# Generating Responses and Explanations from RDF/XML and DAML+OIL

**Graham Wilcock** and **Kristiina Jokinen**
University of Helsinki
00014 Helsinki, Finland
graham.wilcock@helsinki.fi
kristiina.jokinen@helsinki.fi

## Abstract

The paper describes on-going work on generating dialogue responses on two levels: first, generation of factual responses from RDF/XML information; second, generation of meta-level explanations of an ontology's structure from DAML+OIL domain ontologies. We also discuss ways to use an ontology to improve a current dialogue system's responses, for example by identifying misconceptions.

## 1 Introduction

### 1.1 Practical Dialogue Systems

The emphasis in designing practical spoken dialogue systems has been on designing suitable system prompts that will successfully elicit required user input to answer the system's questions. For example using VoiceXML [VoiceXML Forum, 2000], the dialogue designer designs the whole course of the dialogue taking into account the application task's requirements: the VoiceXML language is used to specify what the system will say, what the user is allowed to say, and how to proceed from one dialogue state to another. The whole dialogue is designed to best complete the task and this underlying assumption affects the system responses. The responses are of two types: either the system provides the user with an answer or the system asks clarification questions to which the user is requested to reply. The assumption that the dialogue is designed in terms of factual information exchange results in the simplified functionality of dialogue systems: the system's answer to the original user request is a one-shot answer whereby all the information is given to the user in one go [Moore, 1995], and the system must take care to control the user's responses in order to guide her through the task space.

However, we argue that spoken dialogue systems require an approach, which concerns natural dialogues in the sense of allowing the user to express requests in a natural language and receive responses that exhibit cooperation and understanding of the user's intentions. It is thus necessary to deliberate on appropriate presentation and suitable ways in which the pieces of information will be given to the user. Although the aim need not be to mimic human-human dialogues, it is still useful to be aware of those interaction patterns that make human-human dialogues effective, friendly and enjoyable, compared to most present-day human-computer dialogues. In this way, it is possible to progress towards human-centred and activity-based design, in which the user and her needs, capabilities and desires are put in the foreground. On the other hand, it is important that the system responses are clear and unambiguous, eliciting the user's trust in the system. It is not generally acceptable for the system to show the same kind of uncertainty and fluffy presentation as humans: this is confusing because the user is not used to such 'human-like' interaction with a machine, and moreover, the system would not give an impression of a reliable service provider, an authority that speaks with a clear language.

### 1.2 Semantic Web and Ontologies

Presentation techniques are also relevant when considering the growing number of web-based applications where the user can interactively guide the search for information. Question answering systems that combine text mining with reasoning techniques also refer to the Gricean maxims that have been the basis for cooperative dialogue systems [Harabagiu *et al.*, 2000]. Especially the need has been argued for adaptive systems and completely new usability methods.

In dialogue systems, the world model has traditionally been manually built on the basis of what is important for the application, and the reasoning that the system is capable of doing is limited to simple inferences in the application domain. This of course constrains the system's portability to other domains and also hinders the system's wide coverage. Thus there is a need for extending the system's knowledge so that domain ontology and information from real-size databases could be maximally exploited in the interaction management.

The distinction between an ontology and a database is not always straightforward. In general, a database holds details of specific entities, and an ontology holds meta-level information about different types of entities. When ontologies also hold details of specific *instances* of the members of a class, this distinction becomes blurred. However, we will structure the paper according to this distinction. In Section 2 we describe XML-based spoken dialogue response generation from RDF/XML data which includes information about specific instances. This has been implemented in a research prototype. In Section 3 we discuss generation of meta-level explanations

of the structure of the ontology itself from DAML+OIL representations. This is on-going work. We then discuss in Section 4 how ontologies could be exploited in interactive dialogue systems, for example for identifying user misconceptions. This is also on-going work.

## 2 Generating responses from RDF

### 2.1 XML-based Generation

XML-based techniques for natural language generation are described by [Wilcock, 2001], based on practical experience in developing an XML-based generation component for a spoken dialogue system [Jokinen and Wilcock, 2001].

The basic approach in this form of XML-based generation is to construct a pipeline of XSLT transformations in which the transformations correspond to the processing stages in the well-known natural language generation pipeline architecture. At the start of the pipeline, a form of template-based generation is used to create a response plan tree whose leaves are domain concept messages. The response plan tree is transformed during the microplanning stages into a response specification tree whose leaves are linguistic phrase specifications. The realization stage produces a response marked up in JSML (Java Speech Markup Language) as output from the pipeline. This is passed to the speech synthesizer.

This approach has already been documented, so we will not repeat further details here. In addition to [Wilcock, 2001], there is a tutorial with step-by-step examples of the various XML processing stages [Wilcock, 2002], and a software demo [Wilcock, 2003].

The use of this generation model for spoken dialogues is discussed by [Jokinen and Wilcock, 2001] and [Wilcock and Jokinen, 2003]. In an interactive dialogue system the starting point for the generation pipeline is specified by the dialogue manager component in the form of an *agenda*, in which a set of domain concepts are given to the generator component. The information status of the concepts is tagged by the dialogue manager as old information (Topic) or as new information (NewInfo) [Jokinen *et al.*, 1998]. This status can be used in the referring expressions stage of the generator.

### 2.2 Generating from RDF/XML

As RDF can be directly encoded in XML, the XML-based generation approach described in Section 2.1 can readily be extended to generate from RDF/XML representations.

The prototype system which processes RDF representations and extracts the required content is implemented in Java using Jena [HP Labs, 2003], a Java API for manipulating RDF models. Jena includes an integrated RDF parser (ARP), an integrated RDF query language (RDQL), support for storing DAML+OIL ontologies in an RDF model, and support for persistent storage of RDF models in relational databases.

RDF descriptions like the example shown in Figure 1 can be extracted from relational databases or other persistent RDF storage systems using RDF query languages such as Jena's RDQL. This simple example is taken from the Jena tutorial [HP Labs, 2003], where it is in fact the result of combining two smaller RDF descriptions.

```
<rdf:RDF
  xmlns:rdf='http://www.w3.org/
1999/02/22-rdf-syntax-ns#'
  xmlns:vcard='http://www.w3.org/
2001/vcard-rdf/3.0#'
 >
  <rdf:Description
   rdf:about='http://somewhere/JohnSmith/'>
    <vcard:FN>John Smith</vcard:FN>
    <vcard:N rdf:nodeID='A0'/>
    <vcard:EMAIL rdf:nodeID='A1'/>
  </rdf:Description>
  <rdf:Description rdf:nodeID='A1'>
    <rdf:value>John@somewhere.com</rdf:value>
    <rdf:type rdf:resource='http://www.w3.org/
2001/vcard-rdf/3.0#internet'/>
  </rdf:Description>
  <rdf:Description rdf:nodeID='A0'>
    <vcard:Family>Smith</vcard:Family>
    <vcard:Given>John</vcard:Given>
  </rdf:Description>
</rdf:RDF>
```

Figure 1: Example RDF description from Jena tutorial

Note that the example description is about a specific person instance (John Smith), not about the general class of persons. Note also that the example uses the RDF encoding for vCard (visiting card) objects representing personal information [World Wide Web Consortium, 2001].

If the generator were limited to the information given explicitly in the RDF repesentation, without knowledge of the vCard standard, it would not be able to generate a natural-sounding response which is easy to understand. (This might also be true of a human, if asked to explain the example without knowledge of vCard). The generated response might be something like (1).

(1) User: *What do you know about John Smith?*
System: *This is a description of 'http://somewhere/JohnSmith/'. The description includes 3 items: 'vcard:FN', 'vcard:N' and 'vcard:EMAIL'.*

*The value of 'vcard:FN' is 'John Smith'.*

*The description of 'vcard:N' includes 2 items: 'vcard:Family' and 'vcard:Given'. The value of 'vcard:Family' is 'Smith'. The value of 'vcard:Given' is 'John'.*

*The description of 'vcard:EMAIL' includes a value and a type. The value is 'John@somewhere.com'. The type is 'http://www.w3.org/2001/vcard-rdf/3.0#internet'.*

### 2.3 Exploiting standards

In this case, the generator can take advantage of the fact that a predefined standard is used – the vCard standard for personal information. The generator can use predefined XSLT response plan templates for vCard information. The approach

to domain-specific text plan creation using predefined XSLT templates is described by [Wilcock, 2001]. Specific values from the RDF representation are copied into the appropriate slots in the response plan template. Slots for which there is no information are simply left empty and will not be visible in the output.

Using the vCard information, the text planning stages of the generator can create a much better response plan tree equivalent to (2).

(2) User: *What do you know about John Smith?*
System: *This is a description of John Smith identified by 'http://somewhere/JohnSmith/'. John Smith's given name is 'John'. John Smith's family name is 'Smith'. John Smith's email address is 'John@somewhere.com'. John Smith's email address is type 'internet'.*

Of course, the purpose of a natural language generation pipeline is to produce something more natural than this. By generating suitable referring expressions, the microplanning stages of the generator convert the text plan tree into a text specification tree which is equivalent to (3).

(3) User: *What do you know about John Smith?*
System: *This is a description of John Smith identified by 'http://somewhere/JohnSmith/'. His given name is 'John'. His family name is 'Smith'. His email address is 'John@somewhere.com'. It is 'internet' type.*

Note that one reason the summary in (3) seems more natural is that the generator has assumed that John Smith is male. By also performing sentence aggregation, the microplanning stages of the generator produce a text specification equivalent to (4).

(4) User: *What do you know about John Smith?*
System: *This is a description of John Smith identified by 'http://somewhere/JohnSmith/'. His given name is 'John' and his family name is 'Smith'. His email address, which is 'internet' type, is 'John@somewhere.com'.*

The actual output is marked up in JSML, Java Speech Markup Language [Sun Microsystems, 1999], for the speech synthesizer.

FreeTTS [Sun Microsystems, 2002] is a speech synthesizer implemented entirely in Java. Because it is Java-based FreeTTS can be embedded in Java servlets, and as JSML is XML-based the XSLT pipelines can easily produce JSML output. However, the current version 1.1.1 of FreeTTS has some restrictions: JSML markup is accepted but not actually applied to the speech output, and there are only English and MBROLA voices.

# 3 Generating explanations from ontologies

## 3.1 Shallow generation and ontologies

The XML-based generation approach described in Section 2 follows the strategy of *shallow generation* [Busemann and Horacek, 1998]. One of the key ideas in shallow generation is to deliberately build domain-specific generators, and not to attempt to provide general solutions.

Naturally the shallow generation approach is compatible with using a domain-specific ontology, but it seems at first sight to be incompatible with the idea of using more general ontologies. One of the issues is how to port a shallow generator from one domain to another, and whether ontologies can offer any help. Is there any principled way to move a generator from one domain-specific ontology to another for a different domain, or to extend the domain by working with merged ontologies? It is precisely these flexible possibilities which make ontologies potentially attractive.

Another key idea of shallow generation is to build generators which are task-specific. Typically a shallow generator is both domain-specific and task-specific, because the task is closely tied to the domain. In the case of generating explanations from ontologies, the task is specific: to generate an explanation.

Surprisingly, the domain is also specific: it is the domain of ontologies. Although ontologies are about many different domains at the object level, they are all ontologies at the meta-level. So a domain-specific and task-specific shallow generator (for the domain of ontologies and the task of explanation) is a feasible objective.

## 3.2 Previous work on generation from ontologies

We will briefly look at some previous work on generation from ontologies. The basic motivation for generating natural language explanations from ontologies is that the formalisms used to encode ontologies are difficult to understand.

> Our experience shows that domain experts and human final users do not understand formal ontologies codified in such languages even if such languages have a browser and a graphic user interface to display the ontology content. [Aguado *et al.*, 1998]

Although the languages referred to are older formalisms such as Ontolingua, CycL and LOOM, we cannot assume that RDF, DAML+OIL or OWL are any easier to understand.

[Aguado *et al.*, 1998] describe a system which translates the contents of a domain ontology, formalized in Ontolingua, into natural language. As part of the mapping from domain concepts to a linguistic representation they use the Generalized Upper Model (GUM) based on the earlier Penman Upper Model [Bateman *et al.*, 1990]. They describe GUM as a *linguistic ontology*. Surface realization is done with KPML.

Describing the text planning stage, [Aguado *et al.*, 1998] say that their rhetorical schemas represent standard patterns of scientific discourse. They identified a number of stereotypical paragraph templates, including definitions, comparisons, examples, and classifications. This is an important point. If a small number of explanation schemas are sufficient to generate explanations from ontologies, then the shallow generation approach described in Section 3.1 can be employed.

[Fröhlich and van de Riet, 1998] describe work on domain independent tools for generation based on "a sophisticated representation scheme using different ontologies to represent the domain knowledge for different tasks of the generation process." Like [Aguado *et al.*, 1998], they have a domain-specific layer at the top and a domain-independent layer based

on the Penman Upper Model at the bottom, with KPML for surface realization, but [Fröhlich and van de Riet, 1998] also have a middle layer based on MOOSE [Stede, 1996].

These examples of earlier pioneering work naturally used software tools such as LISP and LOOM and adopted *de facto* standards such as the Penman Upper Model and Sentence Plan Language. Of course we now wish to use cross-platform software such as Java, and follow open W3C standards such as XML, RDF and OWL.

### 3.3 Generating from DAML+OIL

We now show how the RDF/XML-based generation approach described in Section 2 can be extended to generate from a DAML+OIL ontology. The component which processes the DAML+OIL representation and extracts the required content is implemented using Jena.

Jena provides Java methods to read an ontology which is represented in DAML+OIL form and load it as a Jena model. There are also Jena methods to list all the ontology classes and to list all the properties. As a first step, this provides a starting point for verbalising the ontology contents, but raw lists of classes and properties are very difficult to understand. In order to generate something which is an *explanation* of the ontology, the classes and properties need to be organised into meaningful groups.

As a second step, the properties can be grouped within the classes they apply to. We illustrate this using the DAML+OIL ontology for vCards given in the Jena tutorial. In this case we could again use predefined knowledge about vCards as we did in Section 2, but the important question is how to generate an explanation about an ontology using only the information available in the ontology itself.

Figure 2 shows a fragment of the list of DAML classes and properties extracted from the vCard ontology in the Jena tutorial, with properties grouped within classes. From such a partially grouped list it is possible to generate some kind of partially organised explanation, as shown in (5).

(5) User: *What kind of information is in vCards?*
System: *The vCard ontology has 28 classes: car, TELTYPES, pager, work, ADRTYPES, NPROPERTIES, ADRPROPERTIES, ...*
*The NPROPERTIES class has 5 properties: Given, Suffix, Prefix, Family, Other.*
*The ADRPROPERTIES class has 7 properties: Pcode, Country, Region, Pobox, Extadd, Street, Locality.*

Some parts of the explanation in (5) may be useful, such as the short lists of properties for the classes NPROPERTIES and ADRPROPERTIES. Other parts are still very difficult to understand, such as the list of 28 classes which includes both superclasses (TELTYPES) and subclasses (car, pager).

As a third step, we can do more programming to find all the subclasses of each class. Then we can produce a better explanation, in which the concepts are grouped meaningfully. A fragment of such an explanation is shown in (6).

(6) System: *The TELTYPES class has 14 subclasses: car, pager, video, voice, work, modem, fax, home, ...*

```
<DAMLClass http://www.w3.org/2001/vcard-rdf/
3.0#car>
<DAMLClass http://www.w3.org/2001/vcard-rdf/
3.0#TELTYPES>
<DAMLClass http://www.w3.org/2001/vcard-rdf/
3.0#pager>
<DAMLClass http://www.w3.org/2001/vcard-rdf/
3.0#work>
<DAMLClass http://www.w3.org/2001/vcard-rdf/
3.0#ADRTYPES>
<DAMLClass http://www.w3.org/2001/vcard-rdf/
3.0#NPROPERTIES>
  <unique DAMLDatatypeProperty
  http://www.w3.org/2001/vcard-rdf/3.0#Given>
  <unique DAMLDatatypeProperty
  http://www.w3.org/2001/vcard-rdf/3.0#Suffix>
  <unique DAMLDatatypeProperty
  http://www.w3.org/2001/vcard-rdf/3.0#Prefix>
  <unique DAMLDatatypeProperty
  http://www.w3.org/2001/vcard-rdf/3.0#Family>
  <unique DAMLDatatypeProperty
  http://www.w3.org/2001/vcard-rdf/3.0#Other>
<DAMLClass http://www.w3.org/2001/vcard-rdf/
3.0#ADRPROPERTIES>
  <unique DAMLDatatypeProperty
  http://www.w3.org/2001/vcard-rdf/3.0#Pcode>
  <unique DAMLDatatypeProperty
  http://www.w3.org/2001/vcard-rdf/3.0#Country>
```

Figure 2: DAML classes and properties

## 4 Using Ontologies in Dialogue Systems

### 4.1 The Interact System

The Interact system [Jokinen *et al.*, 2002] is a dialogue system that provides information on Helsinki area transportation. All the system components are implemented in Java and XML. The generator design follows the XML-based approach described in Section 2.1.

On the most general level the system contains managers which handle general coordination between the system components and functional modules (such as the Input Manager, the Dialogue Manager and the Presentation Manager), and within each manager there are several agents which take care of the various tasks typical for the functional domain. The modules and agents communicate via a shared knowledge base called Information Storage, where all the information about the system state is kept.

The system exhibits a distributed dialogue management style [Kerminen and Jokinen, 2003] whereby the dialogue situations are handled by several dialogue agents dealing with such subtasks as recognition of dialogue acts and topics. Separate task agents take care of application-related goals and the system's interaction with the database. The decision of the system's next act is determined by a special dialogue evaluator agent.

The semantic content of the system response is a set of concepts, each marked as either Topic or NewInfo, depending

on its status with respect to the discourse context. NewInfo concepts encode the information that is asked or presented to the user, while the Topic concepts encode information already available in the context. The dialogue manager constructs an Agenda, a list of concepts that is then transformed into a natural language expression [Jokinen and Wilcock, 2001].

### 4.2 Using an Ontology

Even though the application domain is restricted, the system can be extended towards a more natural interaction system by providing it with an ontology that better matches the requirements of the transportation domain. Currently the Interact system exploits a frame-based knowledge representation for the the domain knowledge, consisting of such concepts as bus, departure-place, and arrival-time, and is capable of answering questions about connections between two places, arrival and departure times, and arrival and departure places. An example of the type of user enquiries and system responses in the current system is shown in (7):

(7) User: *When does bus number 70 leave?*
System: *The next one leaves at 14:35*

However, to be able to answer meta-questions like *Are there any buses going from Arabia to Pasila?* or *Does the night-bus to Herttoniemenranta go the same route as the bus during the day?*, the system needs more structured knowledge about any unspecified connection between two places, about *night-bus* and *bus* both being a means of transportation that can be specified by a route but may be instantiated as two different items with two different routes. This knowledge can be provided by an ontology that specifies the concepts needed. Also we need a reasoning mechanism that supports the use of ontology, and an ontology representation language like DAML+OIL that attempts to merge web description languages, description logics, and frame based reasoning.

### 4.3 Identifying Misconceptions

The current system has only a limited means of dealing with misconceptions. For example, if the user refers to an out-of-range route number the current response is (8):

(8) User: *When does bus number 700 leave?*
System: *I can't find number 700 in the database*

There is an important distinction here between a route number which is out-of-range and one which is in-range but happens not to be in current use. The exact range of bus route numbers is currently 11–98, but there are several numbers within this range which are not currently allocated to any route, and such numbers will not be found in the database. By contrast, 700 is not a valid route number because routes only have 2-digit numbers. An ontology should provide this kind of meta-level information about the number of digits.

Given this evidence that the user has a misconception about the possible range of route numbers, it is the dialogue manager's responsibility how to handle the misconception in the overall dialogue context. One possibility is to provide both a general clarification and a cooperative response based on the specific user query, as in (9).

(9) User: *When does bus number 700 leave?*
System: *There are no route numbers greater than 100. Do you mean bus number 70?*

Another example from the current system in which an ontology should be useful for identifying misconceptions, is the meta-level knowledge that route numbers 1–10 are reserved for trams, not buses. (Again, this does not imply that there is an instance for every number in the range 1–10. In fact, there is currently no tram number 5). The ontology should facilitate the kind of response illustrated in (10):

(10) User: *When does bus number 7 leave?*
System: *Routes 1 to 10 are* tram *routes, not* bus *routes. Do you mean* tram *number 7?*

Identifying the misconception enables the response to be generated with contrastive stress on *tram* and *bus*, in order to draw the user's attention to the possible misconception.

## 5 Conclusion

We have described on-going work on generating dialogue responses on two levels: first, generation of factual responses from RDF/XML information; second, generation of meta-level explanations of an ontology's structure from DAML+OIL domain ontologies. We also discussed ways to use an ontology to improve a current dialogue system's responses, for example by identifying misconceptions.

## References


[Aguado *et al.*, 1998] G. Aguado, A. Bañón, J. Bateman, S. Bernardos, M. Fernández, A. Gómez-Pérez, E. Nieto, A. Olalla, R. Plaza, and A. Sánchez. Ontogeneration: Reusing domain and linguistic ontologies for Spanish text generation. In *Proceedings of ECAI-98 Workshop on Applications of Ontologies and Problem-solving Methods*, pages 1–10, Brighton, 1998.

[Bateman *et al.*, 1990] John Bateman, Robert Kasper, Johanna Moore, and Richard Whitney. A general organization of knowledge for natural language processing: the PENMAN upper model. Technical report, USC/ISI, 1990.

[Busemann and Horacek, 1998] Stephan Busemann and Helmut Horacek. A flexible shallow approach to text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 238–247, Niagara-on-the-Lake, Ontario, 1998.

[Fröhlich and van de Riet, 1998] Marcel Fröhlich and Reind van de Riet. Using multiple ontologies in a framework for natural language generation. In *Proceedings of ECAI-98 Workshop on Applications of Ontologies and Problem-solving Methods*, pages 67–77, Brighton, 1998.

[Harabagiu *et al.*, 2000] Sanda Harabagiu, Marius Pasca, and Steven Maiorano. Experiments with open-domain textual question answering. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, pages 292–298, Saarbruecken, 2000.

[HP Labs, 2003] HP Labs. Jena Semantic Web Toolkit. http://www.hpl.hp.com/semweb/jena.htm, 2003.

[Jokinen and Wilcock, 2001] Kristiina Jokinen and Graham Wilcock. Confidence-based adaptivity in response generation for a spoken dialogue system. In *Proceedings of the 2nd SIGdial Workshop on Discourse and Dialogue*, pages 80–89, Aalborg, Denmark, 2001.

[Jokinen *et al.*, 1998] Kristiina Jokinen, Hideki Tanaka, and Akio Yokoo. Planning dialogue contributions with new information. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 158–167, Niagara-on-the-Lake, Ontario, 1998.

[Jokinen *et al.*, 2002] Kristiina Jokinen, Antti Kerminen, Mauri Kaipainen, Tommi Jauhiainen, Graham Wilcock, Markku Turunen, Jaakko Hakulinen, Jukka Kuusisto, and Krista Lagus. Adaptive dialogue systems - Interaction with Interact. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue*, pages 64–73, Philadelphia, 2002.

[Kerminen and Jokinen, 2003] Antti Kerminen and Kristiina Jokinen. Distributed dialogue management in a blackboard architecture. In *Proceedings of the EACL-2003 Workshop on Dialogue Systems: Interaction, adaptation and dialogue management styles*, pages 55–62, Budapest, 2003.

[Moore, 1995] Johanna Moore. *Participating in Explanatory Dialogues*. MIT Press, 1995.

[Stede, 1996] Manfred Stede. *Lexical Semantics and Knowledge Representation in Multilingual Sentence Generation*. PhD thesis, University of Toronto, 1996.

[Sun Microsystems, 1999] Sun Microsystems. Java Speech Markup Language Specification, version 0.6. http://java.sun.com/products/java-media/speech/, 1999.

[Sun Microsystems, 2002] Sun Microsystems. FreeTTS: A speech synthesizer written entirly in the Java programming language. http://freetts.sourceforge.net/, 2002.

[VoiceXML Forum, 2000] VoiceXML Forum. Voice eXtensible Markup Language VoiceXML, Version 1.00. http://www.voicexml.org/, 2000.

[Wilcock and Jokinen, 2003] Graham Wilcock and Kristiina Jokinen. Generation models for spoken dialogues. In *Natural Language Generation in Spoken and Written Dialogue, Papers from the 2003 AAAI Spring Symposium*, pages 159–165, Stanford, 2003. American Association for Artificial Intelligence.

[Wilcock, 2001] Graham Wilcock. Pipelines, templates and transformations: XML for natural language generation. In *Proceedings of the 1st NLP and XML Workshop*, pages 1–8, Tokyo, 2001.

[Wilcock, 2002] Graham Wilcock. XML-based Natural Language Generation. In *Towards the Semantic Web and Web Services: XML Finland 2002 Slide Presentations*, pages 40–63, Helsinki, 2002.

[Wilcock, 2003] Graham Wilcock. Integrating Natural Language Generation with XML Web Technology. In *Proceedings of the Demo Sessions of EACL-2003*, pages 247–250, Budapest, 2003.

[World Wide Web Consortium, 2001] World Wide Web Consortium. Representing vCard Objects in RDF/XML. http: //www.w3.org/TR/vcard-rdf, 2001.