Method and tool support for classifying software languages with Wikipedia

Ralf Lämmel, Dominik Mosen, and Andrei Varanovich

University of Koblenz-Landau, Software Languages Team

Abstract. Wikipedia provides useful input for efforts on mining taxonomies or ontologies in specific domains. In particular, Wikipedia's categories serve classification. In this paper, we describe a method and a corresponding tool, WikiTax, for exploring Wikipedia's category graph with the objective of supporting the development of a classification of software languages. The category graph is extracted level by level. The extracted graph is visualized in a tree-like manner. Category attributes (i.e., metrics) such as depth are visualized. Irrelevant edges and nodes may be excluded. These exclusions are documented while using a manageable and well-defined set of 'exclusion types' as comments.

1 Introduction

Ever since 2008, the calls for papers for the *Software Language Engineering* (SLE) conference[1] have contained slightly different, more implicit or more explicit definitions of the term 'software language'. Other community material contains yet other definition attempts; see, for example, the IEEE TSE special section on SLE in 2009 [8]. At SLEBOK 2012 (i.e., an SLE 2012 satellite event dedicated to the the SL(E) body of knowledge), the attendees were also getting into the issue of what exactly a software language is.

A *classification* of software languages is a useful (if not necessary) pillar of a definition of 'software language'. Such classification is the topic of the present paper. One branch of software languages appears to be well understood. That is, *programming languages* are obviously *software languages* and they may be classified in terms of criteria and concepts as organized, for example, in textbooks on programming languages, programming paradigms, and programming language theory such as [13,16]. There is also scholarly (dated) work on the classification of programming languages [1,6]. Actually quite a few sets of criteria or concepts exist for programming languages; there is no obvious contender; there is no comprehensive classification. Several classes of languages (other than programming languages) have been classified in scholarly work, e.g., model transformation languages [5], business rule modeling languages [17], visual languages [3,4,11], and architecture description languages [12]. The ultimate taxonomy of software languages should subsume and integrate existing, fragmented classifications in a

[1] `http://planet-sl.org/`

transparent manner. The *101companies* project[2] hosts efforts targeted at such a taxonomy, but the results are of limited use and quality so far.

In this paper, we try to inform the apparent classification challenge for software languages by means of exploring Wikipedia. Obviously, Wikipedia contains substantial amounts of taxonomy-like (if not ontology-like) information—also for software languages (without though embracing the actual term, at the time of writing). For instance, there are hierarchically organized categories such as *Computer languages*, *Programming languages*, and *Programming language classification* that seem to apply; yet other categories may be relevant. Accordingly, we describe a method and a corresponding tool, WikiTax, for exploring Wikipedia's category graph. Exploration is supported in a manner such that a domain expert can reduce the category graph so that a classification emerges. The overall approach is not specific to software languages, but we apply it to software languages throughout the paper.

Contribution We do not claim to have converged on a good candidate taxonomy for software languages. Rather we contribute procedural, tool-supported elements of a method towards development of the ultimate taxonomy. The resulting tool, WikiTax, is a rather simple graph exploration tool, which however includes a few domain-specific features not available in more generic functionality for searching and exploring Wikipedia's category graph.

Road-map §2 describes the overall exploration approach and sketches corresponding tool support as implemented by WikiTax. §3 explores Wikipedia categories related to software languages. §4 concludes the paper. The source code of WikiTax, a comprehensive manual, and all data covered in this paper are available online.[3]

2 Exploring Wikipedia with WikiTax

Wikipedia's category graph Wikipedia uses several means of organizing its information: plain links giving rise to an article graph, designated article lists, portals meant to introduce users to key topics, info-boxes for semantic ('typed') data, and categories giving rise to a category graph for the classification of articles. When it comes to taxonomy mining, the category graph is particularly relevant; the graph is accessible, for example, through the MediaWiki API, which is the access path chosen by WikiTax.

Graph extraction Initially, WikiTax is pointed to a root category (level 0) for extraction. Iteratively, subcategories and pages (in fact, page titles) can be extracted level by level or exhaustively. Exhaustive extraction may take minutes to hours depending on the root category. The Wikipedia category graph contains many surprising edges, which would easily imply inclusion of large, arguably irrelevant subgraphs. Thus, extraction is controllable.

[2] `http://101companies.org/`
[3] `https://github.com/dmosen/wiki-analysis`

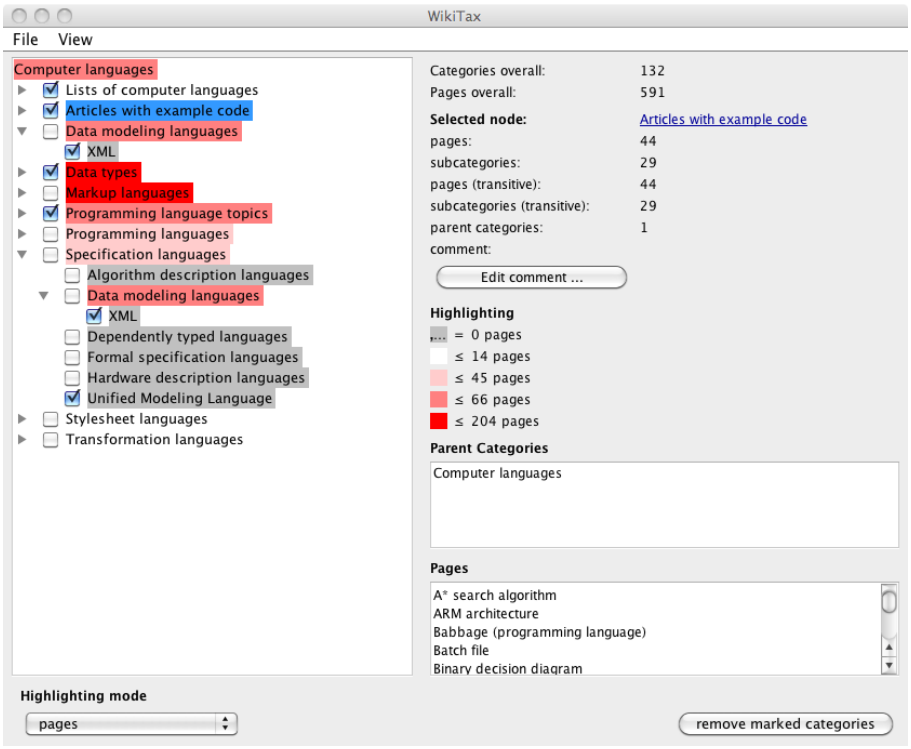

Fig. 1. Exploration of level 1 and 2 subcategories of *Computer languages*.

Graph reduction WikiTax supports reduction of the graph—both during (level-by-level) extraction and post extraction. Reduction boils down to the exclusion of nodes, i.e., categories. (In fact, we may also remove individual edges, given that a category may have multiple parent categories.) A category would be removed, if domain knowledge suggests that the category at hand does not serve the intended kind of classification, e.g., classification of software languages in our case. When exclusion is performed during extraction, then the excluded nodes (edges) are ignored during subsequent extraction steps. When exclusion is performed post extraction, then nodes (edges) are only blacklisted, without actually reducing the graph. In this manner, exclusion decisions can be revisited.

WikiTax's visualization Figure 1 shows the WikiTax exploration view after the extraction of levels 1 and 2 starting from the category *Computer languages*. Some edges are marked for exclusion. (Exclusion would be confirmed with the 'removal' button.) The marked categories are to be excluded because domain knowledge suggests that these categories do not serve language classification in a conceptual manner. Highlighting is applied to the categories according to the metric of immediate member pages. In the figure, the category *Articles with example code* is selected so that extra data is shown in the panel on the right, e.g., member pages. All categories and pages are clickable to navigate to Wikipedia.

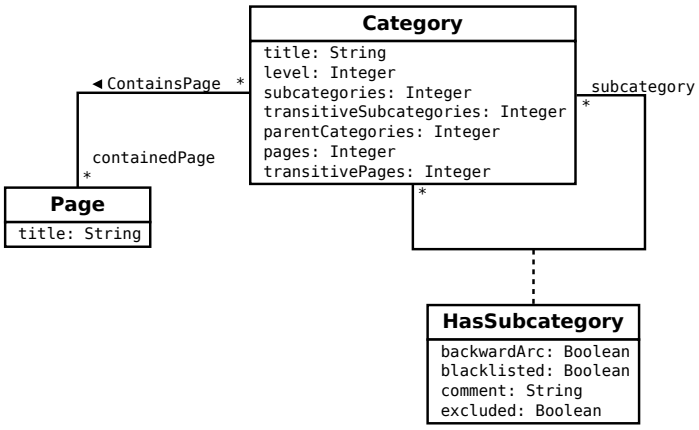

Fig. 2. Metamodel of the WikiTax category graph.

WikiTax's metamodel WikiTax operates on an enhanced category graph; see the metamodel in Figure 2. Thus, each category associates with contained pages and subcategories. The subcategory associations are attributed to keep track of metadata as follows:

backwardArc Marker for cyclic edges in the category graph.
blacklisted Marker for categories blacklisted past extraction.
excluded Marker for categories excluded during reduction.
comment Label ('reason for exclusion') to be associated with the edge.

Categories are associated with measures as follows:

level The level 0, 1, 2, ... of the category in the graph with the root at level 0.
subcategories The number of immediate subcategories.
transitiveSubcategories The number of all subcategories.
pages The number of immediately contained pages.
transitivePages The number of all pages in this category.

The implementation of WikiTax uses the Java-based JGraLab library[4] for the representation of (annotated) graphs with JSON as an export format.

Exclusion types A methodologically important aspect of graph reduction is that reasons for category exclusion are not just simply documented by a comment, but a manageable, well-defined set of exclusion types is to be developed over time. For instance, the category *Unified Modeling Language* could be said to be of an exclusion type 'Singleton classifier' to mean that this category, by design, is primarily concerned with a single language, i.e., UML in this case; the other members or subcategories of the category are concerned with UML concepts, tools, and other related artifacts. §3 lists several more exclusion types. The aggregation and use of exclusion types captures domain knowledge and insight into Wikipedia's category graph in a transparent manner.

[4] `https://github.com/jgralab`

3 Explorative study

In this study, we examine some Wikipedia categories with two objectives: a) to retrieve some candidate classifiers of an emerging taxonomy of software languages; b) to get some experience with Wikipedia's approach to classification and related issues of style and consistency.[5]

Designation of a root Wikipedia's classification hierarchies are complex and thus, it is not straightforward to determine a root for exploration unambiguously. However, we have established by an ad-hoc search that the category *Computer languages* may be a suitable root: its intended coverage may be similar to what the SL(E) community has in mind for the notion of software languages.

Figure 1 showed all the immediate (i.e., level 1) subcategories of the category *Computer languages*. Several of these immediate subcategories are excluded because they are not directly concerned with the *classification* of languages: *Lists of computer languages*, *Articles with example code*, *Data types*, and *Programming language topics*. One of the remaining immediate subcategories is the category *Programming languages*. We found another major classifier for programming languages, namely *Programming language classification*, which is reachable through the excluded category *Programming language topics*.

Level-by-level extraction We decided to extract another level to obtain a graph of manageable size. Again, we excluded several categories, if they did not meet our objective of language classification. As a result, we obtained the categories shown in Figure 3. This is a pretty manageable set of language classifiers. It happens that they all end on "... languages" except for two subcategories of *Markup languages* which end on "... formats". In contrast, most of the excluded categories (see below) do not end on "... languages" . We take this to provide a hint at the different classification styles of Wikipedia.

Exclusion types In order to obtain the reduced result of Figure 3, we had to exclude 29 categories. This may seem like a small number, but it is clear that yet more categories must be excluded once deeper levels are explored. We used these 29 exclusions to develop a small set of exclusion types for the study; see Figure 4 for the list of excluded categories with the associated exclusion type:

Alternative classifier The category classifies software languages in a manner that is not related to software concepts. For instance, the category *Academic programming languages* describes itself as being concerned with languages that are "influential in computer science and programming language theory".

Deviating classifier The category does not actually classify software languages. It rather classifies something else. For instance, category *Articles with example code* describes itself as being concerned with "articles which include reference implementations of algorithms".

Singleton classifier The category is effectively concerned with a single software language for which it serves as a container of related entities such as technologies or standards. For instance, category *Cascading Style Sheets* contains pages on all kinds of topics related to the CSS language.

[5] All Wikipedia data for this study and this paper was retrieved 7-18 June 2013.

Category	Subcategories
Data modeling languages	–
Markup languages	*Declarative markup languages*, *GIS file formats*, *Knowledge representation languages*, *Lightweight markup languages*, *Mathematical markup languages*, *Musical markup languages*, *Page description markup languages*, *Playlist markup languages*, *User interface markup languages*, *Vector graphics markup languages*, *Web syndication formats*, *XML markup languages*
Programming languages	*.NET programming languages*, *Agent-based programming languages*, *Agent-oriented programming languages*, *Concatenative programming languages*, *Concurrent programming languages*, *Data-structured programming languages*, *Declarative programming languages*, *Dependently typed languages*, *Domain-specific programming languages*, *Dynamic programming languages*, *Extensible syntax programming languages*, *Formula manipulation languages*, *Function-level languages*, *Functional languages*, *High Integrity Programming Language*, *High-level programming languages*, *ICL programming languages*, *Intensional programming languages*, *Low-level programming languages*, *Multi-paradigm programming languages*, *Nondeterministic programming languages*, *Object-based programming languages*, *Pattern matching programming languages*, *Procedural programming languages*, *Process termination functions*, *Prototype-based programming languages*, *Reactive programming languages*, *Secure programming languages*, *Set theoretic programming languages*, *Statically typed programming languages*, *Synchronous programming languages*, *Term-rewriting programming languages*, *Text-oriented programming languages*, *Tree programming languages*, *Visual programming languages*, *XML-based programming languages*
Specification languages	*Algorithm description languages*, *Dependently typed languages*, *Formal specification languages*, *Hardware description languages*
Stylesheet languages	–
Transformation languages	*Macro programming languages*

Fig. 3. Reduced subcategory lists for subcategories of *Computer languages*.

List classifier The category collects lists or categories of lists (rather than plain categories) of software languages. For instance, category *Lists of computer languages* has *Lists of programming languages* as a subcategory, which in turn contains pages for some lists of languages, such as the *List of BASIC dialects*.

Maintenance classifier The category is used by the Wikipedia authors to capture some information related to the maintenance of pages or categories. For instance, the category *Uncategorized programming languages* describes itself as serving categories or pages "which need to be classified under more specific categories". Also: "This category may be empty occasionally or even most of the time."

An observation regarding Wikipedia style The resulting classification of Figure 3 with the remaining level-1 and level-2 subcategories is of a manageable size. We may review the classification and observe some of its characteristics in

Category	Exclusion type
Academic programming languages	Alternative classifier
Articles with example code	Deviating classifier
Cascading Style Sheets	Singleton classifier
Data types	Deviating classifier
Discontinued programming languages	Alternative classifier
DocBook	Singleton classifier
Esoteric programming languages	Alternative classifier
Experimental programming languages	Alternative classifier
HTML	Singleton classifier
JSON	Singleton classifier
Lists of computer languages	List classifier
Lists of programming languages	List classifier
Markup language comparisons	Deviating classifier
Markup language stubs	Maintenance classifier
Non-English-based programming languages	Alternative classifier
Programming language families	Deviating classifier
Programming language standards	Deviating classifier
Programming language topics	Deviating classifier
Programming languages by creation date	Alternative classifier
Programming languages conferences	Deviating classifier
Software by programming language	Deviating classifier
SyncML	Singleton classifier
TeX	Singleton classifier
Text Encoding Initiative	Singleton classifier
Troff	Singleton classifier
Uncategorized programming languages	Maintenance classifier
Unified Modeling Language	Singleton classifier
Wikipedia categories named after programming languages	Deviating classifier
XML	Singleton classifier

Fig. 4. Exclusion types for levels 1 and 2 of *Computer languages*; this list is produced by the WikiTax tool based on metadata (comments) entered by us interactively.

this manner. During the study, we realized, for example, an asymmetry between 'query' versus 'transformation'. That is, there is a category *Transformation languages* at level 1, but there is apparently no category for 'query languages', not even at level 2. Let us inspect the page for *SQL*, which is an obvious query language. It turns out that *SQL* is a member of various categories including a category *Query languages* which in turn is a subcategory of various categories including the category *Domain-specific programming languages* which occurred in Figure 3. Let us compare this classification scheme with the one of *XSLT*, which is an obvious transformation language: it is a member of the categories *Transformation languages*, *Declarative programming languages*, *Functional languages*, *Markup languages*, *XML-based programming languages*, and yet other categories that may count as 'alternative classifiers'. However, *XSLT* (unlike *SQL*) is not a member of the category *Domain-specific programming languages*.

WikiTax is helpful in making such observations regarding consistency (or lack thereof) of classification on Wikipedia.

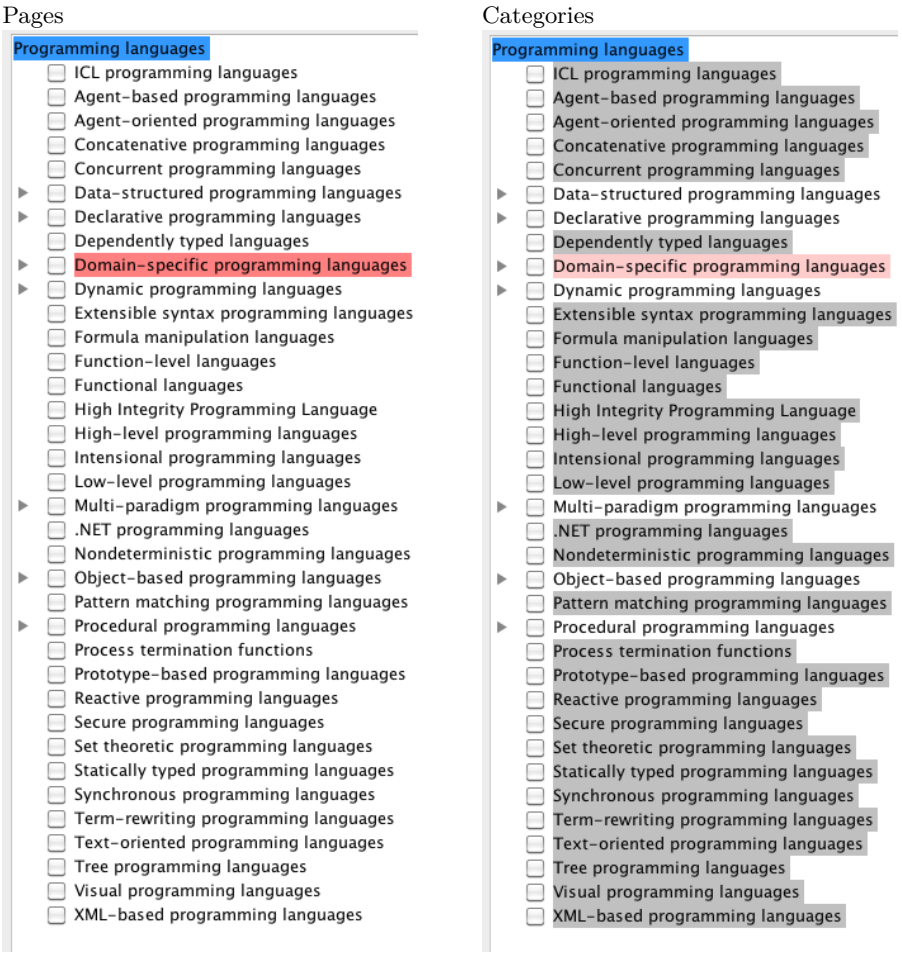

Fig. 5. Metrics-based views on *Programming languages* graph.

Programming languages: all levels According to Figure 3, the subcategory of *Computer languages* with by far the most subcategories is *Programming languages*. Thus, we embarked on a more comprehensive exploration of category *Programming languages*:

Initially, we extracted 423 categories over 8 levels with 7515 pages. The automatic extraction took several minutes. We performed exclusion in two steps. First, we (re-) excluded those direct subcategories that already appeared in Figure 4. After such initial pruning, 288 categories with 6671 pages remained. We completed reduction at all levels of the category graph. This process required about 2 hours of manual work to determine what categories to remove and for what reason. This effort is intrinsically manual; it requires domain knowledge and involves consultation of the relevant and additional Wikipedia pages. Ultimately, 79 categories over 4 levels with 1560 pages remained. Figure 5 visualizes the reduced taxonomy for two different metrics supported by WikiTax.

On the left, the metric for the *number of transitive member pages* is applied for visualization. No category is grayed out, which means that there is no category without members. Most of the categories are shown in a plain font, which means that they all carry members, but less than 25 % of the total members in the category *Programming languages* (which has 1560 member pages). There is actually one heavyweight: category *Domain-specific programming languages* carries 976 members, which is more than 50 % of all members; this status is expressed by highlighting the category.

On the right, the metric for the number of transitive subcategories is applied for visualization. Most subcategories of *Programming languages* do not have any subcategories; thus, they are grayed out. 7 out of 36 level-1 categories carry subcategories. 6 out of these 7 categories carry only very few subcategories (less than 5). Category *Domain-specific programming languages* carries 18 subcategories, which is more than 25 % of all subcategories; this status is expressed by highlighting the category.

4 Conclusion

Any domain with large data to explore ('large' in terms of what the user needs to understand) may benefit from interactive exploration possibly with editing or annotation; see tools for ontologies [2], graphs [9], semantic data [7], software bugs [10], API usage [15]. In this paper, we described an approach to the exploration of Wikipedia's category graph so that candidate taxonomies can be extracted from the graph. We were specifically interested in understanding Wikipedia's classification of software languages. To this end, we developed a domain-specific exploration tool, WikiTax, which supports level-by-level graph extraction, metrics-based graph visualization as well as transparent and revisable graph reduction. Such designated exploration support is missing in more generic tools for searching or exploring the category graph.

The described method of graph reduction is deliberately interactive and relies on domain knowledge for transparent exclusion decisions, as opposed to any means of automated ontology extraction / generation [18,19]. (Without such validation, there is little hope that the resulting taxonomy would be readily meaningful.) An important conceptual contribution is our proposal to document exclusion decisions with (comments for) exclusion types, thereby making reduction more systematic and transparent. This interactive approach can be contrasted with related work on taxonomy or ontology mining, where categories are classified and additional relationships are inferred automatically, e.g., by analyzing the structure of compound category names [14].

We contend that the described approach provides the initial core of a method for actually developing a taxonomy for software languages (and possibly other taxonomies) on the grounds of Wikipedia. Collaborative work and further improved tool support are needed to actually arrive at a comprehensive taxonomy. We imagine that we need powerful refactoring operations on the category graph to facilitate taxonomy extraction and enforcement of consistent style. The exploration of the category graph could also be supported by additional forms of

visualization, e.g., for understanding the overlap of categories. Also, we need to generally better understand (perhaps based on an automated analysis) the different classifier styles used by Wikipedia.

References

1. Babenko, L.P., Rogach, V.D., Yushchenko, E.L.: Comparison and classification of programming languages. Cybernetics 11, 271–278 (1975)
2. Baskaya, F., Kekäläinen, J., Järvelin, K.: A tool for ontology-editing and ontology-based information exploration. In: Proc. of ESAIR 2010. pp. 29–30. ACM (2010)
3. Bottoni, P., Grau, A.: A suite of metamodels as a basis for a classification of visual languages. In: Proc. of VL/HCC 2004. pp. 83–90. IEEE Computer Society (2004)
4. Burnett, M.M., Baker, M.J.: A classification system for visual programming languages. J. Vis. Lang. Comput. 5(3), 287–300 (1994)
5. Czarnecki, K., Helsen, S.: Feature-based survey of model transformation approaches. IBM Systems Journal 45(3), 621–646 (2006)
6. Doyle, J.R., Stretch, D.D.: The classification of programming languages by usage. International Journal of Man-Machine Studies 26(3), 343–360 (1987)
7. Dumas, B., Broché, T., Hoste, L., Signer, B.: ViDaX: an interactive semantic data visualisation and exploration tool. In: Proc. of AVI 2012. pp. 757–760. ACM (2012)
8. Favre, J.M., Gasevic, D., Lämmel, R., Winter, A.: Guest editors' introduction to the special section on software language engineering. IEEE Trans. Software Eng. 35(6), 737–741 (2009)
9. Haun, S., Nürnberger, A., Kötter, T., Thiel, K., Berthold, M.R.: CET: A tool for creative exploration of graphs. In: Proc. of ECML/PKDD (3) 2010. LNCS, vol. 6323, pp. 587–590. Springer (2010)
10. Hora, A., Anquetil, N., Ducasse, S., Bhatti, M.U., Couto, C., Valente, M.T., Martins, J.: Bug Maps: A tool for the visual exploration and analysis of bugs. In: Proc. of CSMR 2012. pp. 523–526. IEEE (2012)
11. Marriott, K., Meyer, B.: On the classification of visual languages by grammar hierarchies. J. Vis. Lang. Comput. 8(4), 375–402 (1997)
12. Medvidovic, N., Taylor, R.N.: A classification and comparison framework for software architecture description languages. IEEE Trans. Software Eng. 26(1), 70–93 (2000)
13. Mosses, P.D.: Action Semantics. Cambridge University Press (1992)
14. Nastase, V., Strube, M.: Decoding Wikipedia categories for knowledge acquisition. In: Proc. of AAAI 2008. pp. 1219–1224. AAAI Press (2008)
15. Roover, C.D., Lämmel, R., Pek, E.: Multi-dimensional exploration of API usage. In: Proc. of ICPC 2013. IEEE (2013), to appear. 10 pages.
16. Sebesta, R.W.: Concepts of Programming Languages. Addison-Wesley (2012), 10th edition
17. Skalna, I., Gawel, B.: Model driven architecture and classification of business rules modelling languages. In: Proc. of FedCSIS 2012. pp. 949–952 (2012)
18. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A large ontology from Wikipedia and WordNet. J. Web Sem. 6(3), 203–217 (2008)
19. Wu, F., Weld, D.S.: Automatically refining the Wikipedia infobox ontology. In: Proc. of WWW 2008. pp. 635–644. ACM (2008)