Biomedical Terminology Mapper for UML projects

Julien C. Thibault, MS, Lewis Frey, PhD
Department of Biomedical Informatics, University of Utah, Salt Lake City, UT

Abstract
As the biomedical community collects and generates more and more data, the need to describe these datasets for exchange and interoperability becomes crucial. This paper presents a mapping algorithm that can help developers expose local implementations described with UML through standard terminologies. The input UML class or attribute name is first normalized and tokenized, then lookups in a UMLS-based dictionary are performed. For the evaluation of the algorithm 142 UML projects were extracted from caGrid and automatically mapped to National Cancer Institute (NCI) terminology concepts. Resulting mappings at the UML class and attribute levels were compared to the manually curated annotations provided in caGrid. Results are promising and show that this type of algorithm could speed-up the tedious process of mapping local implementations to standard biomedical terminologies.

Introduction
With the emergence of next generation sequencing and the push for meaningful use and data exchange between biomedical institutions, biomedical researchers are tasked with managing heterogeneous and complex data. In order to allow interoperability between institutions, and to facilitate translation of basic research to the bedside, researchers that own the data need to use standard terminologies to describe their data sources. Mapping a local implementation to a standard terminology can be a very labor-intensive task. This paper introduces a system that automatically maps a local system implementation to a standard biomedical terminology. The local implementation needs to be described through the Unified Modeling Language (UML), a common approach for defining and documenting complex software systems and a standard de facto. UML classes and attributes are mapped to standard concepts extracted from the Unified Medical Language System (UMLS)\(^1\) Metathesaurus one of the largest sources for controlled biomedical vocabulary. For the evaluation of the system, existing UML projects were extracted from caBIG’s Cancer Data Standards Resource (caDSR)\(^2\). caDSR is an endpoint for developers who want to share the metadata from their projects with the community. Projects are published as UML models and annotated with Common Data Elements (CDE) at the class and attribute levels to provide meaning to the specific implementations. Each CDE is mapped to one or several concepts from the NCI thesaurus. Therefore caDSR provides a good reference to evaluate a system that attempts to map UML classes and their attributes to standard concepts. In a prior work, Kunz et al.\(^3\) showed that using UML class and attribute names and simple algorithms, CDEs could be automatically mapped to UML projects with a good success rate. This study presents a similar system aiming at standard concepts rather than CDEs. As mentioned by Kunz et al., CDEs and UML entity names are somewhat coupled because of the nature of the CDE creation process: when a new UML project is published into caDSR, a new CDE will be created by the caDSR curators if a UML entity happens to have no mapping within the current list of CDEs. Working with an existing terminology that is independent from the data model (UML representation) should reduce this type of bias.

Figure 1. Example of a UML class (gov.nih.nci.cabio.domain.PhysicalLocation) and its mappings at the class and attribute (id, chromosomalStartPosition, chromosomalEndPosition) levels to NCI concepts as defined in caDSR.
Methods

For this study we are interested in mapping UML classes and attributes – i.e. entities – to NCI concepts solely based on their names. A pipeline was created to extract UML entities from caDSR then tokenize and normalize their names, before looking up the resulting terms in a pre-built NCI concept index. The returned matches are compared to actual NCI annotations for the UML entities available in caDSR (semantic metadata). Examples of such mappings are presented in Figure 1. From this we can see that the entity name is not always sufficient to infer concept mappings (e.g. mapping ADXPUNID). Nevertheless, in most cases, we assume that the name by itself bears enough information to find these mappings (or most of them) automatically. These annotations were manually curated by the UML project owners, and we assume they can be used to build a reference standard of NCI concept mappings. This dataset was used as a baseline to evaluate the accuracy of our system.

UML data: UML classes and associated attributes were pulled through the caDSR client API. In total 142 UML projects are available in caDSR. All associated UML classes were selected (6582). After removal of duplicates and classes with missing references to UML projects, 6486 classes remained. In total 14,049 NCI concept mappings for these classes were found in caDSR, which gave an average of 2.17 concepts mappings per UML class. 10,000 UML attributes were extracted from caDSR, out of 33,448. After removal of duplicates and attributes with missing references to UML projects, 9,978 attributes remained. For these remaining attributes 16,419 mappings were available, for an average of 1.65 NCI concept mappings per UML attribute.

Figure 2. Mapping UML entities to NCI concepts and building the reference standard. The first step is to build an index of NCI concepts from the UMLS Metathesaurus (blue pipeline). The UML entities (classes and attributes) are then extracted from caDSR. NCI concept annotations for these entities represent our gold standard. This set is compared to NCI concepts that are mapped to the normalized names of the UML entities.

NCI concept term index: An index of NCI concept terms was built to perform lookups of normalized phrases derived from UML class and attribute names (see Figure 2). NCI concepts were indexed on the terms (i.e. synonyms), and include features from the MRCONSO and MRDEF tables of the UMLS Metathesaurus database. The index was built using Lucene⁴, a high-performance text search engine library. Our lookup module is based on the Lucene v3.0 Java API, and currently uses an index built from the 2009 UMLS Metathesaurus¹. For this study the index was limited to include only the NCI terminology, as it is the standard terminology in caGrid. 173,713 NCI concepts were extracted from the UMLS Metathesaurus, then normalized, before being indexed in the Lucene dictionary. The normalization process included the removal of stop words (e.g. “is”, “are”, “of”, “the”, “a”) and the mutation of tokens to their canonical form using the Lexical Variant Generation (LVG)⁵ tool. The canonical form (or lemma) of a word is the “root” of this word. For example the canonical form of a noun is always its singular form (e.g. “allergies” becomes “allergy”) and the infinitive for a verb (e.g. “prescribed” and “prescribing” becomes “prescribe”).

258
Tokenization and normalization of UML entity names: Mappings are based on the match between NCI concepts and the phrase derived from the names of the UML attribute or class. A pre-processing step is therefore necessary to transform the name of these entities into a phrase. This tokenization is done through different rules that take into consideration capitalization patterns and numbers. Words using “ca” as an abbreviation of cancer (e.g., “caGrid”, “caDSR”, and “caCore”) are explicitly preserved as single tokens. Applying the previous rules to these words would lead to a wrong tokenization with the first word being “ca”. This exclusion is important as these words are fairly frequent in class or attribute names. Along with these rules, special characters (e.g. underscore “_”, dash “-”) are also replaced by white spaces, and the acronym “ID” is expanded as “identifier”. Once the phrase is tokenized, it is normalized using the same process as for the NCI concepts terms. Table 1 presents a few examples of these transformations.

Table 1. Examples of UML entity names and their expected derived phrase.

| UML entity name | Expected phrase after transformation |
|-----------------|--------------------------------------|
| SkyCase_header  | sky case header                      |
| caGridModel     | cagrid model                         |
| Coordinates     | coordinate                            |
| Experiment2DLiquidChromatography2ndSetup | experiment 2d liquid chromatography 2nd setup |

Mapping UML data to NCI: The resulting phrase is then looked up in the Lucene index. Lookups in Lucene were setup to return a maximum of 20 hits. The Lucene scoring is based on the match between the tokens of the input phrase and the tokens of the index terms, with more weight on the terms that occur less often in the index. The hits returned by the Lucene lookup were then filtered by removing matched concepts that would have at least one word that would not match with a word of the name phrase. For a single string there might be multiple exact matches in the NCI concept index. To reduce the amount of duplicate matches, preferred NCI terms are chosen over non-preferred terms. Preferred NCI terms are flagged in the TTY column of the MRCONSO table in the UMLS Metathesaurus database.

Results

NCI concepts automatically mapped by the system were compared to the caDSR annotations through precision (P), recall (R), and F-measure (with β set to 1) calculations, defined as:

\[
R = \frac{TP}{TP + FN} \quad P = \frac{TP}{TP + FP} \quad F = \frac{(\beta^2 + 1) \times R \times P}{(\beta^2 \times P) + R}
\]

where TP, FP, and FN are the number of true positives, false positives, and false negatives. Here true positives for a UML entity are NCI concepts that have been automatically mapped to by the system and that are also present in the gold standard (caDSR). False positives are concepts that were found by the system but not part of the caDSR annotations. False negatives are NCI concepts that were missed by the system.

The results of the system evaluation are presented in Table 2. Measures were performed for two different versions of the system: one that uses normalization for lookups (and concept indexing) as described previously, and another one that preserves the original form of the tokens in the NCI index and the phrases derived from the UML entity names. Measures for the attribute and class mappings were applied at the project level (average by project) and at the entity level (average by class or by attribute). Distributions of precision, recall, and F1-measure per entity and per project are also presented graphically in Figure 3 for the “original” (non-normalized) version.

Results show that the system performs in general better with mappings of UML attributes than UML classes. Also the “normalized” version of the system does not perform as well as the “original” system. Distributions of the measures at the entity level tend to have a higher rate towards a perfect score (median close to 100%). When looking at averages per project, the measures follow a distribution close to normal, centered around 50%.

Discussion

An analysis of the false negatives shows that most misses are due to annotations that required specific knowledge about the project. For example, the UML attribute “controlActRoot” in the ISO2109 project is annotated in caDSR with the NCI concepts "Event" (C25499), "Identifier" (C25364) and "Base" (C48055), which were all missed by the
system. A human, with no prior knowledge about the project would probably not be able to create these mappings either just based on the entity name.

The ISO2109 project actually has the lowest average recall for class mappings (6%), with only 45 true positives and 706 false negatives. Many class names in this project are capitalized acronyms or abbreviations (e.g. PQTIME, TELPERSON, TELURL, UUID), for which no tokenization rule apply. This project deals with the integration of multiple standards for health terminologies, which probably does not represent a typical UML project developed by cancer researchers. Exclusion of projects like ISO2109 for the evaluation step might lead to better and more realistic results.

Table 2. System evaluation for UML attributes and class mappings. The table presents results for both versions of the system (with and without normalization). Both mean and median are reported as in most cases results do not follow a normal distribution.

| Mapping type                  | System version | Precision       | Recall          | F1-measure       |
|-------------------------------|----------------|-----------------|-----------------|-----------------|
|                               |                | Normalized      | Original        | Normalized      | Original        |
| UML class (per class)         | Mean           | 60.34 %         | 61.96 %         | 59.20 %         | 64.32 %         | 52.51 %         | 56.42 %         |
|                               | Median         | 50.00 %         | 60.00 %         | 60.00 %         | 66.67 %         | 50.00 %         | 57.14 %         |
| UML class (per project)       | Mean           | 49.60 %         | 52.04 %         | 57.98 %         | 62.71 %         | 52.42 %         | 55.72 %         |
|                               | Median         | 50.00 %         | 51.61 %         | 57.74 %         | 63.69 %         | 52.53 %         | 56.77 %         |
| UML attribute (per attribute) | Mean           | 72.10 %         | 71.90 %         | 64.85 %         | 77.10 %         | 59.85 %         | 70.00 %         |
|                               | Median         | 100.00 %        | 100.00 %        | 100.00 %        | 100.00 %        | 66.67 %         | 80.00 %         |
| UML attribute (per project)   | Mean           | 59.10 %         | 62.26 %         | 59.10 %         | 70.71 %         | 57.52 %         | 65.64 %         |
|                               | Median         | 59.20 %         | 64.10 %         | 59.01 %         | 71.93 %         | 59.04 %         | 66.67 %         |

On the other hand many false positives are due to NCI concepts that are slightly different semantically but have similar synonyms. For example, the UML attribute “conceptCode” (in UML class gov.nih.cagrid.evs.service.EVSHistoryRecordsSearchParams) was mapped by the system to NCI concepts C43516 (“NCI concept code”) and C45972 (“NCI Metathesaurus concept code”), which have the same synonym “concept code”. Once again, more knowledge about the context of the entity would help select the right concept mapping.

Another source of false positives and negatives is the fuzzy definition of what an annotation should be. For concepts that semantically and syntactically match the entity name it makes sense to define them as correct annotations. For partial matches it becomes more complicated. For example “conceptCode” could be mapped to other concepts that have synonyms such as “code” or “concept”. In the current implementation, the system would map the phrase “concept code” to 3 concept codes: “concept code”, “code”, and “concept”. This leads to two false positives. While we could add a filtering step to remove partial matches if they overlap with a full match, annotations in caDSR did not seem to be consistent regarding this problem. Further investigation is needed to determine the need for this extra filtering step. In order to create more context, one could use the definitions of the UML entities instead of or in addition to their name. Definitions are richer semantically but they are also harder to use for mapping because of the higher potential for false positives. For example the description of the UML class “gov.nih.nci.cabio.domain-Protein” is “A polymer composed of amino acids based on a template encoded in DNA”. Assuming that we restrict lookups to noun phrases, the phrases “polymer”, “amino acids”, and “DNA” could be mapped to NCI concepts. In most cases the only relevant concepts would be “polymer” and “amino acids”. The term “DNA” defines an association with “Protein” but does not belong to a strict definition of protein. Such extra information could lead to wrong mappings (false positives). Finally, although NCI already provides a large source of concepts, the number of synonyms might be limited. The list of NCI terms could be automatically enhanced by adding new UMLS terms that are synonyms of NCI concepts (i.e. that have the same CUIs but a different term).

The difference of performance between the “normalized” and “original” versions of the system was somewhat surprising. One would expect a better recall when performing lookups with phrases that have been normalized. Here the “original” version outperforms the “normalized” system on both recall and precision. One reason for this might
be the loss of granularity created by the normalization process, associated to the limit defined for the number of dictionary matches. While a dictionary lookup will match more terms when normalized, the final filtered terms returned by the module are not guaranteed to represent distinct concepts, and might not include similar terms with equal matching scores that would represent the needed concepts. The filtering algorithm should be redesigned for a future implementation so that the number of maximum hits is defined at the concept level and not at the term (synonym) level. One could also use the normalized lookup for a pre-filtering phase then use the original terms for a more accurate selection of the final concepts.

Despite the simplicity of the algorithm used here the results are promising and are consistent with Kunz et al.\(^3\) in that domain specific models have better performance. The system is right at least half of the time on the mappings it offers, doing better with UML attributes than UML classes. Although the system could not replace human annotations it could greatly speed-up the process by pre-annotating UML projects or offering a list of potential related concepts to the developers.

---

**References**

1. Bodenreider O. The Unified Medical Language System (UMLS): integrating biomedical terminology. Nucleic Acids Research. 2004;32(Database Issue):267-270.
2. Oster S, Langella S, Hastings S, Ervin D, Madduri R, Phillips J, Kurc T, Siebenlist F, Covitz P, Shanbhag K, et al. caGrid 1.0: An Enterprise Grid Infrastructure for Biomedical Research. J Am Med Inform Assoc. 2008;15(2):138-149.
3. Kunz I, Ming-Chin L, Frey L. Metadata mapping and reuse in caBIG. 1st Summit of Translational Bioinformatics, BMC Bioinformatics 2009;10(Suppl 2):S4.
4. Deng-peng Z, Kang-lin X. Lucene Search Engine. Jisuanji Gongcheng / Computer Engineering. 2007;33(18).
5. McCray AT, Srinivasan S, Browne AC. Lexical methods for managing variation in biomedical terminologies. Proc Annu Symp Comput Appl Med Care. 1994; 235-239.