Feature	VB3	VB2
SKOS Editing	YES. - Support for both SKOS and SKOS-XL labels - Support for SKOS reified notes by means of Custom Forms	YES, but with the following limitations: - only SKOS-XL labels - no support for collections
SKOS-XL Editing	YES	YES
OWL Editing	YES (OWL2)	NO
OntoLex Editing	YES	NO
RDF Editing Capabilities	YES, Unrestricted	Governed by Tabs, scoped on skos:concepts and skosxl:Labels
SPARQL Support	YES With syntax highlight and completion fed by the vocabularies imported in the managed dataset Support for storeable queries/updates	YES, with syntax highlight
Interaction with external triple stores	Support for RDF4J, GraphDB and, potentially, with other stores compliant with RDF4J client API. Compliancy with RDF4J's sail mechanism is required for store-side facilities such as history and validation. Improved wrt VB2, it is now possible to directly create and configure the repository, with configuration options for RDF4J and GraphDB (both free and SE sails)	Support for RDF4J, GraphDB (and, potentially, with other stores compliant with RDF4J client API) Limited to connection to existing repositories
Access Control and User Management	YES	YES
Role Based Access Control (RBAC)	YES: a much more flexible system than in VB2, it is possible to completely customize roles and capabilities, and even create new ones very easily	YES
History	YES; stored as triples in a triple store; content as action metadata *and* actual list of changed triples	YES; stored on a separated SQL database; content limited to action metadata
Action Validation	YES: with validation enabled, all changed triples are reified and stored in a staging repository; normal triples are stored in a staging graph for temporary visualization. Upon validation, these are moved to the working graph (and in case of history, saved as reified changed triples on the history repository)	YES (upon rejection of action *A*, a a counter-action to *A* is fired)
Publication Workflow	YES; streamlined wrt VB2, now it is interwoven with the validation mechanism. Any triples added/removed through validation are stored in a dedicated store and *previewed* in a dedicated graph of the main data store. Thus, resources with their definition (i.e. triples with rdf:type) still under validation are assumed to be *new resources* while if their definition is being deleted (under approval of validation) these are	YES

	considered to be *proposed for deletion*, without needing a dedicated status property for that. The only explicit status is related to deprecated resources, adopting the standard *owl:deprecated* property. Resources with their owl:deprecated status being uder validation are considered as *proposed for deprecation*.	
Provenance	YES; everything is stored in RDF, using widespread vocabularies for provenance (e.g. PROV-O)	YES (on the relational DB)
Customizable Forms	YES: possibility to define Custom Forms when adding new instances of certain classes or new values of certain properties	NO
Integrated Constraint Validation	YES: several ICV checks and fixing actions are available	NO
Alignment Support	YES. Improved wrt VB2 as it provides semi-automatic alignment	YES
Alignment Validation	YES: an integrated system for analyzing mapping documents following INRIA's Alignment Ontology and extending it for evaluating alignments and/or validating them in order to	NO
Metadata Vocabularies	YES: Support for VoID, LIME, DCAT, DCAT-AP, ADMS. More can be added through a dedicated extension point.	NO
Metadata Registry	A metadata registry inside VocBench caches all of the metadata information gathered from visited datasets on the Web. The information can be complemented by users. The metadata registry adopts all of the most reknown metadata vocabularies (VoID, LIME, DCAT, DCAT-AP, ADMS) and combines them into a complete registry	NO
LOD Navigation	YES: when mention of an external resource is present in the edited dataset, VB3 will access the web to recover the data and present it uniformly in the same interface used for the local one. The information stored in the metadata registry will be exploited to use the best resource (e.g. SPARQL endpoint over simple http dereferenciation) to recover the data and to recover it in the best way for presentation (e.g. by knowing the adopted lexicalization model thanks to LIME metadata)	NO
Versioning	YES: users can create snapshots of a repository and tag them with a version identifier (and other metadata, such as the time of creation of the snapshot). It is possible to time-travel through different versions of the edited repository	NO
RDF Transformers	YES Extension point for connecting RDF Transformation Engines. A few implementations are already available. The most general one is a SPARQL Update transformer, that allows to virtually apply any transformation to the data being imported/exported. Can be used for dataset import and export, and for processing the results of SPARQL Graph queries (i.e. DESCRIBE and CONSTRUCT) before they are exported. Chain of configured exporters can be stored, with different scopes (user, project, system and user-on-project)	NO Only a few refactoring utilities and an export option for SKOSXL-->SKOS
Advanced Search	YES	YES
Customizable Search Support	YES, powered by user-defined SPARQL queries that can be stored as custom searches	NO
Collaboration System Support	YES: an extension point allows to connect different collaboration systems. Extension for JIRA available.	NO
Spreadsheet Import	Sheet2rdf included in the system. It allows to import spreadsheet's content (in various formats, from Excel to various open formats) and to triplify it by using a transformation language: PEARL. A convention-over-configuration approach allows users to get most of the transformation already computed by the system, basing on heuristics and spreadsheet patterns.	NO