Distributed Vocabulary Development with Version Control Systems
Lavdim Halilaj, Steffen Lohmann, Christian Mader, Sören Auer
Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS)

Abstract. Vocabularies are increasingly being developed on platforms for hosting version-controlled repositories, such as GitHub. However, these platforms lack important features that have proven useful in vocabulary development. We present VoCol, an integrated environment that supports the development of vocabularies using Version Control Systems. VoCol is based on a fundamental model of vocabulary development, consisting of the three core activities modeling, population, and testing. It uses a loose coupling of validation, querying, analytics, visualization, and documentation generation components on top of a standard Git repository. All components, including the version-controlled repository, can be configured and replaced with little effort to cater for various use cases.

Introduction

Vocabulary development is currently a major bottleneck for the wide realization of semantic data integration. It requires a significant investment, which is difficult to make by a single person or organization. Identifying the terms and concepts by finding a consensus among the involved stakeholders and defining a shared vocabulary is an effective approach to tackle this problem. However, the main challenge for vocabulary engineers is to work collaboratively on a shared objective while avoiding misunderstandings, uncertainty, and ambiguity.

On the other hand, Version Control Systems (VCS), such as Subversion (SVN) or Git, are becoming increasingly popular for vocabulary development. In our previous work, we proposed Git4Voc[2], a set of best practices which transfer concepts of VCSs to vocabulary development, on the example of Git. Several aspects of vocabulary development, in particular with regard to revision management, access control, and some governance issues are already well covered by Git-based version control, especially if developers follow the proposed best practices.

Many of the current vocabulary development activities take place on Repository Hosting Platforms like GitHub, GitLab, or BitBucket. In addition to mere version-controlled repositories, these platforms provide features such as change tracking, issue tracking, wikis, and notifications. However, the platforms lack important features that have proven useful in vocabulary development. In particular, they do not provide an integrated environment typically found in tools such as WebProtégé, VocBench, or PoolParty.

Approach

We designed VoCol as a holistic approach for realizing a full-featured vocabulary development environment centered around version control systems (VCSs). It supports a fundamental
round-trip model of vocabulary development, consisting of the three core activities \textit{modeling}, \textit{population}, and \textit{testing}, as illustrated in Figure 1.

In the spirit of test-driven software engineering, VoCol allows to formulate queries, which represent competency questions for \textit{testing} the expressivity and applicability of a vocabulary a priori. \textit{Modeling} comprises the analysis and conceptualization of the domain and the specification of the vocabulary terms, including classes, properties, and the relationships between them VoCol integrates a number of techniques facilitating the conceptual work, such as automatically generated documentations and visualizations, providing different views on the vocabulary as well as an evolution timeline supporting traceability. The next activity is typically \textit{population} which includes the definition of the mappings files such as R2RML in line with the defined classes and properties as well as querying the external data sources.

The governance of distributed vocabulary development is supported by the access control as well as branching and merging mechanisms of the underlying VCS system. As a result, VoCol bridges between the conceptual development of vocabularies and the operational execution in a concrete IT landscape. The implementation of VoCol is based on a loose coupling, leveraging the webhook mechanism combined with tools and techniques focusing on particular aspects of vocabulary development. By providing Vagrant and Docker containers bundling all tools and encapsulating dependencies, VoCol is easily deployable or even usable as-a-service in conjunction with arbitrary VCS installations.

![Figure 1. Round-trip vocabulary development supported by VoCol](image)

\textbf{Architecture}

The integrated VoCol system architecture is illustrated in Figure 2. It is based on the principles of Component Based Software Development (CBSD) [1], which promote the reuse of (off-the-shelf) components to develop large-scale systems. Each of the VoCol components is exchangeable and can be replaced by alternatives.
Version Control System - is an essential component for supporting the loosely coupled collaboration between vocabulary engineers without risking to lose data. It is responsible for the management of vocabulary changes, such as change capturing and propagation: by capturing and storing the changes, various revisions of the vocabulary are created. To ensure a consistent development process, any change should be propagated to all other contributors. In addition, conflicts inevitably arise in an environment where multiple contributors are working simultaneously and changing vocabulary terms. The VCS ensures conflict resolution and allows integration of conflicting changes in an effective and easy way.

On the other hand, the Repository hosting platforms, such as GitHub, GitLab, and Bitbucket act as repository storage where the vocabulary files are saved and accessed. The integrated access control mechanisms authenticate users with the right permissions. Furthermore, contributors can discuss vocabulary terms or definitions and suggest changes or new terms using the issue tracking functionality of the platform.

Syntax Validation - ensures that the latest revision in the VCS is always syntactically correct. Syntax validation could be realized at different stages of the overall workflow. However, with the aim to keep the requirements on the client side at a minimum level, we realized a syntax validation service on the server. As a result, syntactically incorrect commits are rejected and a detailed error report is provided.
Unique Serialization - Different editors may produce different serializations of the vocabulary files. To overcome this problem, VoCol creates a unique serialization of vocabulary files before changes are pushed to the remote repository, thus preventing false-positive merge conflicts.

Documentation Generation - produces an HTML representation of the vocabulary. This permits contributors to easily navigate through the entire vocabulary and provides a concise but still detailed overview (cf. generated documentation for the ChargingPoint term in Figure 3).

Visualization Generation - By visualizing classes, properties and their connections, users are provided with a coherent view of the vocabulary. In addition, this service enables to explore multilingual vocabulary terms (cf. Figure 4).

Querying Service - VoCol integrates a SPARQL endpoint synchronized with the latest version of the vocabulary. During testing, queries derived from competency questions can be used to verify whether the vocabulary fulfills the domain requirements. All defined queries are stored in the repository and pre-loaded in the query user interface.

Inconsistency and Constraint Checking - After the changes have been pushed to the remote repository, checks for semantic inconsistencies and constraint violation are performed and their results are compiled in a detailed report.

Machine Accessibility - Using the content negotiation mechanism and dereferenceable URIs, VoCol delivers various machine-comprehensible representations. By specifying the content type in the HTTP header along with the resource URI, the vocabulary can be accessed by different software agents compliant with the linked data principles.

Evolution Tracking - detects semantic differences between vocabulary versions. It shows which classes and properties have been added, removed or modified, enabling users to see the vocabulary evolution over time (cf. Figure 5).
**Configuration Service** - provides a Graphical User Interface to facilitate the configuration of VoCol. The VoCol administrator can choose between various alternative tools for syntax validation and documentation generation. Furthermore, other services can be activated or deactivated (cf. VoCol Configuration Page in Figure 6).

**Monitoring Service** - Repository hosting platforms expose most of their functionality via web service APIs. Any change pushed to the repository is delivered as a payload event to a VoCol monitoring service, which automatically invokes services for documentation generation, visualization, evolution tracking, etc.

**Conclusions**

VoCol is an integrated environment for the distributed development of vocabularies based on version control systems. Tasks such as content negotiation, documentation and visualization generation, as well as evolution tracking are performed in a fully automated way. In addition, a querying service, synchronized with the latest version of the vocabulary, enables users to execute SPARQL queries. The VoCol environment is easily expandable with other tools to provide additional functionalities. For the future, we plan to provide VoCol as a service where users can simply subscribe their repositories and benefit from all functionalities.

**References**
