Using Linguistic Knowledge for Fine-tuning Ontologies in the Context of Requirements Engineering

JÜRGEN VÖHRINGER¹, DORIS GÄLLE¹, GÜNTHER FLIEDL¹, CHRISTIAN KOP¹, MYKOLA BAZHENOV²

¹Alpen-Adria-Universität Klagenfurt
²National Technical University “Kharkov Polytechnic Institute”

ABSTRACT

Nowadays ontology creation is on the one hand very often hand-knitted and thus arbitrary. On the other hand it is supported by statistically enhanced information extraction and concept filtering methods. Automatized generation in this sense very often evokes “shallow ontologies” including gaps and missing links. In the requirements engineering domain fine-granulated domain ontologies are needed; therefore the suitability of both hand-knitted and automatically generated gap-afflicted ontologies for developing applications can not always be taken for granted. In this paper we focus on fine-tuning ontologies through linguistically guided key concept optimization. In our approach we suggest an incremental process including rudimentary linguistic analysis as well as various mapping and disambiguation steps including concept optimization through word sense identification. We argue that the final step of word sense identification is essential, since a main feature of ontologies is that their contents must be shareable and therefore also understandable and traceable for non-experts.

Keywords: requirements engineering, ontology engineering, rule mapping, incremental linguistic analysis, WordNet querying
1 INTRODUCTION/MOTIVATION

In the last ten years the job of creating ontologies moved from an Artificial-Intelligence question to a central topic of the exploding semantic web community [17]. Usually ontology creation is very often hand-knitted and thus arbitrary or supported by statistically enhanced information extraction and concept filtering methods. This shift resulted in an uncontrolled growth of ontologies on the one hand and a heightened degree of ontology generality on the other hand. Both developments entail that many existing ontologies are not usable in real-world-applications like requirements engineering.

For supporting systematic and application oriented ontology engineering we previously researched and proposed linguistic guidelines for structuring concept and property notions in OWL represented ontologies [1]. But these guidelines only support ontology generation if domain expertise is sufficiently available in a clearly decoded manner. Obviously specific domain information quite often exists in a not explicit and ambiguous textual format. In this case the elicitation of domain specific concepts still poses many difficulties to the ontology designer.

Hence we developed a linguistic system for supporting the ontology designer to make implicit information easier to trace. Our methodology includes algorithms for tagset mapping, multi-level chunking and word sense identification. The tagging task is carried forward to QTAG, a probabilistic tagger written in Java [2]. The mapping engine we developed for splitting up standard tags into ontologically relevant tags and specific attributes generates unique input for our rule based chunker.

Some chunking heuristics needed for grouping words to morphological units and syntactical chunks are then used for decoding linguistic candidates for conceptualization nodes in the ontology layer. The main contribution of our research work presented in this paper is an efficient method for incrementally including contextual information in the ontology representation. By combining standard natural language processing methods with certain expansion strategies we definitely improve the usability of standard ontologies. Our approach preserves the basic and partially generic knowledge format for storing domain knowledge and its guided updating.

The approach consists of the following three main steps:

1) linguistic preprocessing: extracting words and phrases from natural language text
2) linguistically guided incremental ontology engineering
3) filling up ontology concept description slots through WordNet based word sense identification

The paper is structured as follows. In section 2 we give an overview of related work. In section 3 we roughly present our theoretical approach including the description of the linguistic pre-processing layers, in particular the mapping, and multi-level chunking steps. In section 4 the output of our ontology refinement tool is shown and an ontology example is described. We also propose a list of rules for ontology element creation. In section 5 we describe our Wordnet based tool for incremental optimization of standard ontologies through wordsense disambiguation. Section 6 gives a summary of the proposal presented in this paper.

2 RELATED WORK

[29] argue that the accuracy and robustness of automatically or semi-automatically engineered ontologies needs to be improved for real-world applications and they propose fuzzy algorithms for real-world-ontology engineering. [28] proposes the use of glosses in ontology engineering for improving the accuracy. We agree that for real-world applications like ontology engineering in requirements engineering projects, automatically generated ontologies might not be suitable and we therefore propose linguistic heuristics for supporting ontology creation and fine-tune ontologies through step-by-step integration of domain knowledge.

Concerning linguistic preprocessing the most relevant linguistic methods used in our approach are tagging and chunking. For tagging English free texts many open source systems like the decision based “Treetagger” [3], the rule- and transformation-based “Brill tagger” [4], the maximum-entropy “Stanford POS Tagger” [5], the trigram based probabilistic “QTAG” [2] etc. are available. For chunking some NLP toolkits exist, e.g. “MontyLingua” [8], “MontyKlu” (an online-version of “MontyLingua” developed by members of our research group in Klagenfurt [9]), the OpenNLP chunker [10] and the “NLTK Toolkit” [11]. These systems mainly provide standardized and acceptable output, but as we know according to practical requirements engineering needs they have not been tested yet.
3 LINGUISTIC PREPROCESSING

3.1 Extended Tagging format

We have chosen “QTAG” as the basis for our extended tagging format which we have adopted for these special purpose. Since QTAG is a java-based, extendable, trainable, language independent tagger, it was easy to integrate in our engineering toolset [6,7]. We extract relevant information from the QTAG output and transform it into the extended tagset format described below. Therefore, we have to use some additional methods and heuristics to elicit semantic information needed during the further processing steps of the engineering workflow. Our enriched tagset consists of standard POS-categories with lists of additional specialized attributes (e.g. v0 with subclass attribute “tvag2”\(^3\)). These attributes are necessary for identifying ontological key relations. Table 1 shows how typical standard part-of-speech tags are extracted from the QTAG output and reassigned using the NIBA tagset notation\(^4\). Additional information about concrete part-of-speech instances is presented by using fine-granulated attributes\(^5\). As an example, the verb “is” in QTAG gets the tag <BEZ>. This tag decodes, that “is” is an auxiliary verb with the inherent morphosyntactic values present tense, singular, third person and having “be” as the base form.

Table 1. Mapping Rules for mapping standard tags to attributed tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>New Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEZ</td>
<td>v0</td>
</tr>
<tr>
<td></td>
<td>verbclass=&quot;aux&quot;</td>
</tr>
<tr>
<td></td>
<td>temp=&quot;pres&quot;</td>
</tr>
<tr>
<td></td>
<td>form=&quot;ind&quot;</td>
</tr>
<tr>
<td></td>
<td>num=&quot;sg&quot;</td>
</tr>
<tr>
<td></td>
<td>ps=&quot;3&quot;</td>
</tr>
<tr>
<td></td>
<td>baseform=&quot;be&quot;</td>
</tr>
<tr>
<td>NPS</td>
<td>n0</td>
</tr>
<tr>
<td></td>
<td>type=&quot;proper&quot;</td>
</tr>
<tr>
<td></td>
<td>num=&quot;pl&quot;</td>
</tr>
</tbody>
</table>

\(^3\) We use “tvag2” for annotating a mono-transitive verb with agentive subject

\(^4\) Central NIBA tags are e.g. v0 (= main verbal element), n0 (= noun), a0 (= adjective) etc.

\(^5\) Typical tag internal attributes are “base form = go” or “type = common” etc.
3.2 Chunking rules

Based on some variants of the X-bar Theory [24] and on some core definitions in the existing NIBA Tag system [25] we composed a set of chunking rules for English for the production of syntactically and morphosyntactically motivated chunks (Table 2).

<table>
<thead>
<tr>
<th>Rule (Summands → Result)</th>
<th>Rule level</th>
<th>Rule descriptions</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>n0+n0 → n0</td>
<td>1</td>
<td>Compound Noun</td>
<td>blood pressure</td>
</tr>
<tr>
<td>[pt0]+a0 → a2</td>
<td>1</td>
<td>Adjective Phrase</td>
<td>very nice</td>
</tr>
<tr>
<td>[a0]+a0 → a2</td>
<td>1</td>
<td>Adjective Phrase</td>
<td>bright green</td>
</tr>
<tr>
<td>[pt0]+q0 → q2</td>
<td>1</td>
<td>Quantor Phrase</td>
<td>very many</td>
</tr>
<tr>
<td>[q0]+q0 → q2</td>
<td>1</td>
<td>Quantor Phrase</td>
<td>one million</td>
</tr>
<tr>
<td>[pt0]+adv0 → adv2</td>
<td>1</td>
<td>Adverb Phrase</td>
<td>very often</td>
</tr>
<tr>
<td>[adv0]+adv0 → adv2</td>
<td>1</td>
<td>Adverb Phrase</td>
<td>yesterday noon</td>
</tr>
<tr>
<td>pron0(type=pers) → n3</td>
<td>1</td>
<td>Noun Phrase</td>
<td>she</td>
</tr>
<tr>
<td>v0(verbclass=aux)+[adv0]+v0 → v0(type=complex)</td>
<td>1</td>
<td>Complex Verb</td>
<td>will certainly go</td>
</tr>
<tr>
<td>v0(verbclass=aux)+pt0(type=neg)+v0 → v0(type=complex)</td>
<td>1</td>
<td>Complex Verb</td>
<td>would not write</td>
</tr>
<tr>
<td>v0+pt0(type=verbal) → v0(type=complex)</td>
<td>1</td>
<td>Complex Verb</td>
<td>wake up</td>
</tr>
<tr>
<td>q2+q2 → q2</td>
<td>2</td>
<td>Quantor Phrase</td>
<td>two hundred million</td>
</tr>
<tr>
<td>pron0(type=poss)+n0 → n3</td>
<td>2</td>
<td>Noun Phrase</td>
<td>his mother</td>
</tr>
<tr>
<td>[det0]+[a2]+[q2]+n0 → n3</td>
<td>3</td>
<td>Noun Phrase</td>
<td>the nice two girls</td>
</tr>
<tr>
<td>[det0]+[q2]+[a2]+n0 → n3</td>
<td>3</td>
<td>Noun Phrase</td>
<td>the three busy scientists</td>
</tr>
<tr>
<td>p0+n3 → p2</td>
<td>4</td>
<td>Prepositional Phrase</td>
<td>of blood pressure measurement</td>
</tr>
</tbody>
</table>
There are several types of chunking rules, which are arranged in a certain order that should be followed during the chunking process. Summands are the array of input nodes which are needed for building the next resulting upper node of the chunking tree. Some of summands are strictly required for rule producing, they are written without square brackets, but some are not obligatory, they are placed inside brackets.

3.3 Identification of Semantic Roles

Due to the fixed and transparent subject-verb-object (SVO) structure of English, the identification of semantic roles in chunked sentences is by default a quite simple and straightforward task. According to [31] we propose automatic role labeling using partially mainly propbank and verbnet information. The Verbclass Tag in column a in Table 3 of a concrete verb triggers the assignment of a role in column c to a N3(P2) in d via indexation from left to right.

Table 3. Verb classes and their (morpho)syntactic and semantic features[23]

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Tag (a)</th>
<th>Verbclass (b)</th>
<th>PAS(^6) (c) (Argument Structure)</th>
<th>Syntactic context(^7) (d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>aux</td>
<td>Auxiliary verb</td>
<td>V-fin</td>
<td>_V0</td>
</tr>
<tr>
<td>2</td>
<td>eV</td>
<td>Ergative verb</td>
<td>[TH](^8) AGi/THi[]</td>
<td>N3_i</td>
</tr>
<tr>
<td>3</td>
<td>iV</td>
<td>Intransitive verb</td>
<td>AG/TH[i]</td>
<td>N3_i</td>
</tr>
</tbody>
</table>

\(^6\) PAS = “Predicate Argument Structure”; it includes the verb class specific semantic roles and brackets, which decode the argument status of these roles. They can have an external status (subject function) or an internal status (object function).

\(^7\) P2 stands for prepositional phrases; N3 decodes nominal phrases in our framework; N2 is a reduced nominal phrase in predicative function; N3 A2 decodes an adjective phrase in our framework. For further explanation see [23].

\(^8\) The acronyms for the default semantic roles are TH = Thema (neutral object), AG = Agens( the Actor of an Action), GO = Goal (the final point of a process), SO = Source (the starting point of a process), LOC = Location and EXP = Experiencer (a person, who undergoes the process of experiencing something).
Nevertheless we have to take into account that the phrasal structure sometimes inhibits simple solutions like for example left to right counting of nouns. Thus we used the following algorithm to cope with the problem of phrasal complexity:

- create a set of rules which can operate on simple singular term subjects and objects (e.g. proper nouns and personal pronouns);
- consult the exception database with already assigned verbal subclass tags using training sentences which include higher level argument patterns referring to more complex phrases;
- reconstruct the structure of the primarily assigned phrases if relevant morphosyntactic features don’t fit;
- leave open the possibility to manually change wrong/exceptional assignments or to add new information about verb classes, noun phrases and other patterns;

### 4 Our Approach: Linguistically Guided Incremental Ontology Creation

To avoid using non-fitting ontologies for specific domain relevant demands, particularly in requirements engineering, we take textual descriptions as a starting point for our processing. These texts are generated by filtering those text segments from extensive, domain-relevant documents, in which key words or key phrases occur, which
can be accepted as candidates for concept- or relation-notions in the ontology. Utilizing various filtering strategies, in a first step keywords in a text are identified, which are deemed important for a specific domain. Afterwards sentences from the original requirements that contain those keywords are filtered. A precondition is that these sentences form a cohesive text block. For further information about this process see [30].

In the following an examplary text segment from the medical domain is given, that was automatically selected from the original domain-related requirements text using the previously mentioned keyword filtering strategy:

With regard to the monitoring of blood pressure measurements, it is important to clearly define time and date at which the blood pressure of a hemodialysis patient is measured in each hemodialysis session.

We perform the steps of linguistic preprocessing as proposed in section 3 as a first step of transforming the textual input into a domain ontology:

- QTAG output (standard tags)
- Standard Tags transformed to enriched tags
- Chunking output

The XML output of the linguistic preprocessing can be seen in Fig. 1.

This output contains linguistic tags for words, e.g. n0: “regard”, some attributes (e.g. base-form=“regard”, type=“common”, corelex=“coa” etc.) and chunk-tags (e.g. n3: “the monitoring of blood pressure measurements”). This extended linguistic representation of the input text allows mapping and interpretation in the sense of ontology conceptualization. The Table 4 lists rules used for identifying and creating ontology elements from the preprocessed texts.

In the Table 4 some rules for the most relevant linguistic categories like N3, N0, P0 are listed. The interpretation example in the right column shows that an explicit mapping from text to class names is possible. To sum up: all relevant ontology element types are identified in an unambiguous way. The above listed rules transform linguistic annotators to ontological tags. The tags specify words in a unique manner. The output text below shows strings class candidates, relation designators, attribute identifiers and stop word material, which is filtered out during transformation:
With regard to the monitoring of blood pressure measurements, it is important to clearly define time and date at which the blood pressure of a dialysis patient is measured in each hemodialysis session.

Fig. 1. XML output of linguistic preprocessing for a fragment of the example text

\[\text{Underlined words} \text{ decode relations, dotted underlines words function as attributes, Bold words are interpreted as classes; All other elements are categorized as stop words and filtered out.}\]
Table 4. Rules for mapping of linguistic categories to ontology elements

<table>
<thead>
<tr>
<th>Rule Nr</th>
<th>Rule</th>
<th>Description</th>
<th>OWL Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N0 → Class</td>
<td>Default-Rule for N0 if no Exception applies (see Rule 2)</td>
<td>Class</td>
<td>“monitoring” → monitoring (class)</td>
</tr>
<tr>
<td>2</td>
<td>N0 (Exception) → Functional Property</td>
<td>Exception for N0 applies, if N0 is found (according to rules described in [26])</td>
<td>Functional Property</td>
<td>“time” → time (slot in class blood-pressure)</td>
</tr>
<tr>
<td>3</td>
<td>N3 → Class</td>
<td>Rules 3 to 5 always apply for N3</td>
<td>Class</td>
<td>“hemodialysis session” → hemodialysis (class)</td>
</tr>
<tr>
<td>4</td>
<td>N3 → is_a + Class</td>
<td>Rules 3 to 5 always apply for N3</td>
<td>subClassOf</td>
<td>“hemodialysis session” → isa (subClass Of) Session (class)</td>
</tr>
<tr>
<td>5</td>
<td>N3 → belongs_to + Class</td>
<td>Rules 3 to 5 always apply for N3</td>
<td>Functional Property</td>
<td>“hemodialysis session” → belongs_to (Functional Property) hemodialysis (class)</td>
</tr>
<tr>
<td>6</td>
<td>(P0</td>
<td>Functional</td>
<td>“blood”</td>
<td></td>
</tr>
<tr>
<td>Rule</td>
<td>Property</td>
<td>Object</td>
<td>Functional Property</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>--------</td>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>pressure of a hemodialysis patient</td>
<td>bp_of</td>
<td>“monitoring of blood pressure measurements”</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Cardinality of connection is n</td>
<td>“monitoring of blood pressure measurements”</td>
<td>“hemodialysis patient”</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>tvag2</td>
<td>Functional property</td>
<td>“blood pressure is measured”</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2 shows an ontology fragment which is generated by applying the transformation rules in Table 4. For representation of ontology relevant knowledge OWL [15] and RDF [16] are commonly used. We chose Protégé for representing our ontology example.

---

12 Exemplary modern toolkits for ontology engineering are Protégé [12], NeOn [13] and Chimaera [14].
5 ADDING CONCEPT DESCRIPTIONS THROUGH LEXICALLY DRIVEN WORD SENSE IDENTIFICATION

According to Gruber [27] an ontology is "[…] a formal explicit specification of a shared conceptualization". From this follows that ontology contents has to be sharable and reusable among and across projects within the domain. The ontology fragments that were created stepwise from natural language text based on the heuristics described in section 3 still have empty description slots and are therefore not easily shareable and understandable for non-domain experts. Furthermore missing empty description slots make similarity calculation based on WordNet difficult, since they presuppose a known word sense. Such
similarity calculations are important when new concepts are matched with existing ones in further ontology integration steps. For this reason we propose additional measures for fine-tuning the ontology by refining concept notions that are crucial for the specific domain. The description slots of the ontology (e.g. the example ontology in figure 2) are filled by providing a WordNet related engineering mechanism, which will be described in the following. For ontology concepts that have an empty description slot, WordNet is queried regarding the available word senses and their definitions. The following cases are distinguished:

Case 1: The concept is identified in WordNet and has exactly one meaning. This meaning is automatically assigned to the concept. Example: the concept blood pressure was identified from the natural language text, is new to the domain ontology and therefore has an empty description slot. Querying WordNet returns one possible meaning:

**blood pressure** -- the pressure of the circulating blood against the walls of the blood vessels; results from the systole of the left ventricle of the heart; sometimes measured for a quick evaluation of a person's health; "adult blood pressure is considered normal at 120/80 where the first number is the systolic pressure and the second is the diastolic pressure"

This definition is chosen and assigned to the concept, but can still be manually adapted.

Case 2: The concept is identified in WordNet but has more than one possible meaning. In this case the correct meaning is chosen from the list of available word senses. Example: the concept blood has an empty description slot and querying WordNet returns the following possible Word Senses, ordered by probability of appearance:

1. **blood** -- the fluid (red in vertebrates) that is pumped by the heart; "blood carries oxygen and nutrients to the tissues and carries waste products away"; "the ancients believed that blood was the seat of the emotions"
2. lineage, line, line of descent, descent, bloodline, blood line, **blood**, pedigree, ancestry, origin, parentage, stemma, stock -- the descendants of one individual; "his entire lineage has been warriors"
3. **blood** -- temperament or disposition; "a person of hot blood"
4. rake, rakehell, profligate, rip, **blood**, roué -- a dissolute man in fashionable society
5. **blood** -- people viewed as members of a group; "we need more young blood in this organization"
In the medical domain the first, literal, sense of the word is chosen (fluid that is pumped by the heart) and assigned to the concept.

*Case 3*: The concept is not found in WordNet. This usually means that the concept is too specialized and the probability is high that we are dealing with a compound. By applying percolative rules on endocentric compounds we determine the head of the compound, for which again the definitions are determined. Example: the concept *hemodialysis session* is too specialized and hence has no match in WordNet. However it is an endocentric compound with the head *session* and the modifier *hemodialysis*. A search for *session* returns a word sense list as described in case 2:

1. session -- a meeting for execution of a group's functions; "it was the opening session of the legislature"
2. school term, academic term, academic session, session -- the time during which a school holds classes; "they had to shorten the school term"
3. session -- a meeting devoted to a particular activity; "a filming session"; "a gossip session"
4. seance, sitting, session -- a meeting of spiritualists; "the seance was held in the medium's parlor"

The sense 3 (meeting devoted to a particular activity) is selected. Regarding the modifier, one word sense is returned as described in case 1:

*hemodialysis, haemodialysis* -- dialysis of the blood to remove toxic substances or metabolic wastes from the bloodstream; used in the case of kidney failure

The definition of hemodialysis session is thus constructed from the definition of its parts:

*hemodialysis session* -- a meeting devoted to dialysis of the blood to remove toxic substances or metabolic wastes from the bloodstream; used in the case of kidney failure;

*Case 4*: Although the concept is found in WordNet, the description is considered too specialized by a domain expert and therefore inadequate. In this case the chosen description is either manually adapted or the hypernym definition is automatically determined through WordNet querying of its hypernym’s concept definition. Example: the concept *hemodialysis* returns the definition

*hemodialysis, haemodialysis* -- dialysis of the blood to remove toxic substances or metabolic wastes from the bloodstream; used in the case
of kidney failure
Since it is considered too specialized the following hypernym definition is established:

**dialysis** -- *separation of substances in solution by means of their unequal diffusion through semipermeable membranes*

---

The process (summarized in figure 3) is a guided way of fine-tuning ontologies not according to the quantity but the quality of concepts by adding semantics in order to make them easier understandable for non-experts and facilitate reuse. Using a general lexicon like WordNet allows the standardization of definitions. A prototype implementation is available that utilizes Perl for WordNet querying and allows among other things the listing of available word senses in WordNet, the determination of hypernym definitions and the adaption of definitions where required. Furthermore word sense identification is a bidirectional process as gaps in WordNet (see case 3 above) can be identified and filled. The process above is not limited to WordNet: every lexicon providing definitions can be utilized. More comprehensive lexicons are preferable, for this reason WordNet is a good default choice.
In the requirements engineering domain fine granulated ontologies are necessary for efficient generation of models that can be further used in the application engineering steps. In this paper we proposed a step by step strategy of ontology engineering emanating from manually produced or already statistically filtered text.

Our approach focuses on the diversification of standard tags for optimizing the automatic elicitation of classes, relations and attributes in domain ontologies. Doing this with free text input can only be successful, if certain NLP standard techniques like probabilistic tagging get combined with special procedures like filtering, tag-enriching and chunking. The involved procedures are heuristically founded and follow a multilevel chunking strategy. We described a framework for mapping automatically generated linguistic categories to ontology concepts. Beyond that we showed in detail how these concepts can be refined and therefore optimized based on WordNet, in order to ensure their shareability. Our arguments are supported by a tool set that was developed in our research group for linguistically enhanced requirements engineering. The output graph of our example (see chapter 4) proves that creating ontology fragments with linguistic fine-tuning is suitable in the context of requirements engineering.

REFERENCES

2. The QTAG POS-Tagger http://morphix-nlp.berlios.de/manual/node17.html
10. OpenNLP project website: http://opennlp.sourceforge.net/
18. SNOMED Clinical Terms: Overview of the Development Process and Project Status: Michael Q. Stearns, MD', Colin Price, MPhil, FRCS, Kent A. Spackman, MD, PhD” , Amy Y. Wang, MD'

JÜRGEN VÖHRINGER
INSTITUTE OF APPLIED INFORMATICS,
RESEARCH GROUP APPLICATION ENGINEERING,
ALPEN-ADRIA-UNIVERSITÄT KLAGENFURT

DORIS GÄLLE
INSTITUTE OF APPLIED INFORMATICS,
RESEARCH GROUP APPLICATION ENGINEERING,
ALPEN-ADRIA-UNIVERSITÄT KLAGENFURT
USING LINGUISTIC KNOWLEDGE FOR FINE-TUNING ONTOLOGIES...

GÜNTHER FLEIDL
INSTITUTE OF APPLIED INFORMATICS,
RESEARCH GROUP APPLICATION ENGINEERING,
ALPEN-ADRIA-UNIVERSITÄT KLAGENFURT

CHRISTIAN KOP
INSTITUTE OF APPLIED INFORMATICS,
RESEARCH GROUP APPLICATION ENGINEERING,
ALPEN-ADRIA-UNIVERSITÄT KLAGENFURT

MYKOLA BAZHENOV
COMPUTER AIDED MANAGEMENT SYSTEMS DEPARTMENT,
NATIONAL TECHNICAL UNIVERSITY
“KHARKOV POLYTECHNIC INSTITUTE”