VCD Components Versioning Strategy

1. Log of Changes

Version	Date of Change	Changed By	Summary of Change
0.1	2011/08/11	Daniel Reiser	Initial draft version

2. Contents

3. Introduction

This document describes the versioning strategy for handling releases and versions of the different artefacts and software components developed within work package 2.

The definition of a versioning and release management strategy aims at providing a structured and comprehensive approach to manage the development and release cycles of the different VCD artefacts and software components. This should also take interdependencies between artefacts and components into consideration in order to highlight critical relations between two objects, i.e. a new release version of one component having an impact on a specific release version of another.

As there are various artefacts and components with several interdependencies, the following chapter provides a brief description and overview about the different objects that are relevant for this versioning strategy. Afterwards, their interdependencies are outlined. Finally, a release plan shall define the different development cycles per component considering interdependencies that may affect specific release dates of a certain version.

4. General rules for versioning components

4.1. Version numbers

Version numbers are used to express the current state of development of each component. Each version number consists of three parts following the format X1.X2.X3:

- X1: Indicating a major / official release
- X2: Indicating an intermediate release
- X3: Indicating minor updates due to small changes during work in progress phase

Parts X1 and X2 refer to releases that have a global impact on all other components whereas Part X3 refers to minor updates made during development with only a local impact. With respect to the planning of releases, at least release dates and milestones for X1 and X2 releases have to be defined that in turn has an impact on other components.

4.2. Component interrelations

Interrelations between components are of type “affects” / “affected by” describing the impact of releases of one component on releases of another component. They follow a strict meaning of “Release of component A triggers creation of new release of component B”.

The spread sheet itself is subject to on-going revisions and updates if necessary. In result, the list of components and their interdependencies may be adjusted and should be flexible enough. If any missing interdependencies are discovered, they should be included directly.

4.3. Work in progress and release planning

Each component follows its own workflow including development, revisions and updates (work in progress) as well as incremental version releases. Current work in progress of each component is influenced by other component releases that have a direct impact. Necessary changes should hence be included to the list of issues making up the work in progress of each component.

As a general guideline, the request for change methodology defined for the work in progress and releases of the VCD document model should be used. Put into a more general context, the Issue lifecycle that can be applied to each component is depicted in Figure 1.

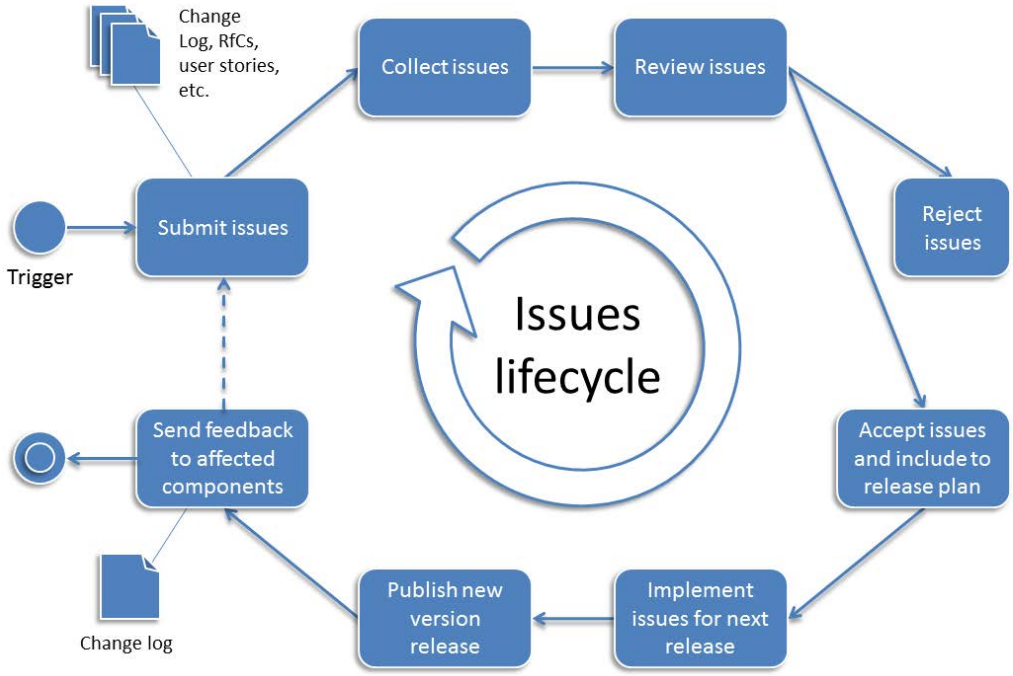

Figure 1: General issue lifecycle model

Starting point for this lifecycle is a trigger resulting from feedback, tests or revisions of this component or releases of other components. Impacts on a component are submitted in the form of issues, which can be anything from change logs to requests for changes, bug-reports, user stories, etc. As several issues may be submitted at different points in time, they are collected in a list of currently submitted issues. For this purpose, if suitable, means similar to the RfC spread sheet already in used for the document model and schema workflow might be used for other components. During the work-in-progress cycles of each component, the issues will be reviewed and rejected or accepted afterwards. Accepted issues are included to the specific release plan of each component, i.e. for each issue it has to be defined in which version it will take effect or will be implemented. Depending on this planning the issues already planned for the next release (i.e. already existing and pending issues) or recently submitted issues to be released with the next version are implemented. The new version release will be published at the given release date. This implies “freezing” the current version and publishing it at a certain location (e.g. BSCW, Mercurial, etc.). With each release, the issue lifecycle of all affected component has to be triggered. To provide an appropriate input to

another component's issue lifecycle, each release should be accompanied by a change log containing the relevant changes in the current release compared to the previous release.

Although this general model is applied to each component, specific details vary from component to component. Therefore, the next chapter provides an overview about all components and specifies these details more precisely.

5. Versioning of components

The excel file delivered together with this document provides a comprehensive overview of the components and interdependencies that have to be considered within the versioning strategy.

It provides information about the current release number, the planned version number and date of the next release as well as web-links to the remote locations where resources for work in progress as well as release packages are stored. In combination with the previously described issue lifecycle model, the subsequent sections will provide further details of specific lifecycle topics such as triggers, work in progress artefacts, release artefacts as well as impacts to and from other components.

5.1. VCD document model

Work in progress: CurrentVCDSchema
Releases: VCDSchemaRelease.

The document model spread sheet defines a hierarchical document model of all information elements that can be used in a VCD document instance.

Development of the document model follows the request for change methodology:

- RfCs are collected in the rfc-spreadsheet. These RfCs can be generated during software testing, document model and schema revisions, releases of concept data base (in particular definitions and labels as well as business rules) or any other impacts affecting changes of the document model.
- After prioritization each RfC is assigned to a specific version number and a targeted implementation date at which this RfC will be implemented in the document model.
- The release planning and definition of release dates is done ad-hoc depending on the number of submitted/accepted RfCs and planned delivery dates of the software components.

After the RfCs have been implemented in the document model, the generation of corresponding XML schema definition files is triggered. Input to the XML schema definition creation is a log of changes (can be extracted from the RfC list) accompanied by the document model spread sheet itself.

5.2. XML schema definition (xsd)

Releases: VCDSchema, Change log

The XML schema definitions represent the technical implementation of the document model spread sheet. It consists of the XML schema definition files (XSD) that define the XML syntax of the VCD data elements.

The release of new XML schema definitions is triggered by releases of the VCD document model. Whenever a new VCD document model release is published, all changes will be transposed into new schema releases.

Whenever new releases of the XML schema definitions are available, a new user story for each software component triggering the implementation of the schema update has to be created.

5.3. VCD container structure specification

Work in progress: VCD Container Structure
Releases: VCD Container Structure Release

The VCD Container structure specification defines the physical structure of a VCD Container. It represents the overall physical implementation of the VCD data format and document model and uses the XML schema definitions and VCD document model specification.

Its specification is input to the implementation guidelines defined in the VCD concept data base, hence each release will trigger

If a new version is released, its specifications have to be transformed in the implementation guidelines.

5.4. Code lists definitions

Work in progress: Code Lists definitions

The code list definition spread sheet defines the content of all code lists used in the VCD system. Type and number of code lists depend on the code list elements defined in the VCD document model as well as specific implementation guidelines.

The spread sheet contains its editing workflow (creating, editing, revision) and status tracking methodology which is used to track the work in progress for each code list definition. Whenever the revision of a code list is finished, the corresponding genericode file can be created.

The code list definitions come without any release files as the actual releases are represented by new versions of the corresponding genericode files which are derived from the spread sheet.

5.5. Genericodes

Work in progress: Code Lists definitions
Releases: Genericodes

Once a single code list definition is finalized, i.e. editing and revision of the spread sheet table is finished, the genericode file is produced by transforming the code list table into an xml-based genericode file (gc).

Whenever new genericode files are released, a corresponding user story for each software component will be created to implement the new genericode versions.

5.6. VCD ontologies

The ontologies define the rule sets used for the criteria-evidence mapping. They are used in the EVS to perform the criteria-evidence mapping.

Updates to the ontology files will be released to the EVS.

Kommentar [MS1]: Updates in the ontologies will be implemented in the VCD ontology management tool and released to the EVS.

Kommentar [DR2R1]: Does this mean, that adjustments to the ontology management tool are required? I thought the OMS provides the GUI to edit the ontologies, hence updates of the ontologies are done by using the OMS and does not have any further impacts on the current OMS version?

5.7. VCD concept data base

Work in progress: VCD concept database

Releases: VCD concept database releases

The VCD concept data base is an overall definition file to specify semantics, usage and implementation of different VCD related concepts. It consists of three sections:

- The VCD concept definitions and GUI labels provide a list of all semantic definitions as well as labels (English and translations) to be used when presenting a concept to end users (e.g. on graphical user interfaces (GUI)).
- The business rules define all rules that govern the creation and usage of the various VCD related concepts.
- The implementation guidelines consist of a list of guidelines for the implementation of all VCD related concepts.

The concept database excel spread sheet contains the work in progress and is updated whenever changes are made. The versioning and release methodology follows and incremental approach: Fixed release dates are defined in advance (maybe adjusted ad-hoc) taking into account the editing status of the three parts and interdependencies with other components:

- Whenever changes or releases of the “affected by” components are made, the concept data base and its relevant parts have to be updated accordingly. This means that the data base content is influenced by VCD document model, the GUI design as well as results of software tests.
- If a specific release date is reached, the concept data base and corresponding artifacts (e.g. language files, business rules, implementation guidelines) are “frozen” and will be copied to the release directory. Language files will be generated from the definitions and labels table.

New versions of the concept data base will be implemented in the affected components as follows:

- **Definitions and labels:**

  The values for definitions and their translations are transformed into language files.

  Definitions and labels are included into the document model, hence corresponding change requests have to be defined.

- **Language files:**

  A user story will be created for each software component for the implementation of GUI labels and translations.

- **Business rules:**

  The impact of certain changes in the business rules table will be transformed directly in the implementation guidelines during the editing phase and hence will already be part of each release of the concept data base.

  In the same way, changes affecting the VCD document model will directly be reported as requests for changes in the RfC spread sheet.

  The validation rules will be adjusted according to the changes made in the business rules table.

- **Implementation guidelines:**

  The table for the implementation guidelines contains columns to track the implementation of each concept in the different software components. A user story will represent the general implementation of all the guidelines of a specific release. Details about the scoped concepts and their implementation status for the specific release will be provided in the corresponding cells of the table.

  Impacts on the GUI design have to be implemented in the corresponding GUI design templates similar to the implementation in software components. Corresponding information will also be given in the relevant columns.

5.8. GUI design

Releases: Common GUI

The common GUI design implemented in all VCD software components.

The GUI design is treated similar to the software components: Changes, improvements or missing GUI components (“bugs”) are reported as issues in JIRA. Sources for such issues may be testing of software components or editing of business rules and implementations guidelines in the concept data base. According to the scrum based approach, the issues related to the common GUI have to be collected, prioritized and selected to be implemented in subsequent sprints with given deadlines.

On the other side, the common GUI design can also affect the definitions and labels as well as implementation guidelines of the concept data base. Whenever adjustments to these affected components are necessary or discovered, they have to be communicated to the concept data base editor in order to include them.

5.9. Software components

Work in progress: VCD Designer, VCD Builder, VCD Viewer, VCD Signer

The software components are the main building blocks of the technical VCD System. They implement several other components and hence are affected by corresponding releases:

- Genericode files
- Concept data base definitions and labels
- Concept data base implementation guidelines
- Bug fixes due to bug reports created during software testing

Development of the various software components follows a scrum based implementation approach. User stories are collected and prioritized. In each implementation sprint, a sub-set of user stories is selected and implemented. Each sprint leads to a new sub-release (indicated by the second version number) of the component which is tested afterwards.

Test results are input to the list of user stories and issues and are reported as bug reports or improvement issues. They will be fixed and implemented in subsequent sprints.

Test results can also affect the VCD document model and the concept data base: If certain issues requiring changes or adjustments of the VCD document model are discovered, they will be reported following the RfC methodology of the VCD document model and VCD schema. Changes affecting the concept data base are reported ad-hoc using the regular net meetings to decide on the inclusion/changing of elements in the concept data base.

Tracking the implementation status of certain issues realized in the corresponding tools, i.e. the concept data base for tracking the implantation of code lists, implementation guidelines and definitions/labels whereas JIRA is used to track bug-fixes and implementation of improvements.

5.10. Validation rules

Work in progress: Validation

The validation rules are based on the business rules and define the constraints which are used to validate given VCD document instances. The technical implementation is either expressed directly in the XSD files, Schematron rules or additional information constraints.

The creation of validation rules is solely influenced by the creation of business rules, which are transformed into validation rules. The release plan of the validation rules can be defined

independently of other components and has to be in line with the overall planning of the system and conformance testing methodology, as the validation rules influences the creation of VCD profiles, which in turn directly affect the creation of VCD instance test systems.

The business rules table contains necessary columns to track the transformation in corresponding validation rules. Hence it is possible to incrementally update the validation rules by transforming additional business rules.

In the same way, the validation rules spread sheet provides necessary columns to keep track of the implementation status of certain validation rules in the corresponding VCD profiles.

5.11. VCD profile test systems

Releases: Test system

The VCD profiles represent the implementation of the validation rules together with specific XSD files for the different VCD document types that may occur in the VCD system.

As the validation rules is the only influencing component, new versions of the validation rules trigger the creation of new versions of the VCD profiles. As described in the previous section, the validation rules file contains relevant means to track the status of transforming each validation rule into corresponding VCD profiles. Depending on the prioritization and release planning of the validation rules, clusters of these rules are collected and transformed into VCD profiles within defined iterations (ad-hoc transformations should also be possible depending on the priority and importance of certain validation rules).

A new version of the VCD profiles in turn triggers the creation of new test systems for each of the profiles. Those can be transformed immediately after a new profile is released.

Test systems are executable software applications, each configured for a specific VCD profile. Hence they are directly generated for the given VCD profiles once a new version is released. These test systems will be used to validate given VCD documents against the validation rules implemented in a certain VCD profile.

5.12. VCD test instances

Work in progess: VCD instances

The VCD test instances represent specific instances of the underlying XML schema definitions with concrete data. They are used as data input during the software testing.

New test instances will be created whenever new versions of the XML schema definitions are released so that appropriate test data exists to be used during software testing as well as input/output validation for software components.