulation of operators for propagation of trust. Hang's CertProp model [Hang et al., 2009] formulates three different operators. Along with concatenation and aggregation which are well-known and already discussed operators he proposes a third operator named selection. It's used for the selection of one path out of a set of overlapping paths (see figure 4.6).

[5] As long as we want to use a fixed intervall. One might consider to overcome this shortcoming by summing up the ratings. However, this is not an option because it leads to almost uncomparable rating values.

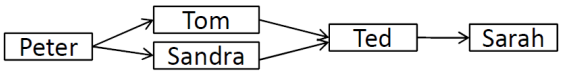

Figure 4.6: Overlapping paths in the trust graph

(*Overlapping paths should not bias trust rating (R17)*)

Trust Calculation Algorithm

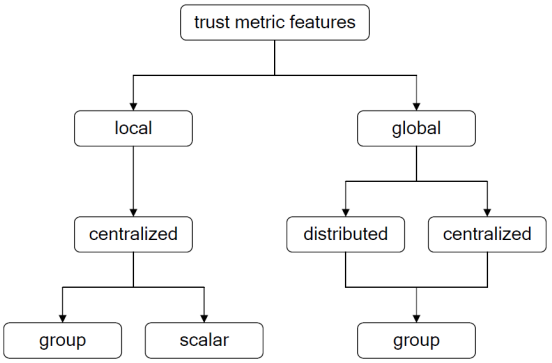

Figure 4.7: Trust metric classification [Ziegler and Lausen, 2004][Ziegler and Lausen, 2005]

Another view on trust rating is the differentiation between global and local trust (see figure 4.7). While global trust algorithms calculate trust with the means of the whole trust graph, a local trust algorithm uses a subgraph representing only the relationship between two specific nodes. For a detailed overview see [Ziegler and Lausen, 2004] and [Ziegler and Lausen, 2005].

When working with opinions, i.e. potentially controversial opinions, local trust metrics are more approriate [Massa and Avesani, 2005]. (*A local trust algorithm should be used (R18)*)

For a more elaborate introduction to trust rating see [Hang et al., 2009].

4.4 Reasoning

Reasoning is one of the core concepts in argumentation. For logical deduction between arguments links have to be semantically interpreted. For this interpretation, links at least need the possibility to be annotated if they should be semantically interpreted or not. In general, binary relations R like links are classified according to their logical properties. Some important classes are:

reflexity Every element x is related to itself i.e $x\mathrm{R}x \forall x \in X$

symmetry If x is related to y then y is related to x i.e. $x\mathrm{R}y \Rightarrow y\mathrm{R}x, \forall x, y \in X$

transitivity If x is related y and y is related to z then x is related to z i.e. $x\mathrm{R}y \wedge y\mathrm{R}z \Rightarrow x\mathrm{R}z, \forall x, y, z \in X$

As it can be seen easily, the most important property of a link for reasoning is transitivity. (*At least transitive links should be supported(R19)*)

4.5 Summary

R.1. Every component should be independent and reusable as itself

R.2. Data source should be accessible on every layer

R.3. Web standards like JSON or RDF should be used as an interchange format between layers

R.4. Data should at least differentiate between users, arguments and relations

R.5. Data model should be able to express ratings between users

R.6. Data model should be able to express rating of arguments by users

R.7. Every argument should have an author because trust in arguments is directly related to trust in their authors

R.8. There should be a possibility to attach semantics to relations to enable reasoning

R.9. Every Element should have an HTTP URI

R.10. Data should be stored in a form that it can be transformed easily into RDF and SPARQL e.g. produce at least 4-star data

R.11. Direct trust should have a higher rating value than indirect trust

R.12. The relation of distrust should be preserved when propagating

R.13. From distrust nothing can be inferred

R.14. Functions for trust aggregation should be self-maintainable

R.15. The aggregation of trust values from multiple paths should be considered higher than one path with the same value

R.16. Trust metric should at least differentiate between trust and uncertainty

R.17. Overlapping paths should not bias trust rating

R.18. A local trust algorithm should be used

R.19. At least transitive links should be supported

5. Design

This chapter describes the theoretical concepts developed out of the requirements from the previous chapter. First, Sec. 5.1 gives an overview of the system as a whole with its four layers data layer, rating layer, reasoning layer and user interface layer. Sec. 5.2 describes the data layer based on the underlying data model whose basic concepts are USER, FACT, LINK and RATING. The rating layer is introduced in Sec. 5.3. Its specific characteristic is that it uses a triple of scalars to express and calculate ratings. Sec. 5.4 specifies the components of the reasoning layer. Finally, Sec. 5.5 takes a look at the user interface layer. For an easier referencing the concepts described in chapter five as a whole are given the name *kNet*. A term first brought up by Heiko Haller in [Haller, 2010] where he envisions an argumentation software that served as a source of inspiration for the developed concepts.

5.1 Layering

Except for the user interface layer, every layer is split into two parts: a data part exposing the corresponding data and an API[1] part used to manipulate the data. The lowermost layer is the data layer. It stores the elements of the data model and is accessible via linked open data. For a more detailed information about the elements persisted in the data layer see Sec. 5.2. Build upon the data layer is the rating layer. It enriches the basic data fetched from the data layer with direct and inferred rating information according to the rating model described in Sec. 5.3. It is also separately accessible via open linked data. The third layer is responsible for reasoning. In the absence of a layer for filtering, this layer also filters out data with a rating less than a specific threshold[2]. After filtering the data the reasoning layer draws conclusions between entities with the help of semantic links and enriches the filtered data with drawn conclusions. Again this enriched data is available as linked open data. The uppermost layer is the user interface. On the one hand, it visualizes the data from the reasoning layer. On the other hand, it is used to manipulate the data on the different layers. Figure 5.1 shows this layered architecture.

[1] Application programming interface
[2] The whole process is described in Sec. 5.6.

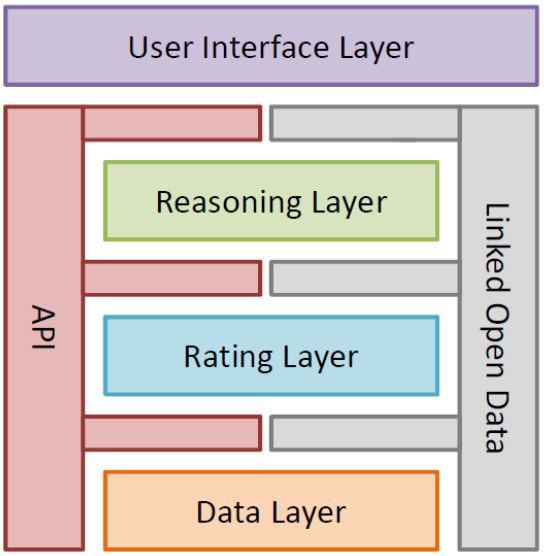

Figure 5.1: Layered Architecture

5.2 Data Model

As already outlined in the introduction, the core idea of *kNet* is that facts lead to other facts. People reach a decision by deciding whether facts are true according to their point of view. To decide whether facts are true or not they need to be evaluated and rated. Due to the huge amount of facts this is a very time consuming and frustrating task. To reduce the number of facts that have to be rated directly, a trust network between users is established. With the means of this trust network the main part of the facts is rated indirectly through relations between users and facts.

Therefore, the core concepts of the data model are USER, LINK, FACT and RATING (see 5.2.

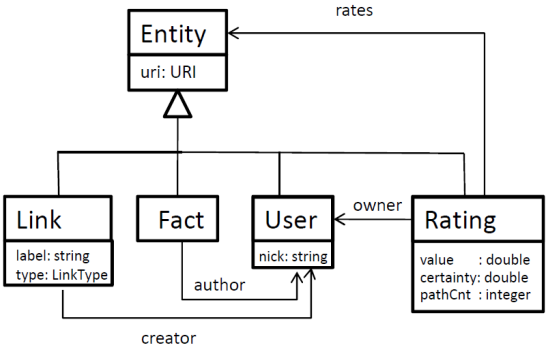

Figure 5.2: Top Level Elements of the Data Model

ENTITY

The basic concept of the model is an ENTITY. ENTITY is the superclass of all other elements. Its only property is an URI. The URI is a uniform resource identifier as decribed in [Berners-Lee, 1994]. Therefore, it's assured that every element of the model is addressable via an URI.

USER

Users are virtual representation of human beeings. They are used as *authors* for FACTS, *creators* of *link*s and *owners* of RATINGS. This representation typically consists of a unique identifier. In most cases an email address that is directly connected to a human. In *kNet* only a unique *nickname* is used to identify a USER. *Nicknames* are a more general concept that is not bound to an email address for identification. Maybe a user does not want to expose its email address and wants to use another unique attribut for identifiaction e.g. a public key.

FACT

FACTS are the basic concept of *kNet*. The represent an direct opinion of a person or a post somewhere on a website. With help of FACTS people gradually gather an insight into the decision problem. They rate them and draw conclusions between them and by this way more and more concretize the problem. A concretized problem is a precodition for a well-grounded decision. A Fact could be a TEXT, a URL, a SMARTURL or a STATMENT which consists of two FACTS connected by a LINK. The different fact types can be seen in figure 5.3.

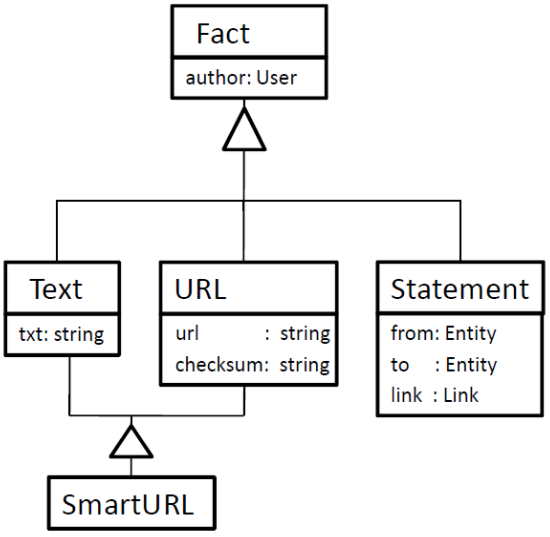

Figure 5.3: Fact Types

TEXT

A TEXT is a piece of plain text expressing the opinion of a USER e.g. “Pasteurized milk is a natural ingredient” see use case (U1) 3.

URL

A URL is a URL pointing to a document on the world wide web as described in [Berners-Lee et al., 1994]. As documents on the web are volatile, i.e. often change, a checksum is stored together with the corresponding URL to do consistency checks of linked documents. An example of an URL is “http://shop2.aol.ca/library/led-vs-lcd-technology-in-tvs-how-to-buy-an-led-lcd-tv/1083” see use case (U2) 3.

SmartUrl

A SMARTURL combines a URL with a TEXT. As web pages often contain very much text and the piece of text users want to reference is in most cases no more than a sentence or two, it would be very inconvenient to link a whole document. This is especially the case when the content is intended to be rated. Therefore, a smart url is introduced which simplifies referencing of single text passages or sentences on web pages. A SMARTURL for the mentioned example would consist of a URL "http://shop2.aol.ca/library/led-vs-lcd-technology-in-tvs-how-to-buy-an-led-lcd-tv/1083" and the corresponding TEXT "LED technology is revolutionary because of the benefits it offers: superior picture quality over traditional LCD TVs" see use case (U2) 3.

Link

A LINK is used to characterize the connection of two ENTITYS. It consist of a *label* giving a textual desciption of the LINK and a corresponding LINKTYPE. There only exist two LINKTYPES: *semantic* and *non semantic* (see figure 5.4).

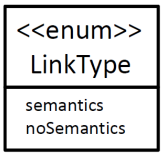

Figure 5.4: Link Types

LinkType

LINKTYPES are needed to control reasoning. LINKS that are instances of LINKTYPE *semantic* can be used by the reasoning engine to draw conclusions (see STATMENT).

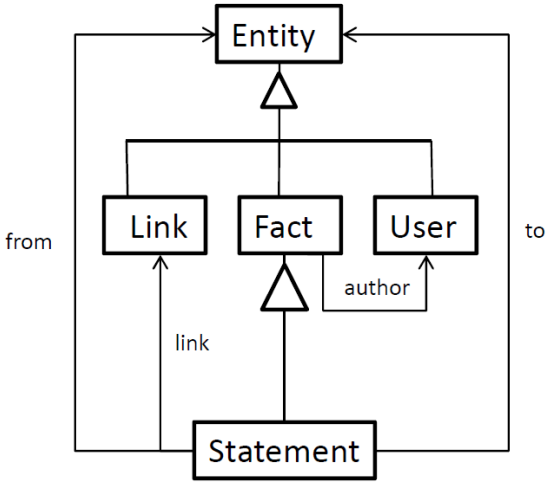

Figure 5.5: A STATEMENT represents a connection between two ENTITYS

Statement

A STATMENT consists of two ENTITYS namely *source* and *destination* and a LINK namely *link*. STATEMENTS are used to make propositions about ENTITYS (see figure 5.5). A example for a STATEMENT is "Playing a video game in standard def" "leadsTo" "game appears blurry".

5.3 Rating Model

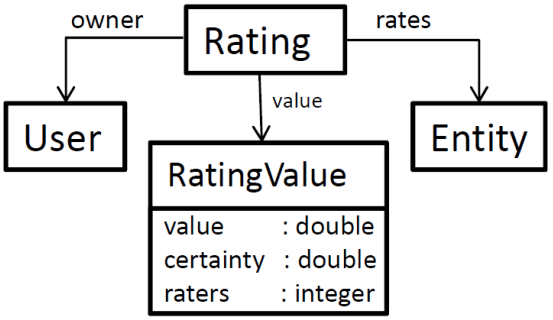

Figure 5.6: A RATING binds a RATINGVALUE to an ENTITY. It is owned by a USER.

RATING $\mathcal{R}$

A RATING stands for the evaluation of an ENTITY whether directly or indirectly made by a certain USER *owner*. It holds a RATINGVALUE which expresses more detailed information about this evaluation. Figure 5.6 shows this relation.

RATINGVALUE $\mathcal{RV}$

A RATINGVALUE is a triple of numerical values $\mathcal{RV} = (\mathcal{V},\mathcal{C},\mathcal{PC})$. Where *value* $\mathcal{V}$ and *certainty* $\mathcal{C}$ are calculated metrics and *paths* $\mathcal{PC}$ gives the number of trust paths that were accumulated in *value* and *certainty* of this RATINGVALUE.

OPINION $\mathcal{O}$

For a better differentiation between RATINGS directly expressed by USERS and those calculated by the rating algorithm, these direct RATINGS are called OPINIONS. An OPINION is therefore a RATING whose RATINGVALUE has a *certainty* value of 1.0 and a *paths* value of 1 i.e. just one trust path is accumulated in the RATING. The textual representations of a USER's OPINION can be seen in figure 5.7.

0	0.25	0.5	0.75	1.0
deep distrust	distrust	-	trust	deep trust

Figure 5.7: Textual representation of a USER's OPINION.

Rating Calculation

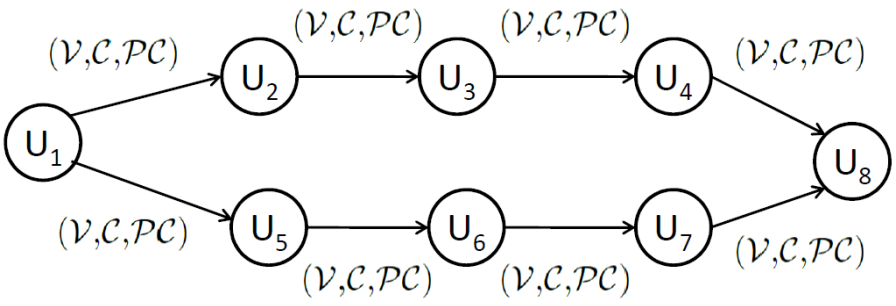

Figure 5.8: Simple trust graph with eight users

As elaborated in Sec. 4.3, a local trust algorithm should be used (R15). Therefore, a subgraph whose vertices represent USERS and weighted directed edges represent direct RATINGS i.e. OPINIONS is the starting point of trust calculation. For example in figure 5.8 USER $\mathcal{U}_2$ RATING of USER $\mathcal{U}_3$ is represented by the edge going from $\mathcal{U}_2$ to $\mathcal{U}_3$. More precisely it's the subgraph containing all paths $\mathcal{P}_{ij}$ going from one specific USER $\mathcal{U}_i$ to a specific USER $\mathcal{U}_j$. For an example see figure 5.8 which shows the subgraph containing all paths $\mathcal{P}_{18}$ going from USER $\mathcal{U}_1$ to USER $\mathcal{U}_8$. This subgraph can be used to calculate a RATING $\mathcal{R}_{18}$ that rates USER $\mathcal{U}_8$ and whose *owner* is USER $\mathcal{U}_1$. This subgraph is generated by the helper method *getSubgraph($\mathcal{U}_i$,$\mathcal{U}_j$)* which will be described later. For a less complicated description of the calculation algorithm this subgraph is defined as a set of all paths $\mathcal{P}_{ij}$ going from one specific USER $\mathcal{U}_i$ to the specific USER $\mathcal{U}_j$. The main method of the trust calculation algorithm can be seen in figure 5.9.

```
R_ij = kTrust(getSubgraph(U_i,Uj))
function kTrust(P_ij)
  rvAll = new RatingValue(1.0,1.0,1)
  foreach p ∈ P_ij do
    if validPath(p) then
      rv = new RatingValue(1.0,1.0,1)
      while hasNextEdge(p) do
        if rv < threshold then break
        rvNext = getRatingValue(nextEdge(p))
        if !hasNextEdge(p) AND rvNext < 0.5 then
          rv = mergeDt(rv,rvNext)
        else
          rv = merge(rv,rvNext)
        end if
      end
    end if
  end
return new Rating(U_i,Uj,rvAll)
```

Figure 5.9: Trust rating algorithm in pseudo code

Function *kTrust* is called with a set of paths $\mathcal{P}_{ij}$ going from USER $\mathcal{U}_i$ to USER $\mathcal{U}_j$. While traversing every valid path from the beginning to the end , the RATINGVALUES of the edges are merged. A valid path is a path either containing only edges with RATINGVALUES representing trust i.e. have *values* greater than or equal 0.5 see figure 5.7. or a path where

all but the last edge contain RatingValues representing trust and the last edge contains a RatingValue representing distrust i.e. has a *value* less than 0.5 (see requirement R13). If the last edge represents distrust, another merge function is called (see requirement R12). The merged RatingValues of the paths are added which leads to one final RatingValue expressing the accumulated Opinions of all Users involved. *kTrust* only follows paths as long as their accumulated *value* is above a certain threshold. By this way, the number of paths that are traversed is drastically reduced. This is necessary because otherwise *kNet* would in most cases traverse the whole trust network (*The small world problem* see [Travers et al., 1969]).

Finally, a new Rating is created holding this RatingValue which is owned by User $\mathcal{U}_i$ judging User $\mathcal{U}_j$.

```
function merge(RV_i,RV_j)
    return new RatingValue(V_i*V_j,V_i*C_j,min(PC_i,PC_j))
end
function mergeDt(RV_i,RV_j)
    return new RatingValue(V_j+(1-V_i)*V_j,V_i*C_j,min(PC_i,PC_j))
end
function add(RV_i,RV_j)
  return
  new RatingValue((V_i*PC_i+V_j*PC_j)/(PC_i+PC_j),(C_i*PC_i+C_j*PC_j)/(PC_i+PC_j),PC_i+PC_j)
end
```

Figure 5.10: Helper functions in pseudo code

Function *merge* takes two RatingValues $\mathcal{RV}_i$ and $\mathcal{RV}_j$ and merges them into a single RatingValues. Whereas the *value* of the new RatingValue is calculated by multiplicating $\mathcal{V}_i$ and $\mathcal{RV}_j$ like in a simple trust graph see figure 4.1, the *certainty* of the resulting RatingValue is calculated by multiplicating *certainty* of $\mathcal{RV}_j$ and *value* of $\mathcal{RV}_i$. The number of paths involved in the RatingValue is calculated in taking the minimum of $\mathcal{PC}_i$ and $\mathcal{PC}_j$ (see figure 5.10).

To satisfy requirement R12 function *mergeDt* (merge distrust) merges the *values* in a way that the relation of distrust is preserved.

Function *add* takes two RatingValues $\mathcal{RV}_i$ and $\mathcal{RV}_j$ and adds them. The result is a single RatingValues. The new *value* (*certainty*) is the weighted average of $\mathcal{V}_i$ and $\mathcal{V}_j$ ($\mathcal{C}_i$ and $\mathcal{C}_j$). As weightings the numbers of paths involved $\mathcal{PC}_i$ and $\mathcal{PC}_j$ are used. The new number of *paths* involved is the sum of $\mathcal{PC}_i$ and $\mathcal{PC}_j$ (see figure 5.10).

Function getSubgraph takes two users $\mathcal{U}_i$ and $\mathcal{U}_j$ and returns a set of paths $\mathcal{P}_{ij}$ containing all trust paths going from $\mathcal{U}_i$ to $\mathcal{U}_j$. To avoid cycles breadth-first search is used. While traversing paths, edges going out of non actual nodes pointing to the actual node are removed. Actual nodes are nodes currently visited by breadth-first search. When reaching the last node no ingoing edges are removed and the path traversed is added to $\mathcal{P}_{ij}$. For an example of a such a subgraph calculation see figures 5.11, 5.12, 5.13, 5.14 and 5.15.

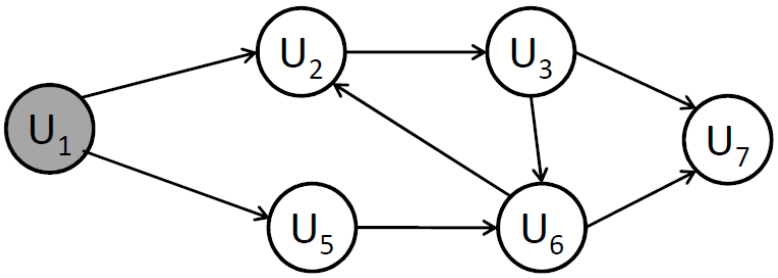

Figure 5.11: {U1}

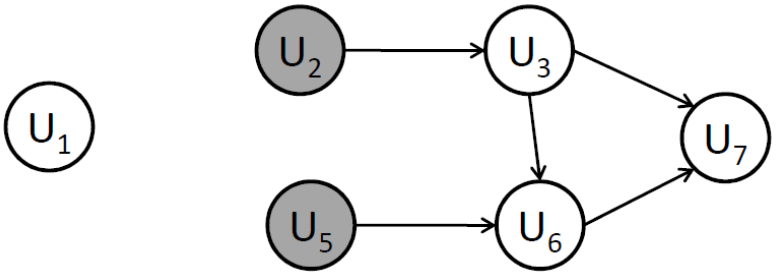

Figure 5.12: {(U1,U2),(U1,U5)}

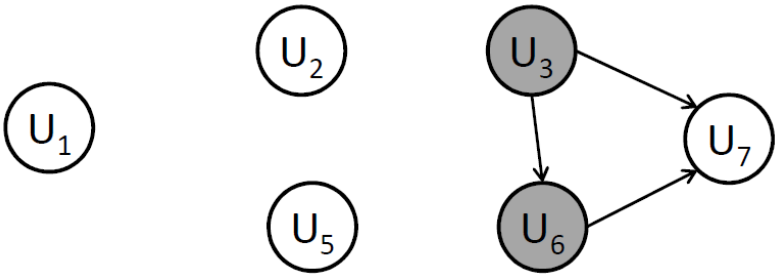

Figure 5.13: {(U1,U2,U3),(U1,U5,U6)}

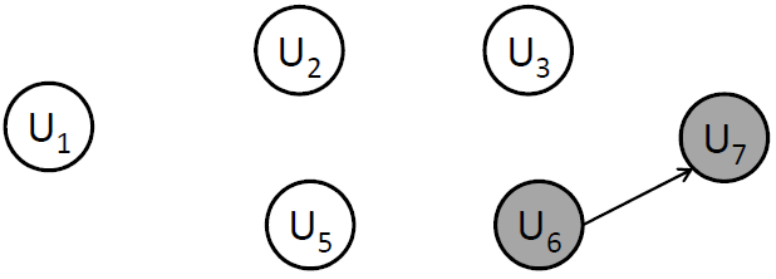

Figure 5.14: {(U1,U2,U3,U7),(U1,U2,U3,U6),(U1,U5,U6,U7)}

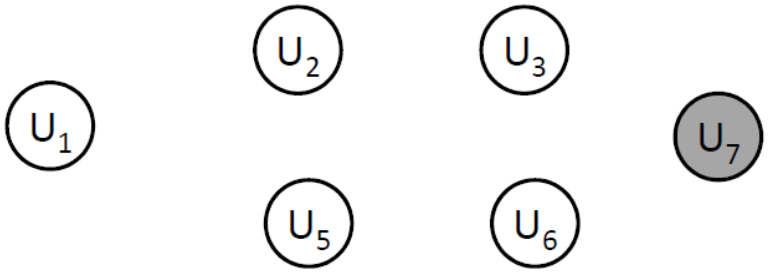

Figure 5.15: {(U1,U2,U3,U7),(U1,U2,U3,U6,U7),(U1,U5,U6,U7)}

As far as now, only the RATINGS of USERS were calculated with the trust algorithm. However, as a USER is a subclass of an ENTITY, these functions are also applicable to ENTITYS with the restriction that ENTITYS can only be the second parameter of *kTrust*. Therefore, *kTrust* can be transformed into a more general form (see figure 5.16):

$\mathcal{R}_{ij}$ = kTrust(getSubgraph($\mathcal{U}_i$,$\mathcal{E}j$))
function kTrust($\mathcal{P}_{ij}$)
  rvAll = new RatingValue(1.0,1.0,1)
  foreach p $\in \mathcal{P}_{ij}$ do
    if validPath(p) then
      rv = new RatingValue(1.0,1.0,1)
      while hasNextEdge(p) do
        if rv < threshold then break
        rvNext = getRatingValue(nextEdge(p))
        if !hasNextEdge(p) AND rvNext < 0.5 then
          rv = mergeDt(rv,rvNext)
        else
          rv = merge(rv,rvNext)
        end if
      end
    end if
  end
return new Rating($\mathcal{U}_i$,$\mathcal{E}j$,rvAll)

Figure 5.16: General rating algoritm in pseudo code

$\mathcal{R}_{ij}$ now stands for USER $\mathcal{U}_i$'s rating of ENTITY $\mathcal{E}_j$ and the set $\mathcal{P}_{ij}$ now stands for all paths going from USER $\mathcal{U}_i$ to ENTITY $\mathcal{E}_j$.

5.4 Reasoning

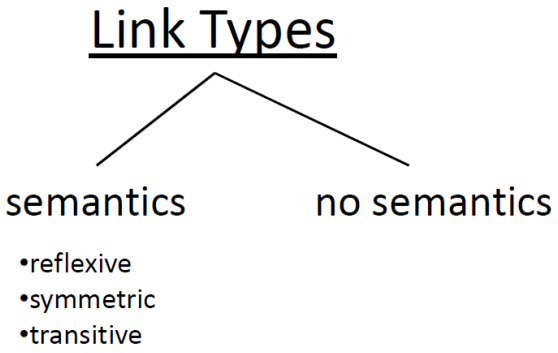

Figure 5.17: LINKS could whether support *semantics* or not

In addition to the basic support for reasoning i.e. the LINKTYPE *semantics* which indicates that a LINK supports reasoning already defined in the data model, the concept of a REASONING ANNOTATIONS $\mathcal{RA}$s is introduced.

Reasoning Annotation

A REASONING ANNNOTATION binds additional information used for reasoning to a LINK. This additional information specifies the *domain* and the *range* of the LINK and whether this LINK is *transitive*, *symmetric* or *reflexive* (see figure 5.17). These semantic characteristics are not mutually exclusive i.e. a LINK can be e.g. *transitive* and *symmetric* or fall into all three categories. Figure 5.18 shows the structure of a REASONING ANNOTATION.

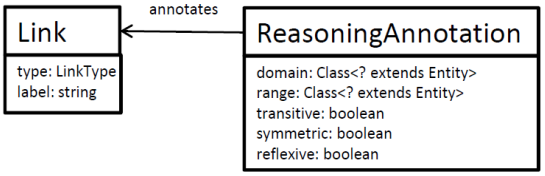

Figure 5.18: A REASONINGANNOTATION adds additional reasoning information to a LINK

To support link hierarchies, two basic links should be predefined named *broader* and *narrower*. They are used to describe hierarchical relations between LINKS. Therefore, their *domain* and *range* are of type LINK.

To enable basic argumentation out-of-the-box i.e without the necessity of creating LINKS, additional LINKS are predefined named *leadsTo* and *contradicts*. An overview of the predefined LINKS can be seen in figure 5.19.

label	domain	range	reflexitive	transitive	symmetric
broader	LINK	LINK	no	yes	no
narrower	LINK	LINK	no	yes	no
leadsTo	FACT	FACT	no	yes	no
contradicts	FACT	FACT	no	no	yes

Figure 5.19: The predifined LINKS

5.5 User Interface

Figure 5.20 shows how the user interface could look like used to create a TEXT element. It only consists of a textfield used to fill in text and a button to start creation.

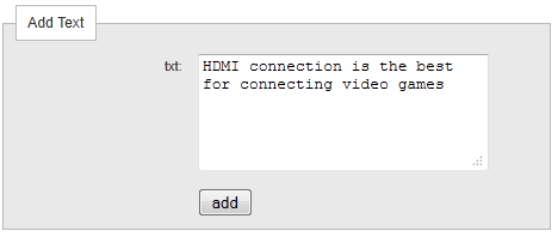

Figure 5.20: Creating a TEXT element

Creating a URL element is nearly the same task with the exception that a checksum of the corresponding web site is calculated and shown under the typed in URL. This is also a test for the existence of the web site as a URL element without checksum could not be created see figure 5.21.

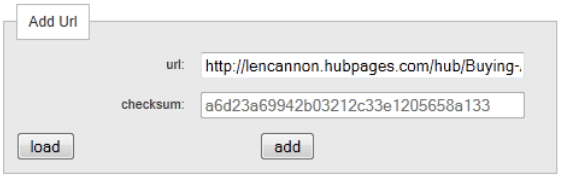

Figure 5.21: Creating an URL element

A more complex element is the SMARTURL. As described in Sec. 5.2, it is a combination of a TEXT and a URL and therefore has two textfields to fill in. One for the URL and one for the corresponding TEXT element. It also does two consistency checks. The first is the checksum calculated from the web page like it is the case for the URL including an existence check. The second even more important one, is to check whether the corresponding text snippet is actually part of the web page. The result of this check is printed out verbally in green or red color as it can be seen in figure 5.22.

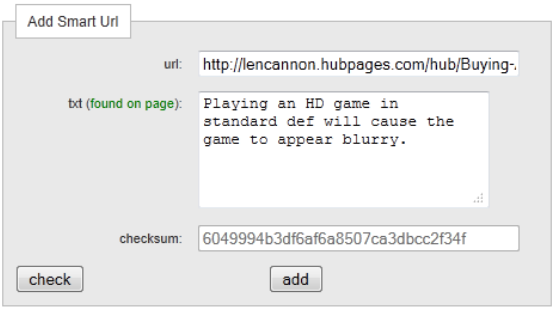

Figure 5.22: Creating a SMARTURL element

Figure 5.23 shows an example for a user interface used to create a LINK. It exists of a textfield for the *label* of the LINK, an option panel where it has to be chosen whether the LINK has *semantics*. Finally, the LINK can be sorted into the link hierarchy.

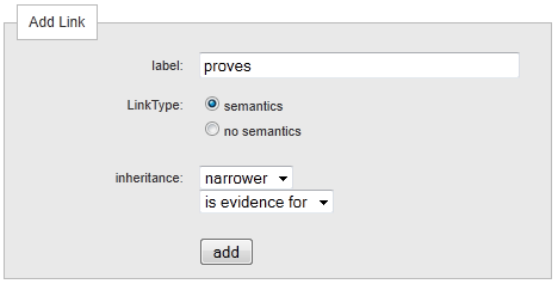

Figure 5.23: Creating a LINK element

A STATEMENT draws a connection between two ENTITYS, in most cases FACTS with the means of a LINK. Therefore, the user interface drawing such a conclusion only consists of combo boxes. The first combo box is to select the source ENTITY and the last combo box is to select the destination ENTITY. With the combo box in the middle the LINK is chosen that connects the ENTITYS (see figure 5.24).

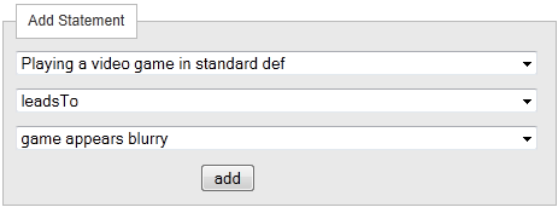

Figure 5.24: Creating a STATEMENT element

A REASONINGANNNOTATION specifies restrictions and semantic properties for LINKS. The first combo box is for choosing the LINK that should be annotated. Number two and three restrict *domain* and *range* on the chosen LINK. Finally, with the three checkboxes the semantic properties are set.

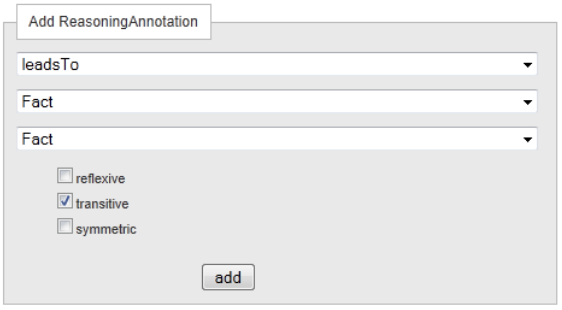

Figure 5.25: Creating a REASONINGANNOTATION

Besides the interfaces for the creation of elements there's a user interface for querying. It consists of three suggest boxes. The suggest boxes show ENTITYS found doing a text-index search using the typed in characters. They are representing the three parts of a STATEMENT *source* which could be any ENTITY, *link* which has to be a LINK and *destination* which also could be any ENTITY. A filled in suggest box restricts the result set according to its position in the corresponding STATEMENT. If the left suggest box is filled in for example it restricts the result set to ENTITYS appearing in STATEMENTS with the filled in ENTITY as its *source*. If the suggest box in the middle is completed, the

result set is restricted to ENTITYS that are *source* or *destination* of a STATEMENT with the chosen *link*. Therefore, it is obvious that ENTITYS not used in STATEMENTS can only be found with the first suggest box leaving the other suggest boxes unattended. Below the three boxes the current result set is displayed. If there exists a RATING for the ENTITY, its RATINGVALUE's *value* is displayed in brackets following the corresponding ENTITY otherwise there's a question mark. If the plus is clicked, more information is shown about the ENTITY. The stars reflect the current direct rating of an ENTITY. They also can be used to change the rating. This can be done by drawing the mouse over the stars until the desired amount of stars is activated. Then a final left click confirms the rating. This query interface can be see in figure 5.26.

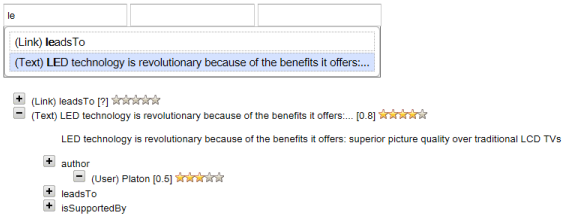

Figure 5.26: With the means of three suggest boxes the result can restricted according to a STATEMENT's *source*, *link* and *destination*

5.6 Summary

The ecology-minded girl described in use case UC.1 wants to make her decision. Starting point for the decision making process are a few rated USERS and ENTITYS. Before inferred ratings are calculated, these elements are filtered according to a specific rating threshold typically 0.2. For the remaining elements, inferred ratings are calculated. The new set of elements is filtered again according to the rating threshold. Now there are only those FACTS left she trusts in to a certain degree. Finally, the reasoning engine is used to draw logical deductions between the FACTS.

6. Implementation

This chapter describes the technologies and frameworks used in implementing the models described in the previous chapter. Figure 6.1 gives a simplified overview of the software components used in the different layers. The basis of the application is SpringMVC. It delegates requests to the corresponding layers. For the implementation of the user interface the JavaScript framework jQuery and Google Web Tookit have been used. Communication between user interface layer and the layers below is based on a REST[1]-API which uses JSON as exchange format. The calculation of RATINGS and logical deductions i.e. reasoning is based on graphs. Therefore, the java framework JGraphT is used for implementation. Data is stored in RDF with the help of the triple store sesame (openRDF). In addition to the RDF triples a fulltext-index is maintained using Apache Lucene. RDF triples and fulltext-index are linked by LuceneSAIL. On top of LuceneSAIL resides RDF2Go which is used as an abstraction layer to access the data. Users are managed by Spring Security which uses a postgreSQL database for persistence. In addition to the REST-API based on JSON, a second REST-API is available on every layer. It is used to expose the underlying data as plain RDF. Data retrieval is done by SPARQL[2]. Either plain SPARQL is encoded into the URL or it is derived from the URL. For example "/lod/urn:rnd:-72eeb624:131f22a2081:-8000" internally creates a SPARQL query querying all triples containing "urn:rnd:-72eeb624:131f22a2081:-8000" either as *subject*, *predicate* or *object*. Figure 6.3 below shows the used technologies at a glance. For are more detailed view of how the different layers are linked together see figure 6.2.

jQuery

jQuery is licensed under MIT License[3] and GPL License[4]. It is a widely used JavaScript framework. Its characteristic is that very little code has to be written. That's why it is an ideal candidate for building web prototypes with less effort. In *kNet* it is used for the user interface that creates the items. More precisely it is used to access the REST-API of the data layer via JSON calls. It is used to check existence and calculate the checksum of URLS and SMARTURLS. In creating SMARTURLS it is also used to check whether the specified TEXT is part of the web site referenced by the URL.

[1] see [Fielding and Taylor, 2000]
[2] http://www.w3.org/TR/rdf-sparql-query/
[3] http://github.com/jquery/jquery/blob/master/MIT-LICENSE.txt
[4] http://github.com/jquery/jquery/blob/master/GPL-LICENSE.txt

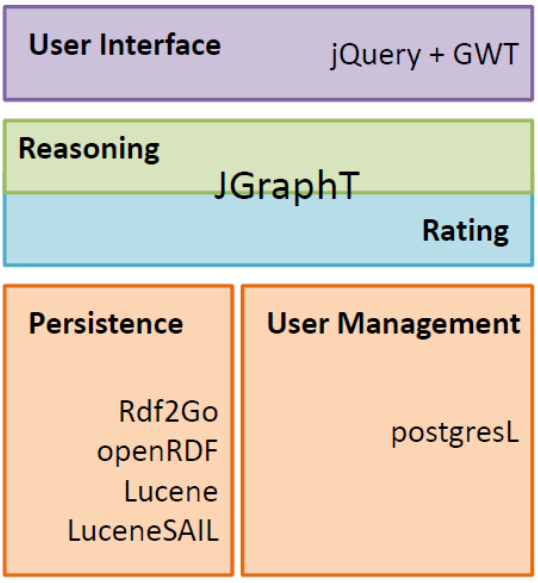

Figure 6.1: Simplistic view of the implementation of the layers

Google Web Toolkit (GWT)

Google Web Toolkit is licensed under Apache License[5]. It is a Java framework that compiles JavaScript out of Java code and therefore is the first choice for Java programmers that don't want to deal with plain JavaScript. As it exists a plugin for eclipse which includes a user interface designer it is a very convenient tool to build modern web applications. In *kNet* GWT is used for the query interface. The query interface consists of three suggest boxes representing the three parts of a statement (ENTITY LINK ENTITY). These suggest boxes are implemented with the help of GWT. They are filled with objects queried using the JSON-REST-API.

JGraphT

JGraphT is an open source graph framework written in Java. It is licensed under the GNU Lesser General Public License[6]. The strength of JGraphT is that it is a generic graph framework that can use any object as nodes and edges. In this way, on the rating layer a graph can be created out of the model objects. FACTS are used as nodes and RATINGS are used as edges. This graph is then traversed by the function *getSubgraph* as described earlier. JGraph is also used to draw logical deductions on the reasoning layer. For reasoning, FACTS are used nodes, too. But this time STATEMENTS are used as edges.

PostgreSQL

PostgreSQL is an open source relation database management system. It is licensed under PostgreSQL License [7]. Besides MySQL it is one of the most popular open source database systems. Its strengths are its maturity and robustness. It is used for user management by *Spring Security*.

[5] http://www.apache.org/licenses/
[6] http://www.jgrapht.org/LGPL.html
[7] http://www.opensource.org/licenses/postgresql

Apache Lucene

Apache Lucene is licensed under Apache License [8]. It is an open source text search engine library. Its strength are its high performance and its flexibility. In *kNet* it is used by LuceneSAIL to build a text index of the RDF data.

Sesame Triple Store

Sesame is along with Jena one of the most widely used frameworks when dealing with RDF. It is licensed under a BSD-style license [9]. Its strength is its *Storage and Inference Layer (SAIL)*. *SAIL* gives developers the flexibility to abstract from storing and inferencing of RDF. Therefore, they are able to develop extensions that can seamlessly be integrated into Sesame. This is necessary e.g. for the implementation of an extension like LuceneSAIL. In *kNet* Sesame is used to store the data of the different layers in a form that can be exposed as open linked data without transformation. It is also required for the usage of LuceneSAIL which is described in more detail below.

LuceneSAIL

As described above, Sesame offers with its *Storage and Inference Layer* a possibility to develop extensions that can be seamlessly integegrated with it. Such an extension is LuceneSAIL. It integrates Apache Lucene and Sesame. On the one hand, triples that are written to Sesame are additionally added to a full-text index. On the other hand, queries to this full-text index can be formulated in SPARQL and seamlessly executed in Sesame. LuceneSAIL is licensed under a BSD-style license [10]. Its strength is its seamless integration into Sesame. In *knet* it is used to fill the three suggest boxes of the query interface as described more precisely in Sec. 5.5. For a more detailed description of LuceneSAIL see [Minack et al., 2008].

RDF2Go[11]

RDF2Go is licensed under the new BSD license [12]. It is an abstraction layer over triple stores. It is used by the data layer to access the underlying triple store. As it uses RDF2GO, *kNet* is not bound to a specific triple store. If it turns out that there's a more suitable triple store it can be exchanged easily. This is especially a strength when parts of *kNet* should be reused.

RDFReactor[13]

RDFReactor is a software generation tool. It creates Java classes out of ontologies. It is licensed under the new BSD license [14]. In *kNet* it was used to create the corresponding Java classes out of the basis ontology. As it is more natural for programmers to work with objects in the corresponding programming language, it faciliates the manipulation of the underlying RDF data.

[8] http://www.apache.org/licenses/
[9] http://www.openrdf.org/download.jsp
[10] http://repo.aduna-software.org/svn/org.openrdf/sesame-ext/lucenesail/trunk/LICENSE.txt
[11] [Völkel, 2005]
[12] http://www.opensource.org/licenses/bsd-license.php
[13] [Völkel, 2006]
[14] http://www.opensource.org/licenses/bsd-license.php

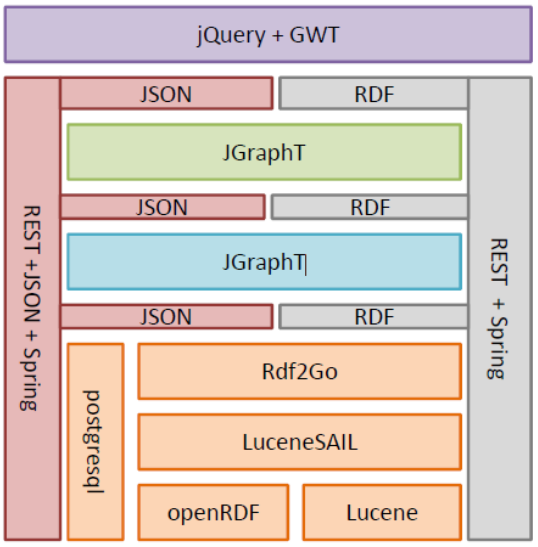

Figure 6.2: Detailed view of the implementation of the layers

Spring

Spring is an open source framework for java developement. It is licensed under the apache license[15]. The two *Spring* components used are *Spring MVC* and *Spring Security*. *Spring MVC* is a web application framework following the paradigma model, view and controller[16]. It can be used to build web applications very fast because nearly no boiler plate code[17] has to be written. *Spring Security* is a very convenient framework dealing with user management in web application. In *kNet* spring is used as the basis framework for the web application. It initializes and wires all the components used and is in charge of the communication between the corresponding layers via REST. *Spring Security* is seamlessly integrated into *SpringMVC*. It handles the creation of new users and manages access restrictions.

JSON

JSON is the Javascript Object Notation [18]. Because ajax frameworks already are implemented in JavaScript, it is a very convenient way to use JSON rather than XML or SOAP as an interchange format for REST-APIs. In *knet* JSON is used as the interchange format for REST interfaces of the different layers. For a better handling of JSON Google's Gson framework is used. It can be used to generate Java objects out of JSON and vice versa. Gson is licensed under the Apache License 2.0[19].

[15] http://www.apache.org/licenses/LICENSE-2.0.html
[16] For a more detailed consideration of design patterns see [Gamma, 2010].
[17] Boiler plate code is source code dealing e.g. with the management of classes
[18] [Crockford, 2006]
[19] http://www.apache.org/licenses/LICENSE-2.0

Name	License	Strengths	Area of usage
jQuery	MIT,GPL	very little amount of code	user interface
Google Web Toolkit	Apache	eclipse plugin and gui designer	user interface
JGraphT	LGPL	very flexible	rating and reasoning
PostgreSQL	PostgreSQL	maturity and robustness	user management
Apache Lucene	Apache	high performance flexibility	indexing RDF
Sesame	BSD-style	Storage and Inference Layer (SAIL)	storaging RDF
LuceneSAIL	BSD-style	seamless integration into Sesame	plain text querying
RDF2GO	new BSD	exchangeability of triple store triple store	accessing triple store
RDFReactor	new BSD	easy mapping between RDF and Java	Java object generation
SpringMVC	Apache	model view controller (MVC) pattern	integration of the components
SpringSecurity	Apache	seamless integration into Spring	user management
Gson	Apache	easy mapping between JSON and Java	communication between layers

Figure 6.3: Used technologies at a glance

7. Evaluation

This chapter evaluates the entire work. First Sec. 7.1 checks whether the developed models and their implementation from the previous chapters suffice the elaborated requirements. Finally, Sec. 7.2 verifies the models and implementation with the means of the use cases with which all started. The best area to demonstrate the usage of *kNet* is purchase advice. Especially when buying a new eletronic device for a example a new television many considerations have to be taken into account. Therefore, use case UC2 is described in detail, the other use cases are only touched on.

7.1 Requirements

R.1 Data should at least differentiate between users, arguments and relations

This requirement is fully satisfied. Whereas users are modeled through a concept with the same name, arguments correspond to FACTS and relations to LINKS.

R.2 + R.3 Data model should be able to express ratings between users and should be able to express ratings of arguments by users

As ENTITY is the superclass of USERS and FACTS and RATINGS could be attached to any ENTITY these requirements are also satisfied.

R.4 Every argument should have an author because trust in arguments is directly related to trust in their authors

As described earlier, an argument corresponds to a FACT. As depicted in the data model, every FACT has an *author* and therefore fulfills this requirement.

R.5 There should be a possibility to attach semantics to relations to enable reasoning

The concept of REASONINGANNOTATION has been developed to deal with this requirement.

R.6 Every element should have an HTTP URI

As every element is a subclass of ENTITY and ENTITY is addressable via an URI, this requirement is also met.

R.7 Data should be stored in a form that it can easily be transformed into RDF and SPARQL e.g. produce at least 4-star data

As described in the implementation chapter data is stored directly in RDF via the Sesame Triple Store. Therefore, data not even needs a transformation to be exposed as RDF or SPARQL.

R.8 Direct trust should have a higher rating value than indirect trust

As trust values are multiplicated when propagating trust along paths, direct trust in almost all cases has a higher value than indirect trust.

R.9 The relation of distrust should be preserved when propagating

This requirement is met by the developed trust rating algorithm. To satisfy this requirement the sub function *mergeDt* has been introduced. It differentiates between paths ending with trust and paths ending with a distrust edge and assures that the relation of distrust is preserved.

R.10 From distrust nothing can be inferred

As paths whose Rating does not exceed a specific treshold are discarded of the rating calculation and the reasoning engine, nothing can be inferred from distrust. This is done with the funtion *validPath* in *kNet* which discards path that are not valid. A valid path is defined as a path either containing only edges representing trust i.e. a RatingValue's *value* greater than or equal 0.5 or a path where all but the last edge represent trust and the last edge represents distrust i.e. a RatingValue's *value* less 0.5. In doing so nothing can be inferred from distrust.

R.11 Functions for trust aggregation should be self-maintainable

This requirement was also directly used in the design of the trust calculation algorithm and therefore is fully satisfied. As described in Sec 5.3., trust is always calculated incrementally which means that it is self-maintainable.

R.12 The aggregation of trust values from multiple paths should be considered higher than one path with the same value

The third scalar of the RatingValue is a direct consequence of this requirement. Because the number of aggregated paths is stored in the Rating, this information can be used directly when comparing Ratings.

R.13 Trust metric should at least differentiate between trust and uncertainty

The proposed trust metric not only differentiates between trust and uncertainty but takes also the number of paths added into account.

R.14 Overlapping paths should not bias trust rating

This requirement is not met. It was discarded according to a less complex trust calculation. To overcome this problem the RatingValue could be extended by a fourth value counting overlapping pathes during caculation. This value could be used to correct the calculated trust value.

R.15 A local trust algorithm should be used

The trust algorithm described in Sec. 5.3 calculates trust between two peers and therefore can be classified as a local trust algorithm.

R.16 At least transitive links should be supported

In fact transitive, reflexive and symmetric links are supported.

7.2 Use Cases

UC2. A patriotic man who likes playing video games wants to buy a new tv.

He first visits a respective shopping website to look for the top sellers because his basic assumption is that a television brand bought by many people is probably a good brand. He also considers the corresponding user reviews and their ratings to gather an overview of the brand's reputation (see UC.2.1). For an example of this first step see figure 7.1. It shows the bestseller list for televisions on www.amazon.co.uk. This can be used as a starting point for the purchase decision.

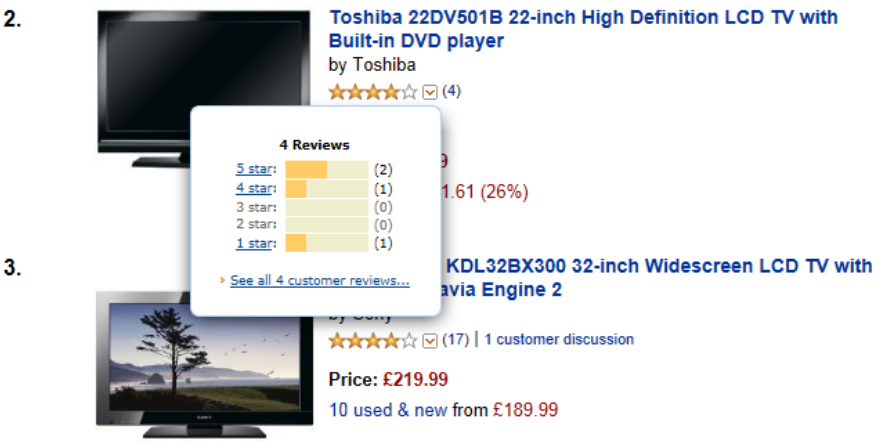

Figure 7.1: Bestseller list for televisions on www.amazon.co.uk

As easily can be seen in figure 7.2 there's not always an agreement about the quality of a product between different buyers. This is often a result of different expectations. Like our patriot with a favour for video games, these users have something in mind they want to do with the product. Another problem with online reviews on shopping websites is that users often don't differentiate between the product itself and the service of the online store.

Therefore, a product could be rated poorly only because the customer is unsatisfied with the shipping. Also the weightings of the criteria, the reviewers rating is based on, are very unbalanced i.e. a high quality product suitable for almost every use case could be rated poorly because it does not have a special feature the buyer expected. But even if he reads all the reviews discarding those which he considers not helpful (which could be a thousand or more), he is not able to decide which television to buy.

By coincidence he could find a review created by a person with a similiar attidtude (patriotic) and similiar expectations (good for video games) to the product. But even if that's the case this person could weight these two decision criteria differently i.e. he probably is a video gamer who likes to play patriotic and not a patriot who likes to play video games.

Figure 7.2: A comparison of the most helpful favourable and the most helpful critical review about television #2 from the bestseller list see figure 7.1 on www.amazon.co.uk

As it is not very likely to find such a review, he needs to split up the problem into small pieces. This strategy is outlined in Chapter 3 in the form of questions. Each question deals with a specific aspect of the problem. Now as the promising brands have been identified with the help of amazon, the patriotic aspect needs to be considered.

He needs to carry out an investigation about the manufacturer of the product. Therefore, he browses official sources like the web page of the corresponding manufacturer or wikipedia for a primarily objective evaluation of the manufacturer. But he also visits discussion boards where people post their personal opinions about the specific company. All this informations needs to be put together to form a personal view of the company (see UC.2.2). When all political aspects have been considered which are the most important aspects for our patriot, the usage of the televison is focused. He now gathers as many information as possible about the important technical properties of a television that are needed for gaming.

This again can be done by browsing several technical discussion boards, looking at the specification of games and the websites of the corresponding manufacturers (see UC.2.3). In addition to his patriotic attitude and his weakness for video gaming he wants to buy a sustainable product. Therefore, a fourth research has to be made. This time about the general technical specifications of the television and their sustainability (see UC 2.4 and UC 2.5). This can be done by browsing similar sources but with another focus.

As if all these researches were not time consuming and frustrating enough, he now needs to bring all this information together to make a substantiated final buying decision. It can be easily seen that this cannot be done without assistance. Because he cannot use *kNet*, he probably uses a table calculation like excel or makes notes with wordpad et cetera. Together with the notes he stores trusted opinions of users, his own opinions about specific topics, his weightings of the specific properties and so on. To reduce the collected information he removes facts he considers less important. But doing this without sofware assistance is barely possible. For this reason, it is more a gut decision than a substantiated decision.

If he had used *kNet*, things would have been easier. Facts and opinions about products can be brought together in *kNet* without needing another environment. They also can be rated by himself and and persons he trusts in directly in *kNet*. Therefore, the filtering of the facts can be done automatically with the help of the corresponding ratings. This leads to a more substantiated decision than the manual approach. Also logical deductions can be drawn to expose shortcomes of the arguments. To sum up, *kNet* assists the patriot in every task he needs to carry out. Especially the filtering of facts and the drawing of

logical deductions is barely possible without *kNet*'s assistance. Figure 7.3 shows a short comparison between standard software tools that could be used in decision making and *kNet*.

area of usage	**standard tools e.g. notepad, excel etc.**	**kNet**
collecting facts	+	++
rating facts	+	+++
filtering facts	-	+++
drawing conclusions	–	++

Figure 7.3: Standard software tools VS. *kNet*

UC1. An ecology-minded girl wants to buy a yogurt with natural ingredients

As the patriot man needed to browse technical discussion boards to find the specifications of the technologies, eco-girl's primarily goal is to find out which ingredients are natural. This can be done by browsing discussion boards about food or by studying test results of health institutes. Anyhow, the final decision has to be made by the eco-girl itself. But instead of maintaining a list of ingredients and asking on discussion boards about ingredients it uses *kNet* to link the connected facts and asks friends to state their opinions. This collaborated approach especially plays as role when it comes to questions like “”What's the amount of CO2-emission produced regarding the whole chain from filling the yogurt until placing it in the fridge”.

8. Conclusions and Outlook

As the previous chapters have shown, *kNet* is able to assist people confronted with difficult decision problems. Along with well examined classical areas of application e.g. philosophical discussions, e-participation, crime investgation, collaborative engineering et cetera, purchase advice is a very promising area which remained previously relatively unregarded. There exists a huge amount of discussion threads on the web about the pros and cons of product features. The question whether to buy a certain product or another is discussed as well. It is very time consuming and frustrating to make a decision because these discussions lack semantics.

As far as now, *kNet* is no more than a prototype. Further steps would be (1) to gather a critical mass of users and (2) to build a nice user interface . As *kNet* needs a certain amount of users that rate entities and other users to work, the most important next step is to gather a critical mass of users. This could be done by connecting *kNet* to successful social networks like *facebook*[1], *twitter*[2] or *myspace*[3]. Also mobile applications for smartphones are a very promising area for gathering users. As it has been shown in the short history of web 2.0, the user interface of an application is one of the key factors for the success[4] of a social application. Therefore, the beautifying of the user interface is also a very important task.

Summing up, if *kNet* gets a wide distribution, making complex decisions would be a lot easier. There's a huge amount of unstructured information on the web and people are willing to participate in collaborative decision making but they're choosy in adapting new tools. Because in deveoping *kNet* the focus has been set on layering and reusability it can be very easily attached to already adapted applications. This could be a key factor for the adaption of *kNet*.

[1] www.facebook.com
[2] www.twitter.com
[3] www.myspace.com
[4] For a more detailed look at succcess factors for web 2.0 see [Isaías et al., 2009]

List of Figures

Bibliography

[Berners-Lee, 1994] Berners-Lee, T. (1994). RFC 1630: Universal Resource Identifiers in WWW: A unifying syntax for the expression of names and addresses of objects on the network as used in the World-Wide Web. Status: INFORMATIONAL.

[Berners-Lee, 2006] Berners-Lee, T. (2006). Linked data.

[Berners-Lee et al., 1994] Berners-Lee, T., Masinter, L., and McCahill, M. (1994). RFC 1738: Uniform resource locators (URL). Updated by RFC1808, RFC2368 [Fielding, 1995, Hoffman et al., 1998]. Status: PROPOSED STANDARD.

[Bizer, 2009] Bizer, C. (2009). The emerging web of linked data. *IEEE Intelligent Systems*, 24:87–92.

[Blackmore, 2000] Blackmore, S. (2000). *The Meme Machine*. Oxford University Press.

[Bratt, 2007] Bratt, S. (2007). Semantic web, and other technologies to watch. http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/(24).

[Buckingham Shum, 2008] Buckingham Shum, S. (2008). Cohere: Towards web 2.0 argumentation. In *Proceeding of the 2008 conference on Computational Models of Argument: Proceedings of COMMA 2008*, pages 97–108, Amsterdam, The Netherlands, The Netherlands. IOS Press.

[Chesnevar and McGinnis, 2006] Chesnevar, C. and McGinnis, J. (2006). Towards an argument interchange format. *The Knowledge Engineering Review*, 21(04):293–316.

[Conklin and Begeman, 1988] Conklin, J. and Begeman, M. L. (1988). gibis: a hypertext tool for exploratory policy discussion. In *Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, CSCW '88, pages 140–152, New York, NY, USA. ACM.

[Crockford, 2006] Crockford, D. (2006). RFC 4627: The application/json media type for javascript object notation (json).

[De Liddo and Buckingham Shum, 2010] De Liddo, A. and Buckingham Shum, S. (2010). Cohere: A prototype for contested collective intelligence. In *ACM Computer Supported Cooperative Work (CSCW 2010)*.

[Fielding, 1995] Fielding, R. (1995). RFC 1808: Relative uniform resource locators. Updates RFC1738 [Berners-Lee et al., 1994]. Updated by RFC2368 [Hoffman et al., 1998]. Status: PROPOSED STANDARD.

[Fielding and Taylor, 2000] Fielding, R. T. and Taylor, R. N. (2000). Principled design of the modern web architecture. In *Proceedings of the 22nd international conference on Software engineering*, ICSE '00, pages 407–416, New York, NY, USA. ACM.

[Gamma, 2010] Gamma, E., editor (2010). *Design patterns : elements of reusable object-oriented software*. Addison-Wesley professional computing series. Addison-Wesley, Boston, Mass., 38. print edition.

[Goodwin and Fisher, 2000] Goodwin, J. and Fisher, A. (2000). Wigmore's chart method. *Informal Logic*, 20.

[Haller, 2010] Haller, H. (2010). Knitting the knet – towards a global net of knowledge. *Open Journal of Knowledge Management*, 1(1).

[Han, 2005] Han, J. (2005). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Hang et al., 2009] Hang, C.-W., Wang, Y., and Singh, M. P. (2009). Operators for propagating trust and their evaluation in social networks. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pages 1025–1032, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[Hartmanis and Stearns, 1965] Hartmanis, J. and Stearns, R. E. (1965). On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:pp. 285–306.

[Hoffman et al., 1998] Hoffman, P., Masinter, L., and Zawinski, J. (1998). RFC 2368: The mailto URL scheme. Updates RFC1738, RFC1808 [Berners-Lee et al., 1994, Fielding, 1995]. Status: PROPOSED STANDARD.

[Hupfer et al., 2009] Hupfer, S. C., Ross, S. I., Rasmussen, J. C., Christensen, J. E., Levy, S. E., Gruen, D. M., and Patterson, J. F. (2009). Crafting an environment for collaborative reasoning. In *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 379–382, New York, NY, USA. ACM.

[Isaías et al., 2009] Isaías, P., Miranda, P., and Pífano, S. (2009). Critical success factors for web 2.0 - a reference framework. In Ozok, A. and Zaphiris, P., editors, *Online Communities and Social Computing*, volume 5621 of *Lecture Notes in Computer Science*, pages 354–363. Springer Berlin / Heidelberg.

[Jøsang, 1998] Jøsang, A. (1998). A subjective metric of authentication. In Quisquater, J.-J., Deswarte, Y., Meadows, C., and Gollmann, D., editors, *Computer Security - ESORICS 98*, volume 1485 of *Lecture Notes in Computer Science*, pages 329–344. Springer Berlin / Heidelberg. 10.1007/BFb0055873.

[Klein and Iandoli, 2008] Klein, M. and Iandoli, L. (2008). Supporting Collaborative Deliberation Using a Large-Scale Argumentation System: The Mit Collaboratorium. *SSRN eLibrary*.

[Kunz and Rittel, 1970] Kunz, W. and Rittel, H. W. J. (1970). Issues as elements of information systems. Technical report, Institut fur Grundlagen der Planung.

[Lange et al., 2008] Lange, C., Bojars, U., Groza, T., and Breslin, J.G.and Handschuh, S. (2008). Expressing argumentative discussions in social media sites. In *The 1st International Workshop on Social Data on the Web (SDOW 2008) at the 7th International Semantic Web Conference (ISWC 2008)*.

[Lassila et al., 1998] Lassila, O., Swick, R. R., Wide, W., and Consortium, W. (1998). Resource description framework (rdf) model and syntax specification.

[Massa and Avesani, 2005] Massa, P. and Avesani, P. (2005). Controversial users demand local trust metrics: an experimental study on epinions.com community. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 1*, pages 121–126. AAAI Press.

[Minack et al., 2008] Minack, E., Sauermann, L., Grimnes, G., Fluit, C., and Broekstra, J. (2008). The sesame lucenesail: Rdf queries with full-text search.

[Passant et al., 2010] Passant, A., Bojars, U., Breslin, J. G., and Decker, S. (2010). The sioc project: semantically-interlinked online communities, from humans to machines. In *Proceedings of the 5th international conference on Coordination, organizations, institutions, and norms in agent systems*, COIN'09, pages 179–194, Berlin, Heidelberg. Springer-Verlag.

[Reed and Rowe, 2004] Reed, C. and Rowe, G. (2004). Araucaria: Software for argument analysis, diagramming and representation. *International Journal of AI Tools*, 14:961–980.

[Rittel and Webber, 1973] Rittel, H. W. J. and Webber, M. M. (1973). Dilemmas in a general theory of planning. *Policy Sciences*, 4(2):155–169.

[Roberts, 2000] Roberts, N. (2000). Wicked problems and network approaches to resolution. *The International Public Management Review*, 1.

[Rowe and Reed, 2006] Rowe, G. and Reed, C. (2006). Translating wigmore diagrams. In *Proceeding of the 2006 conference on Computational Models of Argument: Proceedings of COMMA 2006*, pages 171–182, Amsterdam, The Netherlands, The Netherlands. IOS Press.

[Schneider et al., 2010] Schneider, J., Passant, A., Groza, T., and Breslin, J. G. (2010). Argumentation 3.0: how semantic web technologies can improve argumentation modeling in web 2.0 environments. In *Proceeding of the 2010 conference on Computational Models of Argument: Proceedings of COMMA 2010*, pages 439–446, Amsterdam, The Netherlands, The Netherlands. IOS Press.

[Simon, 1973] Simon, H. A. (1973). The structure of ill structured problems. *Artificial Intelligence*, 4(3-4):181 – 201.

[Toulmin, 1958] Toulmin, S. (1958). *The Uses of Argument*. Cambridge University Press.

[Travers et al., 1969] Travers, J., Milgram, S., Travers, J., and Milgram, S. (1969). An experimental study of the small world problem. *Sociometry*, 32:425–443.

[Tzagarakis et al., 2009] Tzagarakis, M., Gkotsis, G., Hatzitaskos, M., Karousos, N., and Karacapilidis, N. (2009). Cope_it!: argumentative collaboration towards learning. In *Proceedings of the 9th international conference on Computer supported collaborative learning - Volume 2*, CSCL'09, pages 126–128. International Society of the Learning Sciences.

[Völkel, 2005] Völkel, M. (2005). Writing the semantic web with java.

[Völkel, 2006] Völkel, M. (2006). Rdfreactor – from ontologies to programatic data access. In *Proc. of the Jena User Conference 2006*. HP Bristol.

[Walton, 1996] Walton, D. (1996). *Argument Schemes for Presumptive Reasoning*. Lawrence Erlbaum Associates.

[Walton, 2010] Walton, D. (2010). Types of dialogue and burdens of proof. In *Proceeding of the 2010 conference on Computational Models of Argument: Proceedings of COMMA 2010*, pages 13–24, Amsterdam, The Netherlands, The Netherlands. IOS Press.

[Weick, 1995] Weick, K. (1995). *Sensemaking in organizations*. Foundations for organizational science. Sage Publications.

[Wigmore, 1913] Wigmore, J. H. (1913). The problem of proof. *Illinois Law Review*, 8:77–103.

[Ziegler and Lausen, 2004] Ziegler, C.-N. and Lausen, G. (2004). Spreading activation models for trust propagation. In *Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04)*, EEE '04, pages 83–97, Washington, DC, USA. IEEE Computer Society.

[Ziegler and Lausen, 2005] Ziegler, C.-N. and Lausen, G. (2005). Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7:337–358. 10.1007/s10796-005-4807-3.