ARGONTONAUT: An Argumentation Ontology for Mailing Lists

Colin Fraser, Harry Halpin, Kavita E. Thomas*
colinf@inf.ed.ac.uk hhalpin@ibiblio.org kavita.e.thomas@gmail.com

School of Informatics
University of Edinburgh

Abstract.

1 Introduction

2 Related Work

Argumentation is commonly construed as involving a way of organising statements in a way that allows for a structured justification of a particular position, where statements are seen as essentially defeasible and perhaps also subjective in some way. The representation of argument thus tries to expose how these justifications are linked. Unlike proof arguments, an agent is not commited to inferring a conclusion from a series of premises. But like proof arguments, one can see the structure through which a particular conclusion may be obtained. The modern study of argument was revived by Toulmin (1958) who investigated what is involved in establishing conclusions in the production of arguments, and modern argumentation schemes essentially derive from his attempts to represent this process.

The development of *Issue Based Information Systems* (IBIS) springs from work in the field of Computer Supported Collaborative Work (CSCW) by Rittel and Webber (1973) which identified the issue of "wicked" problems. IBIS is composed of three distinct entities, *Issues, Positions* and *Arguments* consisting of the relations *supports, objects-to, replaces*, etc. (Buckingham Shum, 2003). An *issue* may be characterised in terms of a *position* that an agent may take, and an *argument* an agent may put forward for taking this *position*. In this way, *issues* may evolve in a manner which exposes their continual reformulation depending on the context of use.

It is common to look at the process of argumentation as being characterised by two distinct aspects: *having arguments*, which is where individuals collaboratively try to pursue disagreement and controversy, and *making arguments*, which is the process of giving justifications and making inferences from this to come to some form of action (O'Keefe, 1977; de Moorl and Efimova, 2004). We focus on the latter of these, and view the process of argumentation as being in some sense

* Authors' names appear in alphabetical order.

goal directed, where agents are working collaboratively to reach some shared objective by giving justifications for what it is they assert. Theorists also tend to distinguish different types of argument dialogues into *persuasive* dialogues involving conflicting points of view, *negotiation*, involving a conflict of interest and a need for cooperation and *deliberation*, where there is a need to agree on a course of action (Walton and Krabbe, 1995). In the domain of W3C mailing lists, the type of dialogue we encounter predominantly involves *deliberation*.

On the level of argumentation moves or points, there are a plethora of different taxonomies to categorise different domains' needs. Pragmaticians and cognitive scientists talk about *coherence relations*, discourse analysts address *discourse relations* and those interested in communicative purpose talk about *dialogue acts* or *speech acts*. Seminal work by Mann and Thompson (1988) proposed *Rhetorical Structure Theory* (RST), a theory of text organisation, in which text spans (at the clausal level) are identified as being either a nucleus or a satellite, and relations are posited linking satellites to nuclei according to definitions of the rhetorical purpose of the text spans. RST posits *informational* and *presentational* rhetorical relations, where the former relate the content of the text spans themselves, while the latter define relations in terms of the effect the text spans have on the reader, so for example, *concession* is *presentational* while *contrast* is *informational*. *DAMSL*, a dialogue annotation approach proposed by Allen and Core (1997), goes on to distinguish dialogue acts on several levels, most notably distinguishing between *forward-looking functions* relating the current dialogue utterance to some future segment, e.g., *information-request*, and *backward looking functions* like *agreement*.

Speech act theory is a form of conversational analysis which has been particularly influential in CSCW. Winograd and Flores implemented speech act theory in the *Coordinator* project (1986) to model email conversations. They propose a general structure of *conversation for action*, which models conversations via finite state machine transitions. The *Coordinator* project attempted to structure email by getting users to annotate the *illocutionary* force (i.e., conventional interpretation of communicative action; Levinson, 1983) of emails, e.g., *reject*. Our approach is quite similar to the *Coordinator* approach except that instead of using speech acts, we use a hybrid between the rhetorical and speech act approaches for argument, and add to that a large amount of domain-specific categories. In practice, this means that we get at a finer-degree of specificity in representing what is being communicated. That is, our ontology enables a much richer set of descriptions which is tailored to a specific domain.

There has been relatively little work on argumentation ontologies for the Semantic Web. The *DILIGENT* project proposes an argumentation ontology for the Semantic Web to model engineering design decisions, and they advocate decentralised and evolving collaborative ontology development. The main concepts in their ontology are *issues, ideas* and *arguments*, which are represented as classes. *Ideas* refer to how concepts should be formally represented in the ontology and relate ontology change operations. *Ideas* respond to *issues*, and indicate how they should actually be implemented in the ontology. *Arguments* debate

either a particular *idea* or *issue*. They exemplify domain experts proposing new *issues* to be introduced into the ontology which are then argued over and formalised through concrete *ideas*. While this approach indicates how a distributed group can achieve concensus in constructing a shared ontology, their ontology does not extend neatly to modelling argumentation in emails, where we are less interested in the process by which we arrive at an ontology than in the end result and how it applies to the data. Given the difference in their goals and ours, it is perhaps not surprising that their ontology does not fit the email data.

3 ARGONTONAUT RDF and OWL Ontologies

This ontology is designed to capture not just argumentation that occurs in emails on the mailing list, but also reference to software, documents, meeting transcriptions, etc. The main (top-level) classes including inter-class OWL and RDF relationships are:

- **Message**: corresponds to an email and contains sub-classes which pertain to argumentation features
  - **Meeting** is a sub-class of Message, and corresponds to either an IRC (*Distributed*) or *FacetoFace* meeting of the W3C Working Group (disjoint sub-classes)
- **Topic**: a discussion that has not yet been formalized as an issue or even a raised issue. It's also a change of direction in the discussion within an issue.
- **Issue**: addresses topics which have been formalised as issues and decisions and is distinguished according to the stage of group acceptance which applies to them: *RaisedIssue disjointWith AcceptedIssue disjointWith DeclinedIssue disjointWith AgreedIssue; AnnouncedDecision disjointWith AgreedDecision disjointWith ProposedDecision disjointWith AnnouncedDecision disjointWith ObjectionDecision*
- **Document**: distinguished into W3C documents and other documents, and also into submission agency (i.e., submitter), requirements, change and publication features
- **Software**: *Beta* or *Release* disjoint sub-classes
- **Group**: *InterestGroup, WorkingGroup, CoordinationGroup, Votes*
- **Action**: *subClassOf* RDF Schema class *Resource*, typically corresponds to someone committing to perform an action item
- **Person**: *Attendees* or *Nonattendees* at a Meeting, disjoint sub-classes

All of these classes are sub-classes of the RDF Schema Resource class, with the exception of Group, which is part of the XML Namespace Friend of A Friend ontology. An important point to be noted is that with the exception of the Message class, the other classes all introduce procedural information that arises out of the domain itself. That is, the meat of the argumentation arises via the Message subclasses.

Messages, i.e., emails, tend to be between 1 and 7 paragraphs long, and average to between 3 and 4 paragraphs per email. This naturally means that the

author makes several points in the body of one message. This is quite reasonable given the interaction mechanism of the email, which facilitates making all one's points in one message and then posting it, rather than making separate points in separate emails all successively posted. For us, this means (among other things) that emails should get annotated with multiple tags, since presumably multiple points will have been made, resulting in a bundle of argumentation tags describing what was communicated in the message rather than just one or two tags. However we do not have the many diverse relations we would otherwise have if we were annotating points rather than emails, which significantly reduces both the annotation burden and the bundle size of annotation tags. One benefit of this is a simpler taxonomy which is more readable, since we expect 2 or 3 tags, maximum 5 per email, rather than the 10-20 tags we would otherwise expect if we were annotating on the point level. Given the goal of browsing a mailing list and interpreting argumentation made in the emails, it makes more sense to annotate emails rather than points. Implementation-wise, emails have UUIDs unlike points, which makes referring to them much easier than to points, which would have to have UUIDs assigned.

The major sub-classes of *Message* are as follows:

- *AgreeMessage*: in which the current message agrees with a previous message
- *DisagreeMessage*: the current message disagrees with a previous email
- *NewInformation*: introduces new information, whether document, software, topic or issue, etc.
- *RequestInformation*: asks for information
- *StatePrinciple*: message content which states abstract principles
- *UseCase*: message which puts forward a use-case
- *Example*: message content which uses exemplification to make a point

refersTo is the main way that messages refer to other resources, whether other messages, documents, or software, and has *domain, range* and *subPropertyOf* all defined as RDF Schema class *Resource*. *refersTo* is defined as transitive in OWL, embodying a simplifying assumption we made that responses of responses recursively respond to the opening email in the thread. However, this ignores the fact that arguments by nature are dynamic, i.e. they evolve, as was argued above in the context of IBIS. The initial argument stated in the position email of the thread is rarely the same argument by the time two or three people have responded to it. People by nature tend to reinterpret and qualify arguments much like in the popular Chinese Whisper children's game, in which what is whispered from one person to the next is never the same thing at the end of the circle as at the beginning. However our simplifying assumption is not as debilitating as it seems, since the messages posted to the W3C mailing lists tend to stay very much on topic, and qualifications rarely diverge from the topic. *refersTo* would not be able to be transitive in a discussion list for which the issues themselves evolve and speakers are more speculative.

AgreeMessage and *DisagreeMessage* are defined in OWL as disjoint. This means that they can't both apply to the same message. However, we know that it's certainly reasonable for someone to play both the devil's advocate and

advocate her own perspective regarding something under discussion in the same email. Additionally the author could agree with one aspect of the preceding message while disagreeing with another aspect of the message's points. While our OWL ontology represents an idealised (and simplified) version of reality in which this does not happen, there are no RDF constraints on disjoint classes annotating the same email.

There are several different ways in which *refersTo* works which are characterised by its sub-properties. The following message properties are all sub-properties of *refersTo* and have *domain=Message*:

- *newPoint: range=NewInfo*
- *agree: range=Message; disjointWith disagree*
  - *elaboratePoint: range=AgreeMessage; subPropertyOf agree*
  - *supportingExample: range=Example; subPropertyOf agree*
- *disagree: range=Message*
  - *modifyPoint: range=DisagreeMessage; subPropertyOf disagree*
  - *counterExample: range=Example; subPropertyOf disagree*
- *provideInfo: domain=RequestInfo (subClassOf Message); range=NewInfo (subClassOf Message)*
- *proceduralPoint: range=Message*

Although the *Document* class is the biggest one in the ontology, the bulk of categories it contains mostly have to do with due process in the W3C mailing list domain. We list below the sub-classes of *Document* with their OWL relations:

- *1. W3CDocument*
  - *W3CPreStandardDocument disjointWith W3CNonStandardDocument*
    - *CandidateRecommendation disjointWith ProposedRecommendation*
  - *W3CStandard disjointWith W3CPreStandardDocument*
    - *Recommendation, ApprovedFunding*
  - *W3CNonStandardDocument disjointWith W3CStandard*
- *2-5. Draft disjointWith Standard, Normative disjointWith Informative*
- *6-7. MemberSubmission disjointWith TeamSubmission*
- *8. Note*
- *9-11. IntroduceRevisionDocument disjointWith RequestRequirementDocument, FulfillRequirementDocument disjointWith IntroduceRevisionDocument*
- *12-14. RecommendationPublicationDocument disjointWith RescindPublicationDocument, ChangePublicationDocument disjointWith RecommendationPublicationDocument*
- *15-17. NoTextChangesDocument disjointWith CorrectionChangesDocument disjointWith ConformanceCorrectionChangesDocument disjointWith NoTextChangesDocument*
- *18. Agenda*

Classes 1 through 18 are all sub-classes of the *Document* class, and reflects the terminology and process used in the W3C mailing lists. Of these, *W3CDocument* is the only class which is not flat, although the other classes are intuitively grouped in a way which is hard to encode as either RDF or OWL groupings in the ontology, since classes 2 through 18 are all siblings of *W3CDocument*. Classes 2 to 5 inclusive address general document types. Classes 6 and 7 address

who submitted the document, and 8 is a type of document. 9 through 11 have to do with asking for revisions and requirements to be fulfilled in documents and complying with those requests. Classes 12 through 14 have to do with whether the document is ready for publication or not. Classes 15 through 17 have to do with specific types of changes to a document, and finally class 18 is a type of document.

The following object properties generally have (unless otherwise stated) *domain* as *Message*.

- **introduce** *subPropertyOf newPoint;* **cite** *subPropertyOf refersTo; both have range= Document;* **publicationDraft**, *range=WorkingDraft, subPropertyOf introduce*
- **requestComments, requestRevision**, *range=Resource;* **introduceRevision**, *range=IntroduceRevisionDocument;* **fulfillRequirement**, *range= FulfillRequirementDocument*
- **lastCall**, *range=WorkingDraft, subPropertyOf RequestComments;* **implementationsCall**, *range=CandidateRecommendation;* **reviewCall**, *range= ProposedRecommendation*
- **recommendationPublication**, *range=RecommendationPublicationDocument, subPropertyOf fulfillRequirement;* **rescindPublication**, *range= RescindPublicationDocument;* **changePublication**, *range= ChangePublicationDocument*
  - *subPropert(ies)Of changePublication:* **noTextChanges**, *range= NoTextChangesDocument;* **correctionsChanges**, *range= CorrectionChangesDocument;* **conformanceCorrectionsChanges**, *range= ConformanceCorrectionChangesDocument*
  - *subPropertyOf changePublication:* **newFeatures**, *range=Recommendation*
- **draftFinding**, *range=W3CPreStandardDocument;* **archivalFinding**, *range= W3CDocument*
- **issueRaised**, *range=RaisedIssue;* **issueAccepted**, *range=AcceptedIssue;* **issueAgreed**, *range=AgreedIssue;* **issueDeclined**, *range=DeclinedIssue;* **issueMoved, issueSubsumed, issueReferred**, *range=Issue; subPropert(ies)Of refersTo:* **referredIssue**, *range=Issue;* **toGroup**, *range=Group*
- **decisionAnnounced**, *range=AnnouncedDecision;* **decisionAgreement**, *range= AgreedDecision;* **decisionProposed**, *range=ProposedDecision;* **decisionObjection**, *range= ObjectionDecision;* **decisionAccepted, decisionMaintained**, *range=Issue*
- **actionAccepted, actionPostponed, actionDropped, actionCompleted**, *range=Action;* **actionAssigned**: *domain=Person, range=Action*
- **unanimous, concensus, dissent**, *domain=Meeting, range=Issue*
- **scribe**, *domain=Meeting, range=Person, InverseFunctionalProperty*
- **attending**, *range=Attendees;* **regrets**, *range=Nonattendees; both have domain=Meeting*
- **belongGroup, groupMember**, *domain=Person, range=Group;* **haveCharter**, *domain=Group, range=Document*
  - *subPropert(ies)Of groupMember:* **invitedExpert, coChair, chair, teamContact**; *domain=Person, range=Group,*

We will now turn to some examples to illustrate how the ontology can be put to use.

3.1 Some Examples

In the text from an email below, the author refers to a document, which is why the *cite* property is used. Furthermore the author gives a supporting example, and hence the resulting property is tagged. Authors tend to make points, and while there is a strong temptation to label a point with *newPoint*, we refrain from doing so in order to reduce the plethora of such tags which would otherwise be annotated for every major point an author made in the email.

Email 1:

```
The kernel of the issue is my interpretation of
the definition of namespace as it appears in the
Namespaces recommendation. The definition is that
a namespace is a collection of names *identified*
by a URI.

[So, for example, the namespace ({lang, space},
http://www.w3.org/XML/1998/namespace) is not equal to
({lang, space, base, id}, http://www.w3.org/XML/1998/namespace)]
supportingExample

[The W3C director directed the W3C to this interpretation in
http://www.w3.org/1999/10/nsuri, which states that a recommendation
cannot add more than clarifications and bug fixes without changing
the namespace URI.] cite
```

An interesting question of whether or not to annotate the first reference to an existing document could be asked; we argue that *cite* allows us to indicate the reference to the document the first time it is mentioned, and then subsequent references to this document should be annotated as referring to this email's action. (Recall that *refersTo* is transitive.) The namespace document is listed above as an example and is therefore not annotated.

Email 2:

```
[Some of you may be aware that an issue with the
xml:id specification began erupting the day
before it became a CR.

The issue has flowered nicely into a more general
discussion of what namespaces mean and what is
the W3C policy regarding their assignment in
recommendation track documents.] referredIssue

[I've been asked to provide this information to you,
and as PureEdge AC rep I'd like to please request that
the TAG make a formal statement to all working groups
regarding these issues as soon as possible.] requestComments
```

This is a reference to an issue that has already been raised; if the issue were being raised for the first time, *RaisedIssue* should be used. The last paragraph above indicates that the author wants the TAG to make a formal statement, which is best represented by the *requestComments* tag.

There is an often ambiguous choice one must make between annotating classes or properties; in several cases, both the appropriate property and class exist, e.g., *issueAgreed* and *AgreedIssue*. However we argue that it makes more sense to annotate emails with classes unless there are more appropriate properties available, in which case, authors should annotate the property instead. While it makes more sense to think of content of an email as belonging to a class rather than being predicative, annotating predicates makes more sense than a general (and not so informative) class which happens to be the range of the property, as would occur if we annotated *Issue* instead of *referredIssue* above.

In cases where what is annotated directly relates to some previous email, document, etc., the author should annotate what their email relates to, regardless of whether a class or property has been annotated, so we would have *DisagreeMessage(Email 1)* or *refersTo(Email 1)*.

Email 3:

```
[Which would be equivalent to saying that the state of a resource
cannot change without also changing its URI. We know that is false
in the general case,] StatePrinciple

[so I hope you forgive me for rejecting *identified* as having the
meaning you suggest.] DisagreeMessage(Email 1)
```

The email below responds to Email 1:

Email 4a:

```
[No, it says that groups *should* use a template in the original
namespace-defining document that specifies either that no updates
will be made or that updates will be made only for clarification
or bug fixes. It does not say whether adding a new name to the
namespace is considered an update, nor whether such an update
shall be considered major or minor.] modifyPoint(Email 1)
```

Notice that here it would have been correct to annotate this message as *DisagreeMessage*, but we chose to annotate it as the more specific property (since there is a 1:n class to property relation, with both *disagree* and *modifyPoint* having *DisagreeMessage* as their range. Interpreting *modifyPoint* gives the *DisagreeMessage* class interpretation for free, since that is its range.

The following text is also from Email 4:

Email 4b:

```
[In my opinion, you should put aside the process issues and state
clearly what the technical benefits/drawbacks are of allowing a
new name to be added to an existing namespace.  Merely claiming
"it is defined that way in Namespaces" doesn't really argue for
anything more than changing the Namespaces recommendation to
better suit the architecture.  What is important to us is to get
the definition right for the future.] modifyPoint(Email 1)
```

Since Email 4 already has been annotated with *ModifyPoint(Email 1)*, the author would not need to add multiple annotations for each modification made, since the granularity of annotation is on the level of the email, and she/he has already annotated that this email modifies a point in Email 1.

Email 6:

```
[If you don't accept that the general ev part of your request is covered
by finding/web arch, then can you elaborate on what is missing?]
RequestInfo
```

Notice that *RequestInfo* doesn't refer directly to anything else, so we omit linking it to a prior email, document, etc.

4 Evaluation

We evaluate inter-coder agreement via the Kappa statistic (Carletta, 1996) on an unseen set of five emails from the online corpus. Comparing the results of two experts in the domain with each other and against two newcomers to the domain, we get the following results:

Table 1. Evaluation of Inter-coder Agreement

Annotators	A,E	A,P	G,G
Expert-Expert	1	4	1
Expert-Naive		1	
Naive-Naive			

5 Conclusions and Future Work

In this paper we have presented an argumentation ontology for emails in the W3C mailing list domain. Our evaluation shows good agreement between both experts in the domain and naive users on the categories, which indicates that the ontology should be simple to use.

As is usually the case, the proof is in the pudding; that is, the real evaluation results will come from real users using the ontology to annotate their emails. We hope that in a domain like the W3C mailing lists, in which users are paid to participate and are committed to achieving the goals of W3C, they will take the one minute extra to annotate their emails according to our ontology.

The next area for future work is to implement a server-side tool which enables easy annotation of one's emails to put the ontology presented here into practice.Further issues to consider include extendability of this ontology to other domains, even to other mailing list domains. We hope to explore how much this ontology scales to annotating argumentation in other domains.

References

1. Allen, J., Core, M.: Draft of DAMSL: Dialogue Annotation Mark-up in Several Layers (1997) http://www.cs.rochester.edu/research/speech/damsl/RevisedManual/
2. Buckingham Shum, S.: The Roots of Computer Supported Argument Visualization In Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making, P. Kirschner, S. Buckingham Shum and C. Carr (Eds.), Springer-Verlag, London (2003)
3. Carletta, J.: Assessing Agreement on Classification Tasks: the Kappa Statistic In Computational Linguistics vol.22(2), (1996)
4. Levinson, S. Pragmatics Cambridge University Press (1983)
5. Mann, W., Thompson, S.: Rhetorical Structure Theory Text, vol.8 (1988)
6. de Moorl, A., Efimova, L.: An Argumentation Analysis of Weblog Conversations Proceedings of the 9th International Conference on Perspective on Communication Modelling (2004)
7. O'Keefe, D.: Two Concepts of Argument Journal of the American Forensic Association, vol.13 (1977)
8. Tempich, C., Pinto, H.S., Sure, Y., Staab, S.: An Argumentation Ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies In Second European Semantic Web Conference (ESWC 2005)
9. Toulmin, S.: The Uses of Argument Cambridge University Press (1958)
10. Walton, D., Krabbe, E.: Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning Albany, NY: SUNY Press (1995)
11. Winograd, T., Flores, F.: Understanding Computers and Cognition: A New Foundation for Design Pearson Education, NJ (1986)