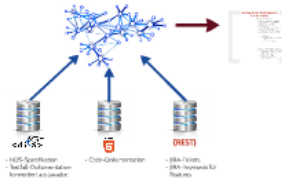


Entwicklerdokumentation in den Griff bekommen

Prozesse und Infrastruktur verbessern

- Umstieg auf moderne Toolsets: Entwicklung mit JIRA und Git
- ✓ Gleichzeitige Arbeit an Code & Dokumentation wird kein Problem
 - ✓ Alle relevanten Informationen in JIRA, auch Bestand der Dokumentation
 - ✓ Automatische Build bei Dokumentenänderungen und Code
 - ✓ Erklärungen in Code ergänzen Branchenpraxis
 - ✓ Ein Abstrakt Syntax Notation (ASN) für Dokumentation möglich

Content-Delivery-Portal einführen



Fazit



- Prozesse sind genauso wichtig wie Tools
- Dokumentation ist Teil der Produktentwicklung
- Standardisierungen wie DITA ermöglichen flexible Tool-Auswahl
- Systematische Anreicherung der Dokumentenbestände und regelmäßige Updates und Auswertungen
- Systematisches Content-Delivery-Portal bietet Zusatznutzen

Fragen?



write.parson@parson-cuepc.com

Wertföhrliche Informationen

- Was ist die Bedeutung von DITA?
- Was ist die Bedeutung von DITA?
- Was ist die Bedeutung von DITA?
- Was ist die Bedeutung von DITA?
- Was ist die Bedeutung von DITA?
- Was ist die Bedeutung von DITA?
- Was ist die Bedeutung von DITA?
- Was ist die Bedeutung von DITA?

Das Projekt



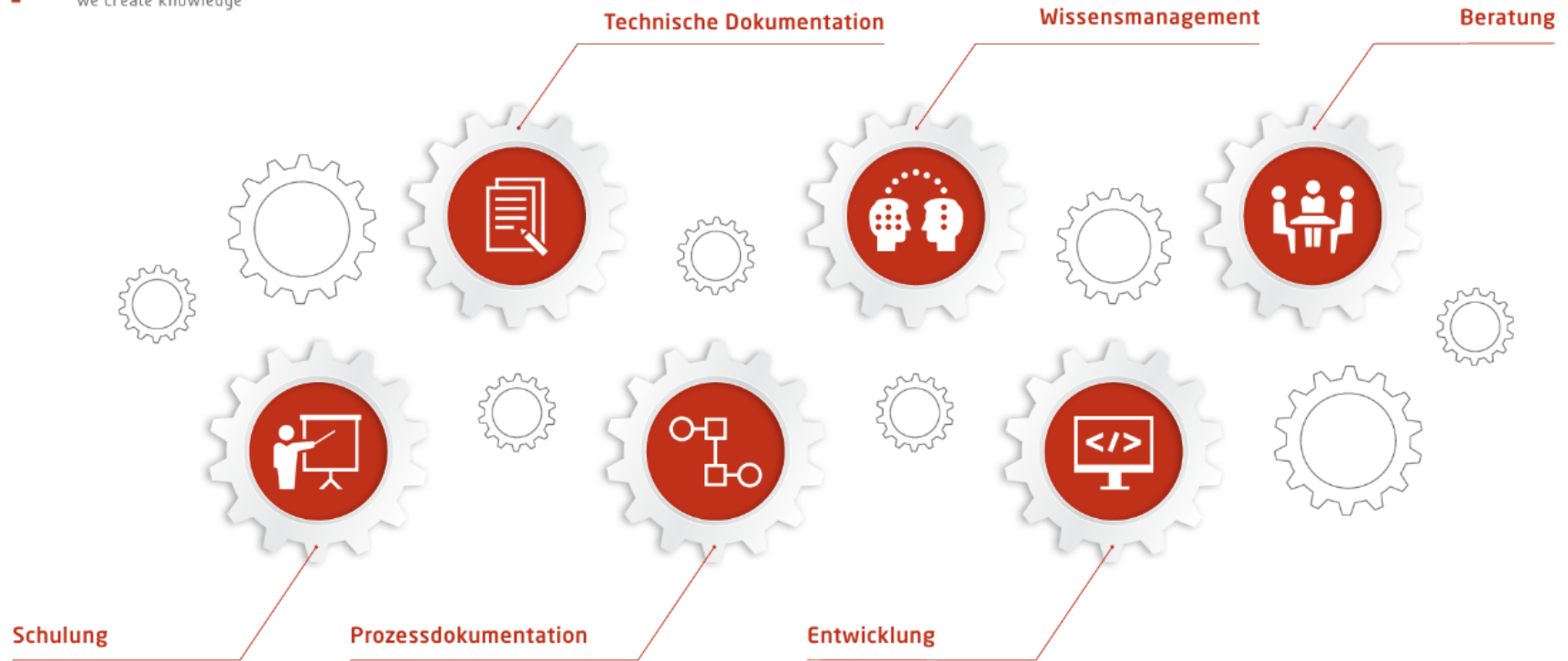
https://www.dita-standards.org/



Dokumentationsquellen verbessern



- Kontinuierliche Verbesserung der Qualität
- ✓ Tiefere Modularisierung über Screens
 - ✓ Vorkonzeptionierung für ADP-Verfahren mit Variablen und Testen-basierten Bedingungen
 - ✓ Validierung der dokumentierten Code-Strukturen über Skripte
 - ✓ Pflege und Erweiterung der Rollen-Taxonomie



Das Projekt



Navigation Data Standard

<http://www.nds-association.org/>

Ausgangssituation



- Dokumentation ist wesentliche Repräsentation des Standards
- Kontinuierliche Entwicklung mit flexiblen Releasezyklen
- Änderung im Standard lösen Entwicklungsaufwand bei Herstellern von Navigationslösungen aus
- Technologische Basis: DITA für Dokumentation und Testfälle, Code-Dokumentation in HTML
- Publikation auf Website und in Offline-Formaten

Probleme



- Entwicklung und Dokumentation zeitversetzt
- Nachverfolgbarkeit von Änderungen schwierig
- Publikation zeitraubend
- Website nicht skalierbar
- Verknüpfung von Dokumentation zu Code-Dokumentation und Testfällen hard-codierte Annahmen, keine echten Relationen

Ziele



- Code-Doku, Testfälle und Dokumentation gleichzeitig publizieren
- Nachverfolgbarkeit von Änderungen verbessern
- Prozesse automatisieren, Publikation beschleunigen
- Relationen zwischen Informationsobjekten abbilden und sichtbar machen
- Nutzbarkeit der Website verbessern
- Dokumentation enger mit Entwicklung verzahnen
- Umstellung auf feature-basierte Entwicklung mitgestalten

Ausgangssituation



- Dokumentation ist wesentliche Repräsentation des Standards
- Kontinuierliche Entwicklung mit flexiblen Releasezyklen
- Änderung im Standard lösen Entwicklungsaufwand bei Herstellern von Navigationslösungen aus
- Technologische Basis: DITA für Dokumentation und Testfälle, Code-Dokumentation in HTML
- Publikation auf Website und in Offline-Formaten

Probleme



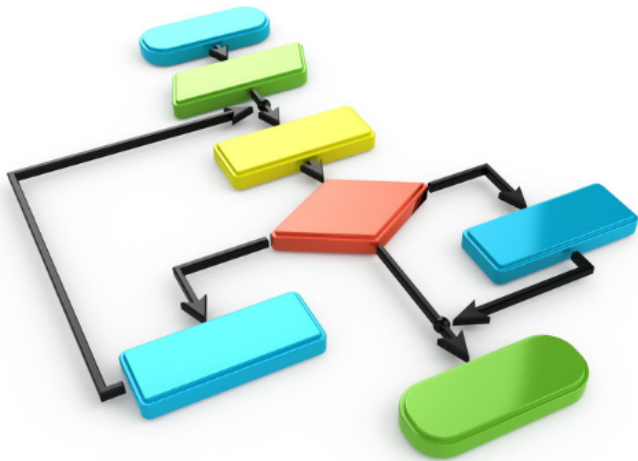
- Entwicklung und Dokumentation zeitversetzt
- Nachverfolgbarkeit von Änderungen schwierig
- Publikation zeitraubend
- Website nicht skalierbar
- Verknüpfung von Dokumentation zu Code-Dokumentation und Testfällen hard-codierte Annahmen, keine echten Relationen

Ziele




- Code-Doku, Testfälle und Dokumentation gleichzeitig publizieren
- Nachverfolgbarkeit von Änderungen verbessern
- Prozesse automatisieren, Publikation beschleunigen
- Relationen zwischen Informationsobjekten abbilden und sichtbar machen
- Nutzbarkeit der Website verbessern
- Dokumentation enger mit Entwicklung verzahnen
- Umstellung auf feature-basierte Entwicklung mitgestalten

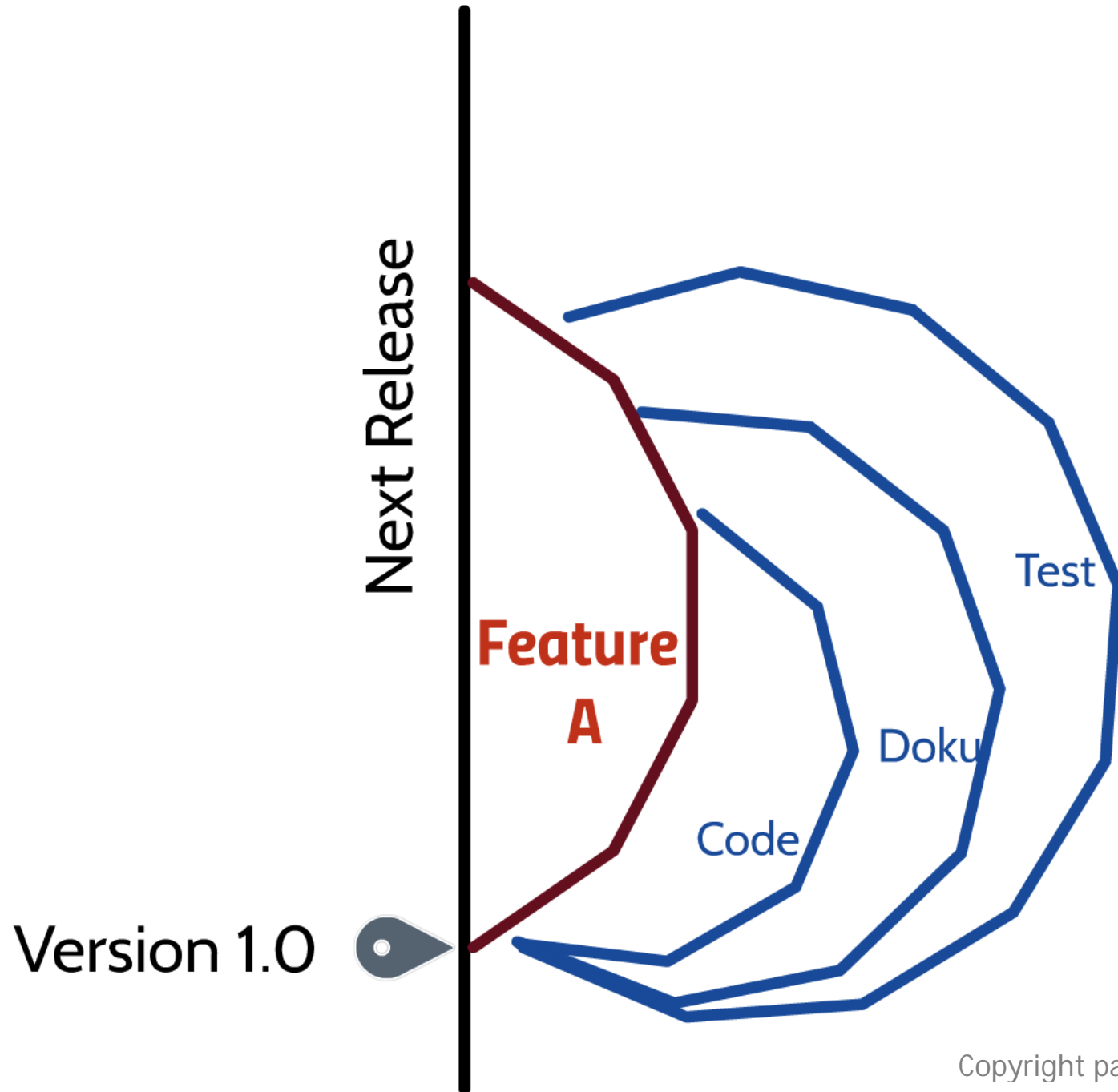
Prozesse und Infrastruktur verbessern



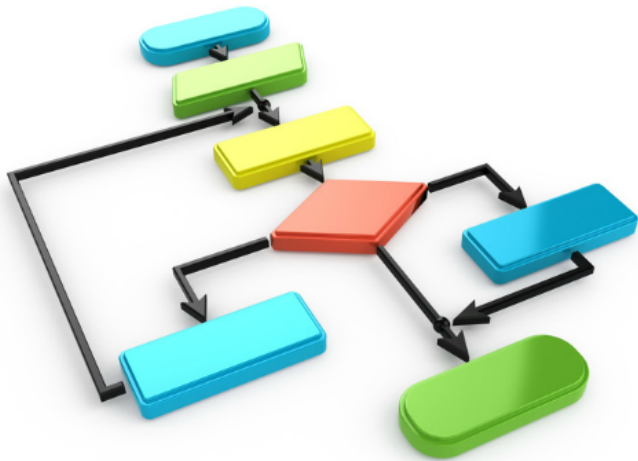
Umstieg auf feature-basierte Entwicklung mit JIRA und Git

- ✓ Gleichrangige Arbeitsabläufe für Entwicklung, Dokumentation und Test in JIRA
- ✓ Alle Arbeitsabläufe komplett in JIRA, auch Review der Dokumentation
- ✓ Automatischer Build der Dokumentation mit Code
- ✓ Dokumentation in Git, eigener Branche pro Feature
- ✓ Für Abnahme eines Features muss die Dokumentation fertig sein. 


Branching-Strategie in Git



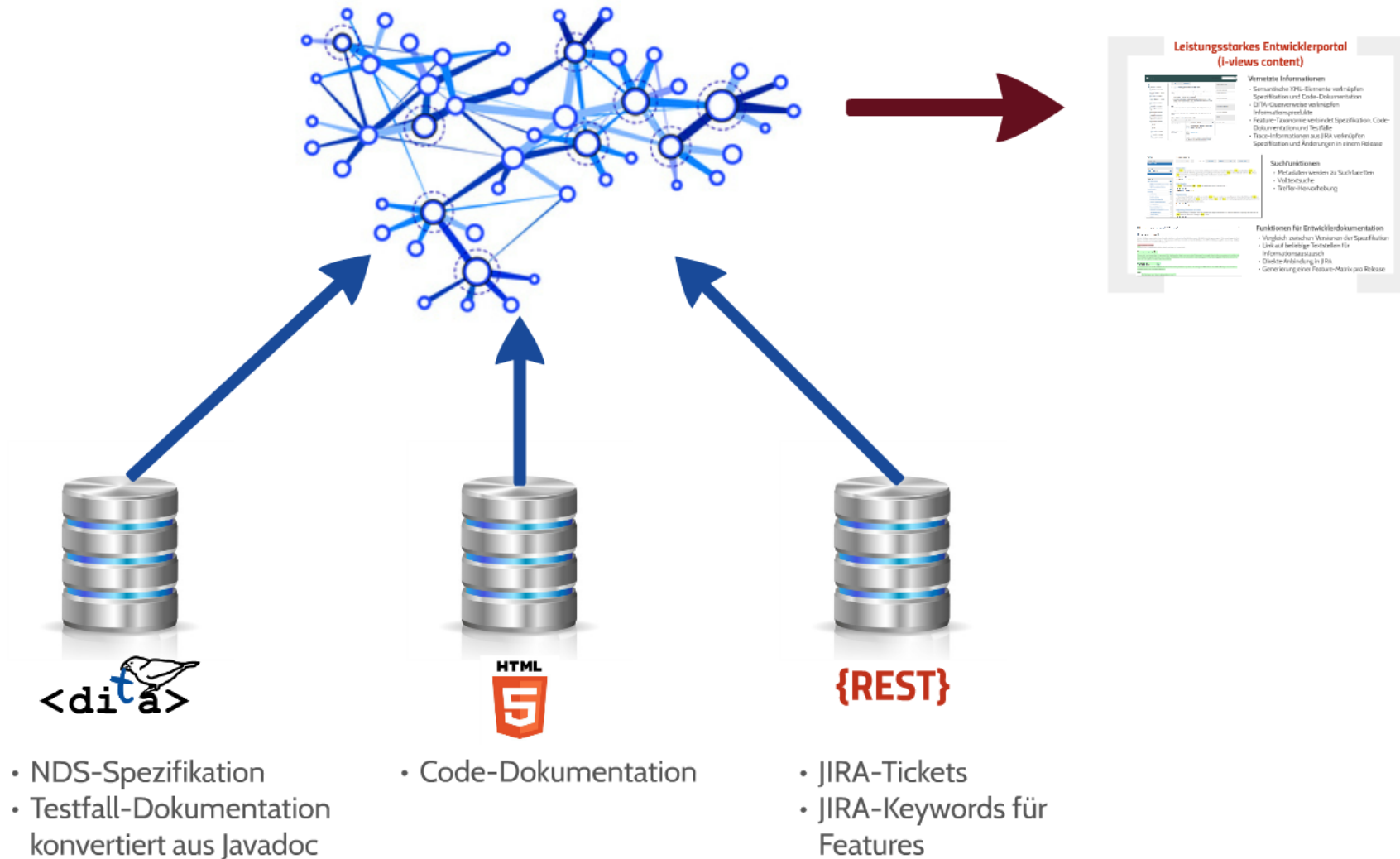
Prozesse und Infrastruktur verbessern



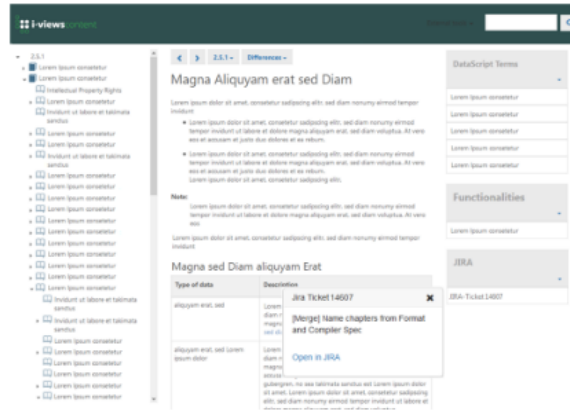
Umstieg auf feature-basierte Entwicklung mit JIRA und Git

- ✓ Gleichrangige Arbeitsabläufe für Entwicklung, Dokumentation und Test in JIRA
- ✓ Alle Arbeitsabläufe komplett in JIRA, auch Review der Dokumentation
- ✓ Automatischer Build der Dokumentation mit Code
- ✓ Dokumentation in Git, eigener Branche pro Feature
- ✓ Für Abnahme eines Features muss die Dokumentation fertig sein. 

Content-Delivery-Portal einführen

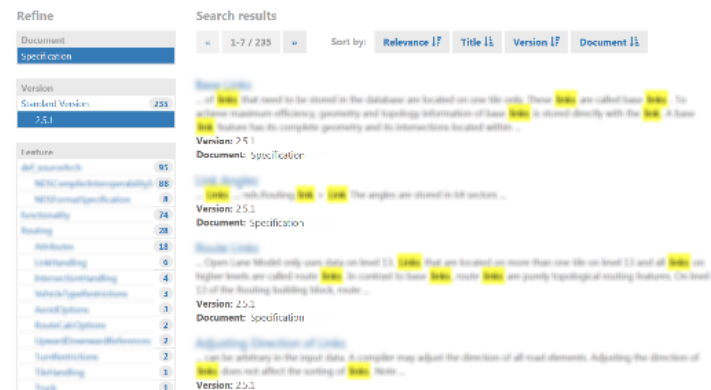


Leistungsstarkes Entwicklerportal (i-views content)



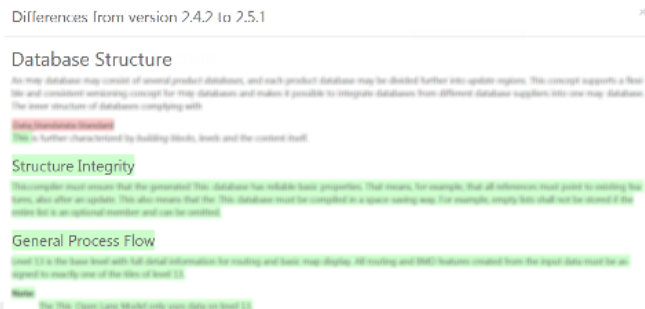
Vernetzte Informationen

- Semantische XML-Elemente verknüpfen Spezifikation und Code-Dokumentation
- DITA-Querverweise verknüpfen Informationsprodukte
- Feature-Taxonomie verbindet Spezifikation, Code-Dokumentation und Testfälle
- Trace-Informationen aus JIRA verknüpfen Spezifikation und Änderungen in einem Release



Suchfunktionen

- Metadaten werden zu Suchfacetten
- Volltextsuche
- Treffer-Hervorhebung



Funktionen für Entwicklerdokumentation

- Vergleich zwischen Versionen der Spezifikation
- Link auf beliebige Textstellen für Informationsaustausch
- Direkte Anbindung in JIRA
- Generierung einer Feature-Matrix pro Release

Leistungsstarkes Entwicklerportal (i-views content)

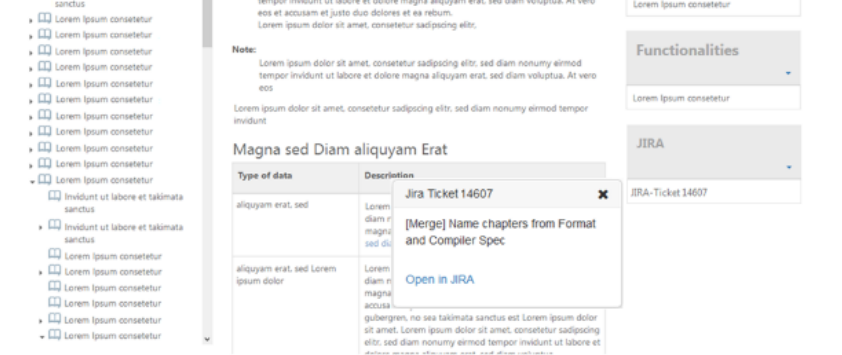
Vernetzte Informationen

- Semantische XML-Elemente verknüpfen
- Spezifikation und Code-Dokumentation
- DITA-Querverweise verknüpfen
- Informationsprodukte
- Feature-Taxonomie verbindet Spezifikation, Code-Dokumentation und Testfälle
- Trace-Informationen aus JIRA verknüpfen
- Spezifikation und Änderungen in einem Release

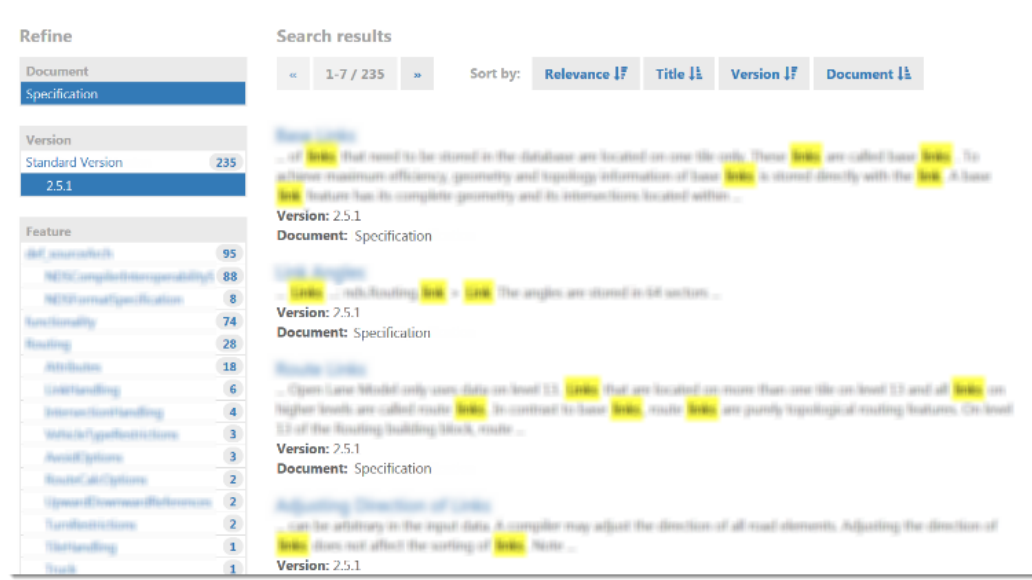
The screenshot shows the i-views content developer portal interface. The main content area displays a document titled "Magna Aliquam erat sed Diam" with a "Differences" tab selected. The document content includes a "DataScript Terms" section, a "Functionalities" section, and a "JIRA" section. A modal window is open over the JIRA section, showing "Jira Ticket 14607" with a description and an "Open in JIRA" link. The bottom of the page shows a "Refine" section with filters for Document, Version, and Feature, and a "Search results" section with a list of search results and sorting options.

Suchfunktionen

- Metadaten werden zu Suchfacetten
 - Volltextsuche
 - Treffer-Hervorhebung
- Copyright parson AG, 2017

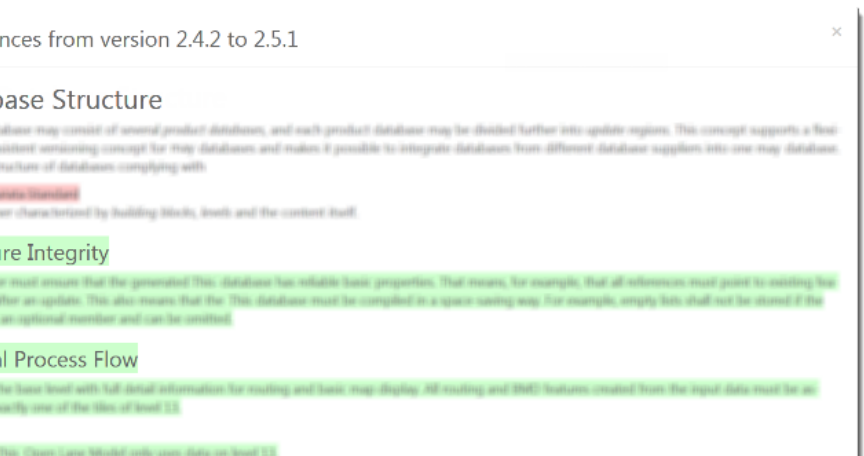


- DITA-Querverweise verknüpfen Informationsprodukte
- Feature-Taxonomie verbindet Spezifikation, Code Dokumentation und Testfälle
- Trace-Informationen aus JIRA verknüpfen Spezifikation und Änderungen in einem Release



Suchfunktionen

- Metadaten werden zu Suchfacetten
- Volltextsuche
- Treffer-Hervorhebung



Funktionen für Entwicklerdokumentation

- Vergleich zwischen Versionen der Spezifikation
- Link auf beliebige Textstellen für Informationsaustausch
- Direkte Anbindung in JIRA
- Generierung einer Feature-Matrix pro Release

Refine

Document
Specification

Version
Standard Version 235
2.5.1

Feature

API Interactich	95
API ComputerInteroperability	88
API ControlSpecification	8
Functionality	74
Routing	28
Attributes	18
LinkHandling	6
InteractionHandling	4
MediaTypesReferences	3
HeadOptions	3
RouteAPIOptions	2
OpenAPIHeaderReferences	2
TransferOptions	2
Headering	1
Track	1

Search results

« 1-7 / 235 » Sort by: Relevance | Title | Version | Document

Base Links
... of **links** that need to be stored in the database are located on one file only. These **links** are called base **links**. To achieve maximum efficiency, geometry and topology information of base **links** is stored directly with the **link**. A base **link** feature has its complete geometry and its interconnections located within ...
Version: 2.5.1
Document: Specification

Link Angles
... **links** ... with floating **link** = **link**. The angles are stored in 64 vectors ...
Version: 2.5.1
Document: Specification

Route Links
... Open Lane Model only uses data on level 11 **links** that are located on more than one file on level 12 and all **links** on higher levels are called route **links**. In contrast to base **links**, route **links** are purely topological routing features. On level 11 of the Routing building file is, route ...
Version: 2.5.1
Document: Specification

Adjusting Direction of Links
... can be arbitrary in the input data. A compiler may adjust the direction of all road elements. Adjusting the direction of **links** does not affect the sorting of **links**. Note ...
Version: 2.5.1

Suchfunktionen

- Metadaten werden zu Suchfacetten
- Volltextsuche
- Treffer-Hervorhebung

Differences from version 2.4.2 to 2.5.1

Database Structure

As may database may consist of several product databases, and each product database may be divided further into update regions. This concept supports a flex file and consistent naming concept for may databases and makes it possible to integrate databases from different database suppliers into one may database. The inner structure of databases complying with

Links Interconnect Structure

This is further characterized by building blocks, levels and the content itself.

Structure Integrity

This compiler must ensure that the generated This database has reliable base properties. That means, for example, that all references must point to existing file links, also after an update. This also means that the This database must be compiled in a space saving way. For example, empty links shall not be stored if the entire list is an optional member and can be omitted.

General Process Flow

Level 11 is the base level with full detail information for routing and basic map display. All routing and SMC features created from the input data must be adjusted to nearly one of the files of level 11.

Note

For This Open Lane Model only uses data on level 11

Funktionen für Entwicklerdokumentation

- Vergleich zwischen Versionen der Spezifikation
- Link auf beliebige Textstellen für Informationsaustausch
- Direkte Anbindung in JIRA
- Generierung einer Feature-Matrix pro Release

Dokumentationsquellen verbessern



Kontinuierliche Verbesserung der Qualität



Tiefere Modularisierung über Submaps



Variantenmanagement für NDS-Varianten mit Variablen und feature-basierten Bedingungen



Validierung der dokumentierten Code-Strukturen über Skripte



Pflege und Erweiterung der Feature-Taxonomie

Fazit



- Prozesse sind genauso wichtig wie Tools
- Dokumentation ist Teil der Produktentwicklung
- Standardarchitektur wie DITA ermöglicht flexible Tool-Auswahl
- Semantische Anreicherung der Dokumentationsdaten ermöglicht Features und Auswertungen
- Semantisches Content-Delivery-Portal bietet Zusatznutzen



- [http://www.parson.com](#)
- [http://www.parson.com](#)
- [http://www.parson.com](#)
- [http://www.parson.com](#)
- [http://www.parson.com](#)
- [http://www.parson.com](#)

Fragen?



ulrike.parson@parson-
europe.com

Weiterführende Informationen

- <http://www.nds-association.org/>
- https://de.wikipedia.org/wiki/Navigation_Data_Standard
- <https://i-views.com/de/produkte/i-views-content/#>
- Agile Dokumentation: <https://www.parson-europe.com/de/wissensartikel/370-agile-dokumentation.html>
- <https://www.parson-europe.com/de/blog/440-api-dokumentation.html>