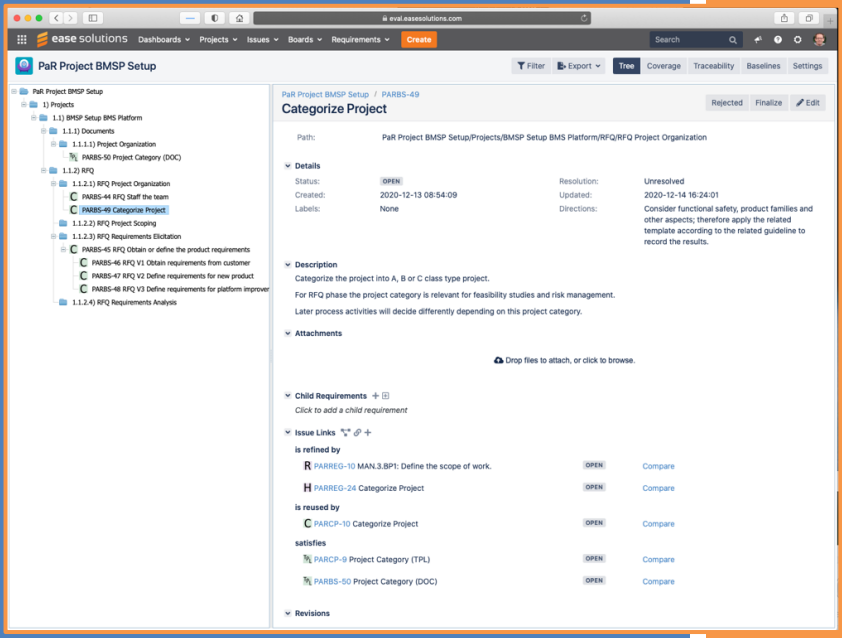
PaR

Processes as Requirements

for Jira with R4J



Systematic

Software

Engineering

Version: 15th October 2022

Publisher: Ralf Bürger
SSE – Systematic Software Engineering
Wilhelmstraße 24
45527 Hattingen - Germany
Ralf.Buerger@ProcessesAsRequirements.info

Print: epubli – a service of neopubli GmbH, Berlin

Community: https://ProcessesAsRequirements.info
http://ProAsReq.info

Acknowledgements: To the people and companies that support me:
Bernhard Doleschel (ease solutions)

"First, don't be afraid. ...
Second, do what you think is right. ...
Finally, build a community.
No one does big things by themselves."
(President Obama, 18.May.2020)

PaR excellence

ProcessesasRequirements.info

your challenges / PaR solutions

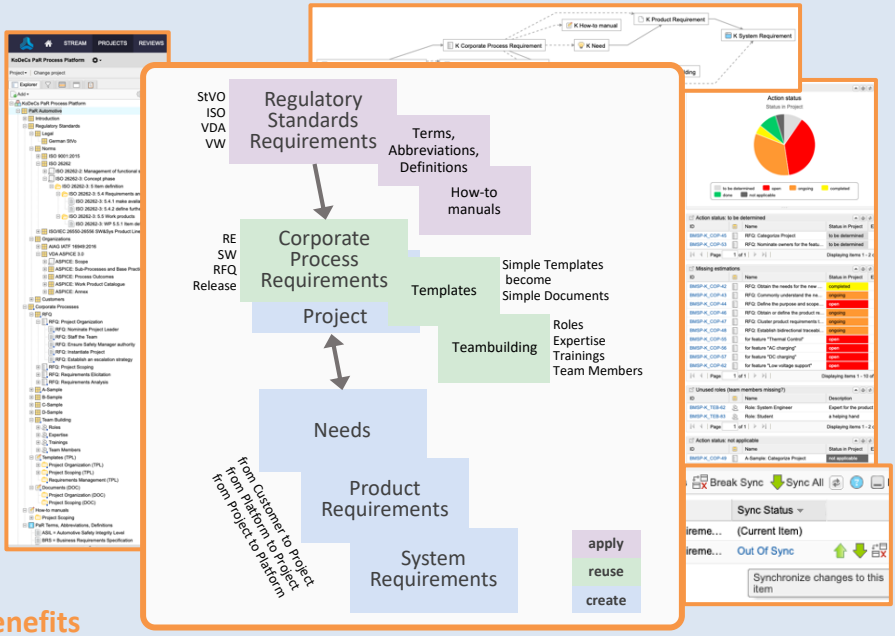
1 / ***help*** **all types of projects with flexible corporate development processes**
1 **PaR:** define and reuse the processes as requirement sets in your RE tool

2 / ***merge*** **regulatory standards with corporate development processes**
2 **PaR:** define also the standards as requirements and add traceability

3 / ***learn*** **with the project teams by established sustainable processes**
3 **PaR:** unite process and product requirements, but improve both

4 / ***comply*** **continuously with processes and standards in the projects**
4 **PaR:** use the features of your RE tool for bi-directional traceability

5 / ***monitor*** **actual project and product maturity progress**
5 **PaR:** measure the status of all requirements, documents and reviews



benefits

① lightweight efficient processes are welcomed by developers
② uniting “what” and “how” in the teams’ tools is more agile
③ standards and processes are focusing on projects and learning culture
④ project teams are empowered to self-organize the compliance
⑤ step by step, teams apply flexible platform techniques also for processes
⑥ it’s a systematic holistic methodical framework that is easy to adopt and adapt
⑦ it makes true transparency for actual process and product maturity

Table of Contents

PARADE OF CHALLENGES SHOWING THE NEEDS 5

PARIS (PAR INFORMATION SYSTEM) 6

PARTOUT - TOOL FEATURES TO SATISFY THE NEEDS OF PAR 7

FEATURE 1: DEFINITION OF REQUIREMENT ITEM TYPES 7
FEATURE 2: IMPLEMENTATION OF THE PARIS MAP 12
FEATURE 3: EVALUATION OF PROJECT MATURITY 16
FEATURE 4: COMPLIANCE CHECKS BY STANDARDS COVERAGE 22
FEATURE 5: SUPPORT FOR PROCESS VERSIONS 24
FEATURE 6: REUSE OF REQUIREMENTS SETS 25
FEATURE 7: SYNCHRONIZATION OF REQUIREMENTS SETS 26
FEATURE 8: DEFINITION AND MANAGEMENT OF VARIABILITY 26

PARTIAL IMPORT, EXPORT, BACKUP 27

PaRade of challenges showing the needs

I have identified **5 challenges** from my coaching of large projects over the past few years. They all can be addressed by changing the way to deal with bulky standards and processes. Bringing these deeply into the projects makes them more intrinsic and natural.

From those challenges I derived needs for a methodical framework. These are outlined on **The Page** (see at the beginning of this document) that is also available as **The Slide**. The whole **PaR** framework is described in a nutshell in **The Booklet**. For more details **The Book** is available.

This methodical framework requires some tool features for realization in organizations. Most modern requirements management tools that are in use in those organizations have at least the basic features on board that are requested here. Nonetheless it is sometimes tricky to configure the tools correctly.

Implementing multiple regulatory standards and setting up a reusable corporate development project process is still a lot of work and needs to be configured correctly. Often it is good advice to get help, support, coaching and maybe also manpower for setup from the experts.
PaR offers help from the community of experts as described on the website.

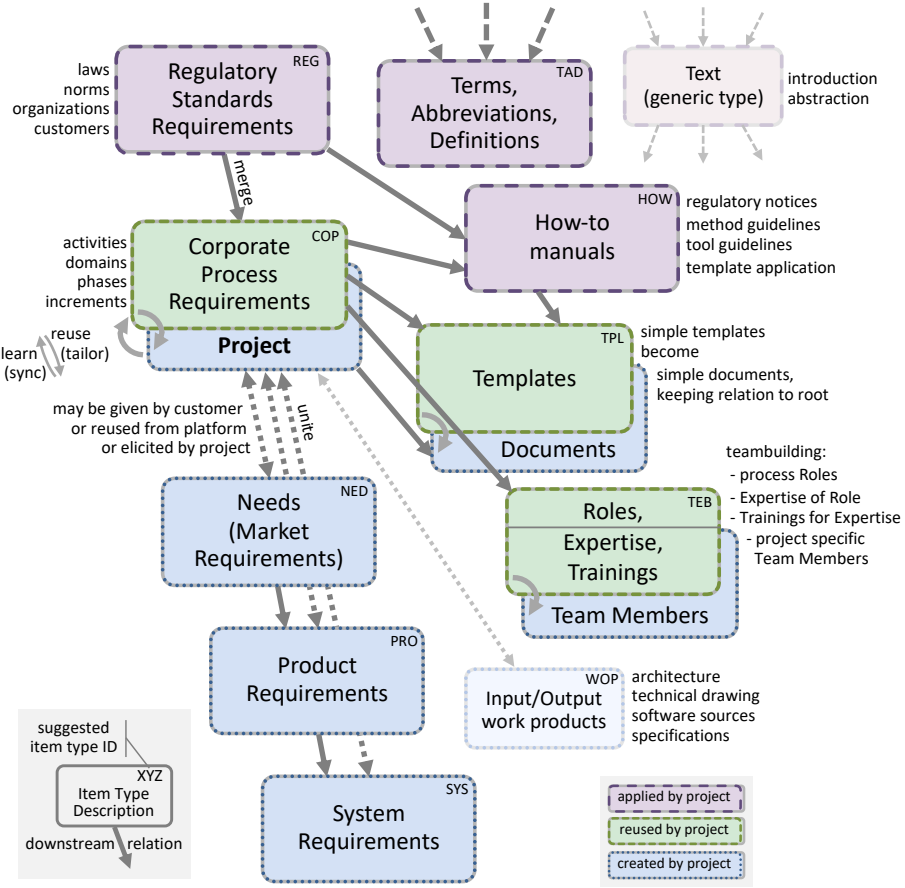
This document shows an exemplary implementation for the tool
Jira from *Atlassian* (https://www.atlassian.com) with the plug-in **R4J** from **ease solutions** (https://www.easesolutions.com). Currently the focus is on the server and datacenter version, not yet on the fairly new cloud version because there we still see the ramp-up of the features. In 2023 we will add T4J (Testing for Jira) to this booklet for both product testing and process reviews with test cases (review checklists), test plans (review plans), test runs (review execution), and test reports (review findings).

This document has been created by Ralf Bürger with support from Bernhard Doleschel from **ease solutions** (he is also a **PaR** community member and strongly supports **PaR** as a methodical framework)**.**

PaRis (PaR information system)

This PaR information system satisfies the discovered needs. It can be implemented in tools by a corresponding set of requirements item types with an item type relationship model. The PaRis is explained in The Booklet and The Book. We show it here only for quick lookup.

- Some requirements are simply applied by projects without change, for detailed lookup or guiding help.
- Other requirements are rather inputs to be reused by projects, also to be modified or extended.
- Reused items, including certain work products, finally become items that are created by projects.



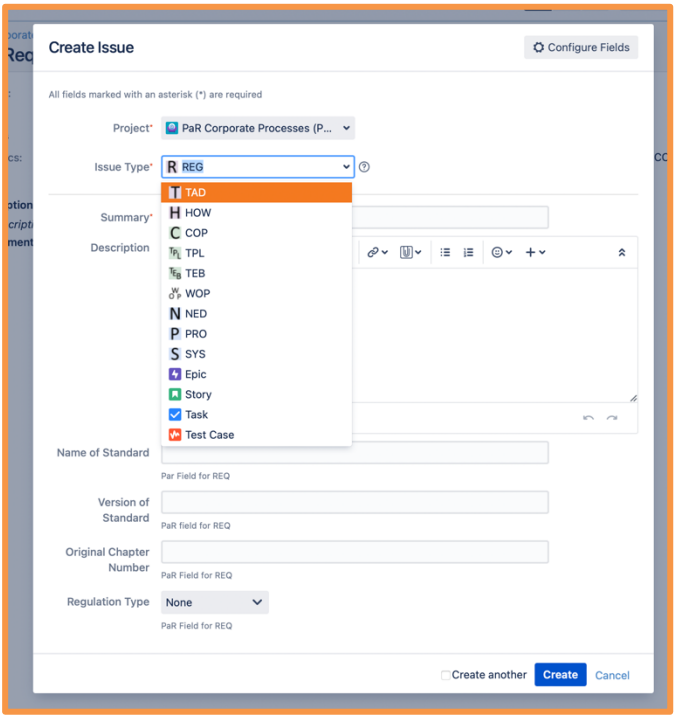
PaRtout - Tool Features to Satisfy the Needs of PaR

This set of features defines what a tool should bring in to be able to fully support the methodical framework with the described PaRis. These features are explained in The Booklet and in more detail in The Book.

Feature 1: Definition of requirement item types

What we call “item”, Jira calls “issue”. We stick with our term “item” here to be consistent with The Booklet and The Book.

In Jira item types can be individually created and configured. In this Jira R4J exemplary implementation, we define the following item types according to the PaRis:

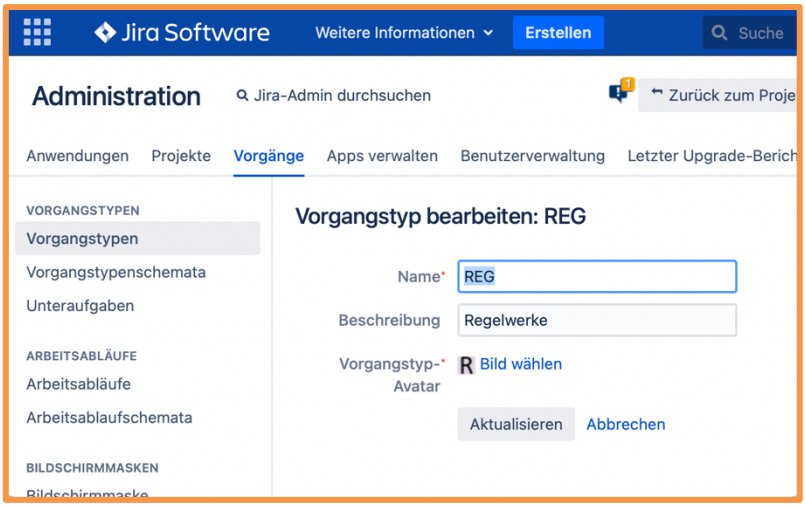


The Jira tool admins create the item types, and the project admins select them for usage in the project.

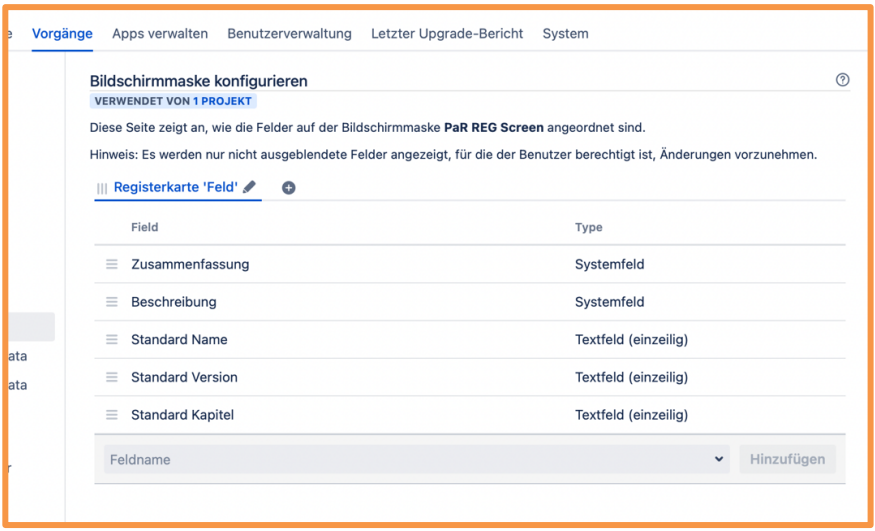
Custom fields allow creation of “select lists” of multiple and single choice. The tool admins define the values, with an addon this can also be done by the project admins.

Jira has a name for each item type, which can be the TLA (Three Letter Acronym) ID from PaRis (e.g., REG). Also, a description and an icon can be selected, which is very useful.

The following screenshot shows a configuration example for the REG item type with a corresponding red icon in a huge German project.



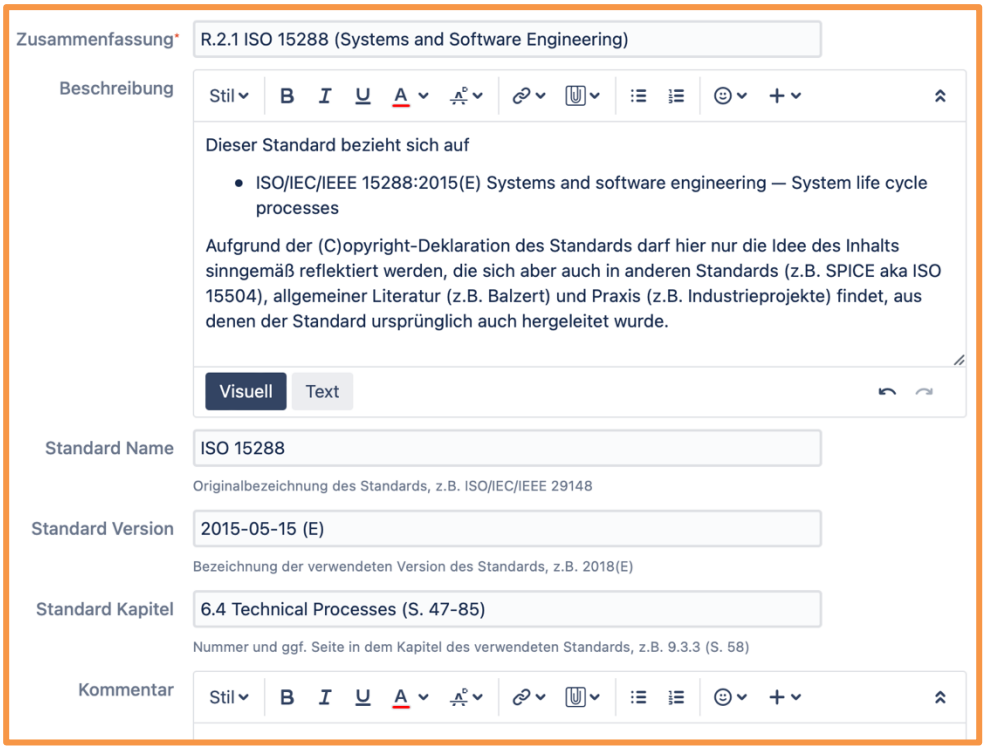
The special field screen mask configuration for the REG in that project looks like this:



We see the system field “Summary” (German: Zusammenfassung), the system field “Description” (German: Beschreibung) and 3 special fields:

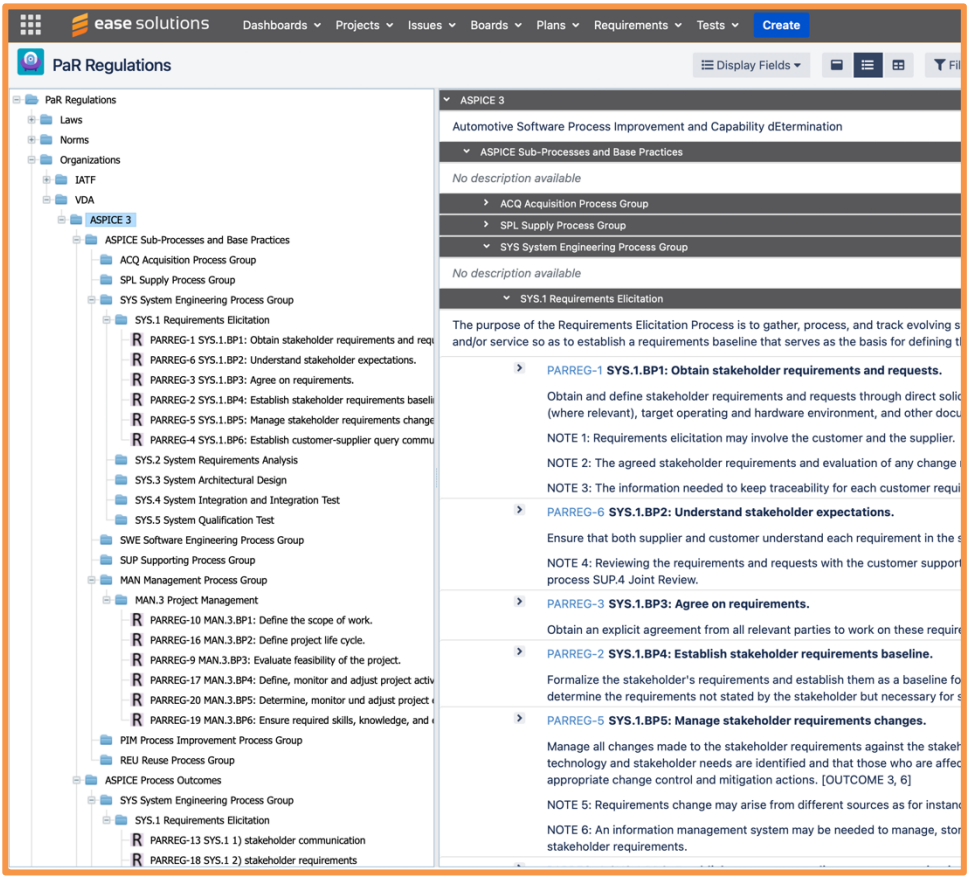
- Standard Name for the name of the referenced regulatory standard
- Standard Version for the version of the referenced regulatory standard
- Standard Kapitel for the referenced chapter of that standard

When the mask is filled for a REG item it may look like this (and here the configured hints are also visible below the fields):



With the help of R4J the requirement items can be listed in Jira in a tree view, something Jira cannot do with items out of the box.

The previous screenshot showed a single item selected with Detail View on the right side of the screen. The following screenshot now shows an example Reading View:



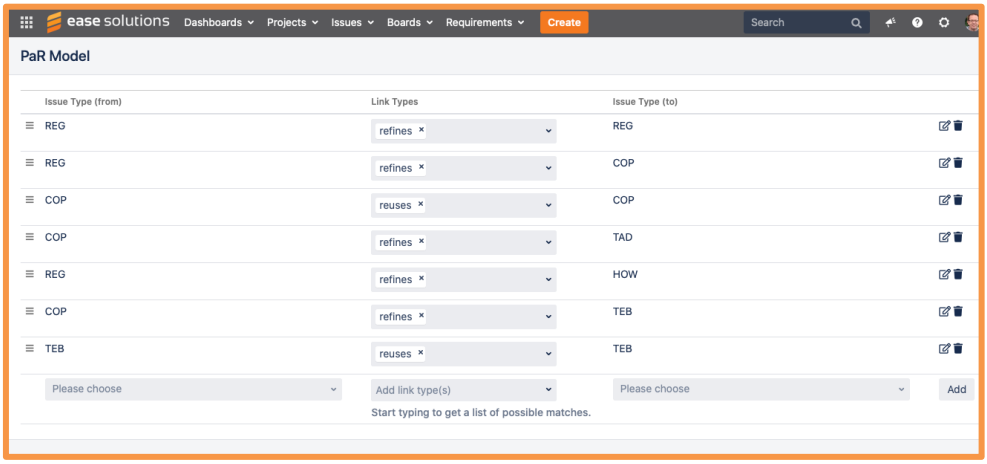
This view gets closer to what an export as a Microsoft Word document looks like, which is much more readable as PDF and on paper.

Feature 2: Implementation of the PaRis map

The following R4J screenshot shows an example PaRis map implementation in Jira R4J, based on the item types as described in the previous chapter.

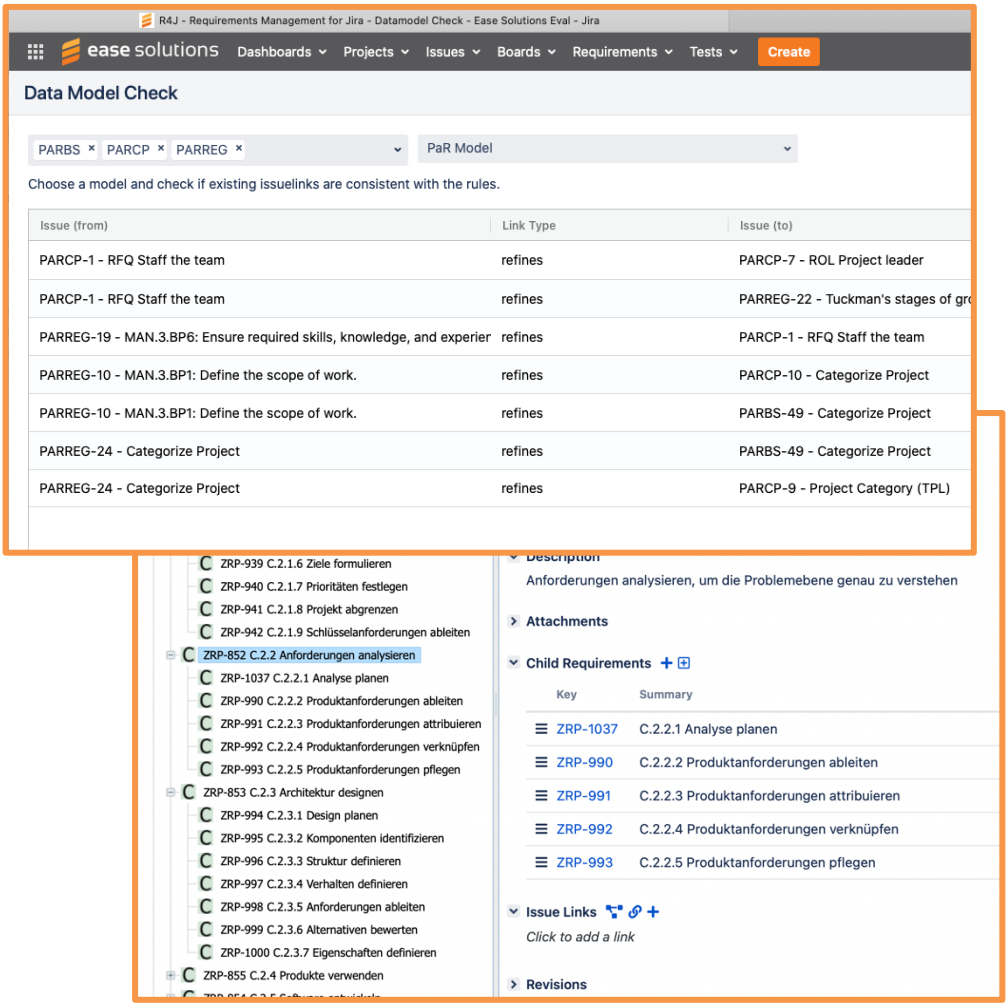
Standard relations for incoming and outgoing are given by Jira by default, others can be defined. The relations are always bi-directional traceable links, e.g., “refines” is looking upwards, and the corresponding downwards looking relation that is automatically created will be “is refined by”. Only the “is related” linkage is the same for both directions because it is on the same level, neither upwards nor downwards.

Jira does not restrict usage of link types to certain issue types. Within R4J there is a solution in place to apply such type of control. When we later create links, then the offered relations are filtered regarding the defined relations.



In Jira R4J there is no graphical tree showing the relations as in the PaRis or any UML/SysML model. But PaR - The Book also has lists of relations for item types in the UML and SysML chapters that can be used for help.

In Jira R4J there is also no automatic check for coverage by types or links, but a Data Model Check can be applied to find items that are not related according to the defined model, as shown in the screenshot below. A Coverage View can also be used to get good overviews and is discussed later in this booklet.

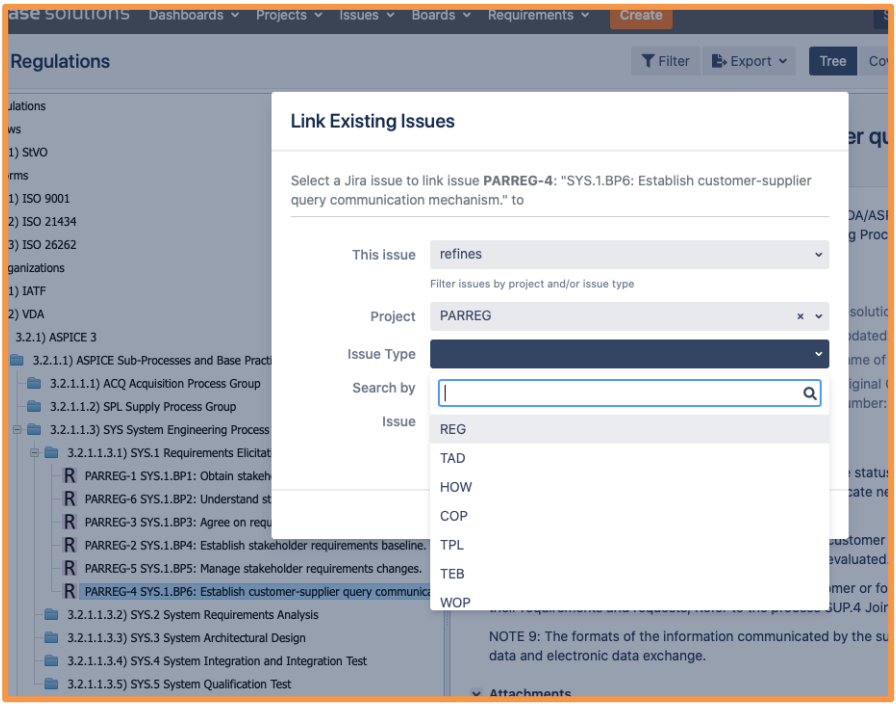


Hierarchical parent-child relations are implemented as implicit links and not as explicit links. The screenshot above shows the “Child Requirements” in a separate list and an empty list of “Issue Links”.

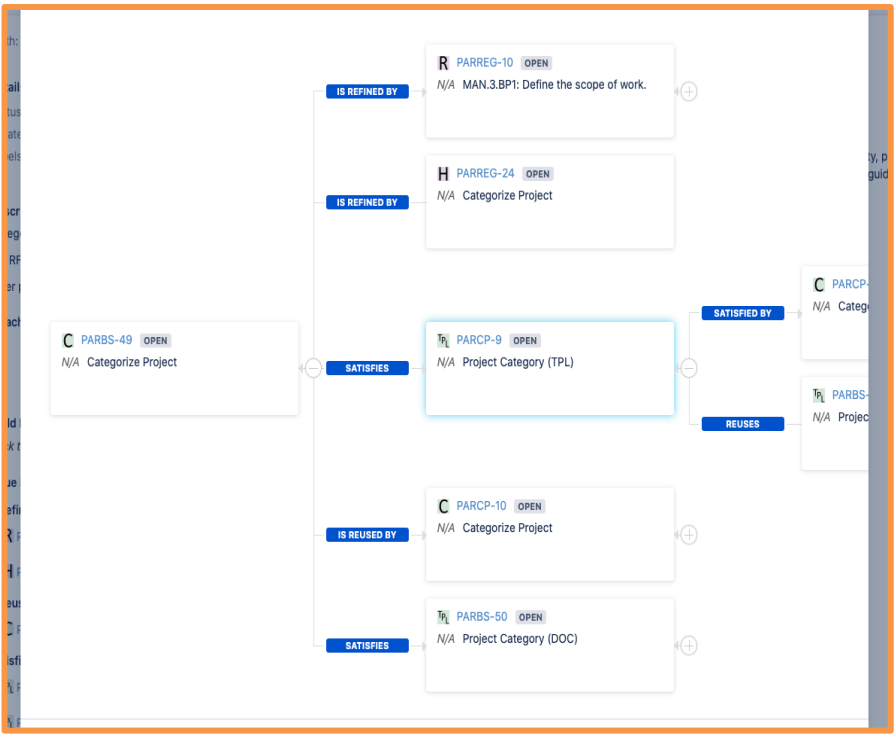
Explicit links to not yet existing items can be created by clicking the + symbol; the new item is created then.

Explicit links to other items that already exist can be created by clicking the link symbol.

The screenshot below shows what happens when the link symbol is clicked: The defined item relations and the defined item types show up as selection options.

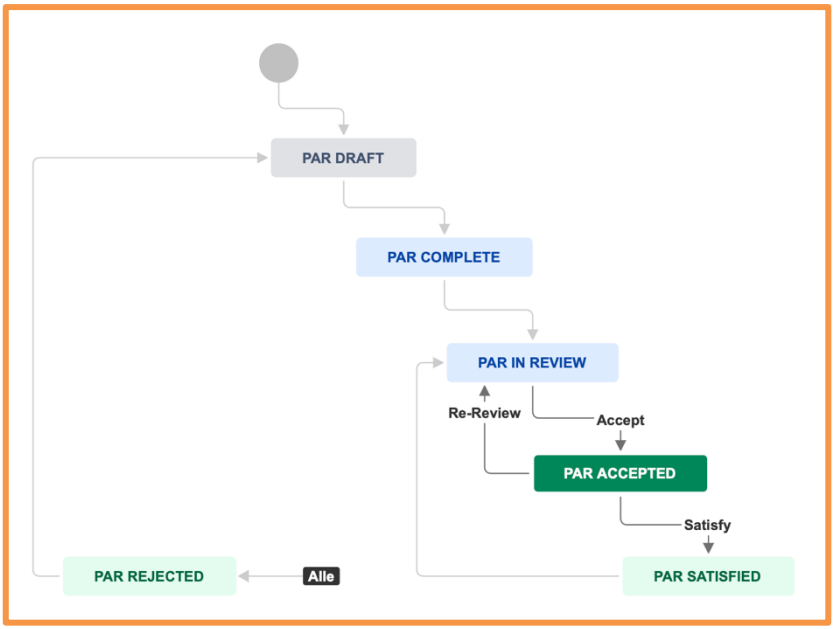


For each item the relations to other items can be displayed as graph with a click of a button. The following screenshot shows such a graph that allows further view options also (filtering relations and showing more fields).



Feature 3: Evaluation of project maturity

Jira allows to define a different workflow for each item type to apply states to the items. The following screenshot shows an example workflow for defined PaR specific item types.



With this workflow a REG, COP, PRO or any other process- or product-related requirement can be

- roughly created (PAR DRAFT),
- defined with all attributes (PAR COMPLETE, for PRO and SYS also with estimated effort and evaluated risk),
- discussed by all relevant stakeholders and team members (PAR IN REVIEW),
- agreed for implementation (PAR ACCEPTED, maybe with related Epics and Stories in a Backlog) and finally
- implemented (PAR SATISFIED, maybe all related Stories and Epics done, tested, and closed) or
- logically deleted (PAR REJECTED), coming from any other status, to prevent implementation (can be used as non-requirement also).

With these status options of the process- and product-requirement items the project progress can be monitored.

When selecting the top folder of the project requirement tree, a statistic comes up which shows in 3 columns the overall number, the numbers by workflow status, and the numbers by item type.

Issues in Folder:	0	PAR DRAFT	310	C COP	91
Issues (recursive):	1254	PAR COMPLETE	73	N NED	57
Folders (recursive):	83	PAR IN REVIEW	255	P PRO	821
		PAR ACCEPTED	561	R REG	138
		PAR REJECTED	4	T TAD	118
		PAR SATISFIED	51	TEB TEB	24
				WOP WOP	5

We see in the example that 310 + 73 + 255 (= about **630**) requirements are still in preparation and analysis, while **561** are accepted to be implemented and **51** are done (PAR SATISFIED). It looks like the project took off for quite a while but will also run for quite some more time because there is a lot of work left to be done: 630 → 561 → 51 could finally look like 0 → 0 → 800 (if only product requirements will be set to PAR SATISFIED).

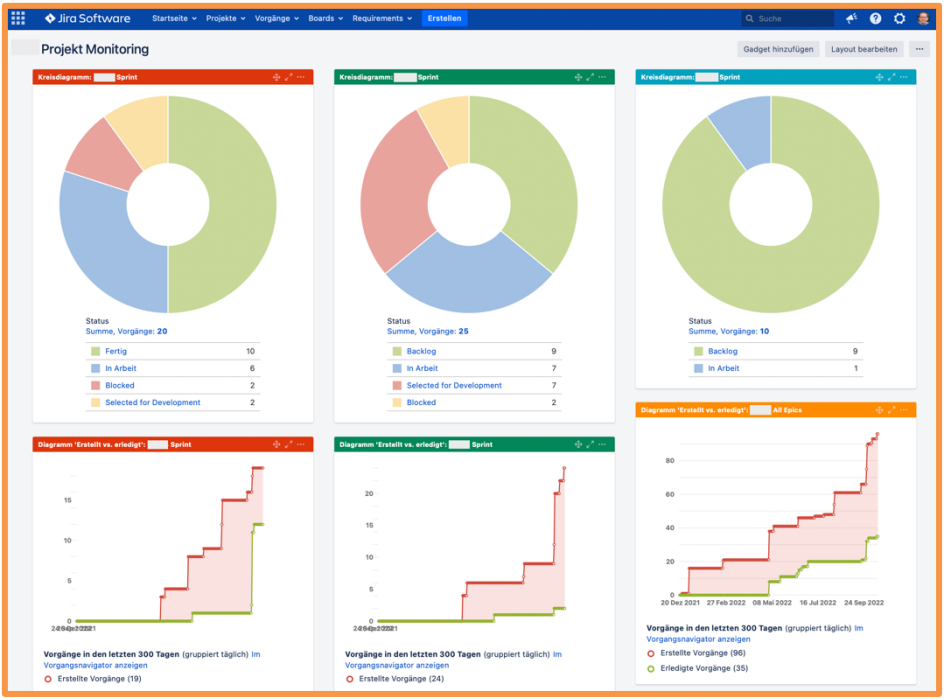
This impression is somehow consistent with 138 regulatory standard requirements leading to 91 process requirement and 57 needs with overall 821 derived product requirements. If it would be just about 100 product requirements the project would have been just started.

By the way: This example project applies PaR even if it is creating a special product directly supporting regulatory standards and processes. This is not the standard case for PaR, and it is explained as tailoring of the PaRis in PaR – The BOOK in more detail. But anyway, we see that with the right tool features and a correct implementation of PaR this is still working fine.

The example above showed that typically not the process is measured to be fully applied, but rather the product requirements to be fully implemented, because the product will be shipped at the end. Well, both is important and in practice it will be a mix of both to check for real project and product maturity from the status statistics, but at the very end the product maturity is always more in focus than the project maturity.

Since R4J is a plug-in for Jira for sure also Epics, Stories, Tasks, and Sub-Tasks will be applied in the project. These items will follow another workflow as the one for requirements shown on page 16, rather like Backlog, Selected, Working, Blocked, Tested, and Done.

These status options can be used in the Jira Dashboard to create donuts and burn-up diagrams to monitor the ongoing project work. And here only the Epics that are derived from and linked to the product and process requirements are used because they finally show the ongoing project work to create the product.

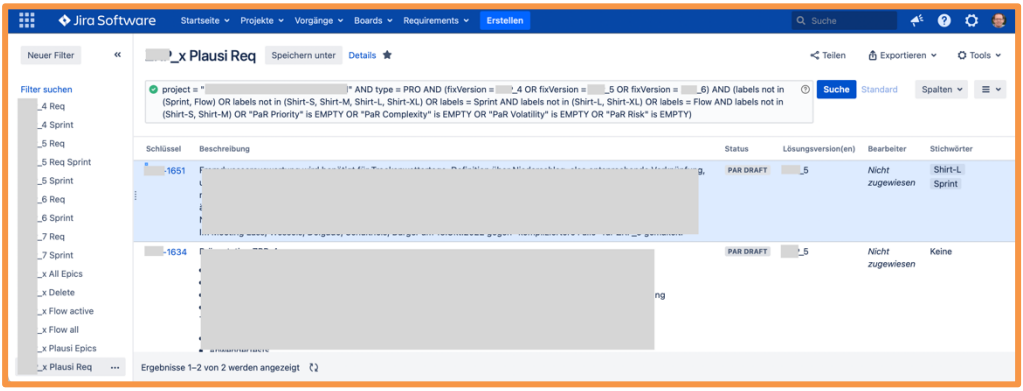


According to the standard Jira dashboard features the diagrams are created by Gadgets, with default Gadgets available for free and further Gadgets available from the Atlassian Marketplace. The screenshot above shows standard Gadgets only, and sadly the colors of the donuts are not configurable and therefore the meaning differs from donut to donut.

The Gadgets can be arranged and configured, e.g., with a colored title bar as shown. The diagram parts and legend elements can be selected to see the queries behind.

In the example above the previous Sprint (left), the current Sprint (middle) and the next Sprint (right) are shown. The next Sprint doesn't have a burn-up yet and we show the overall progress instead in the 2nd row.

The diagrams in the dashboard base on filters (queries) that can be applied to **Jira** items and to **R4J** items as well, and not only as source for diagrams. I use them also for plausibility checks. They are managed by the “Issue Navigator” where they can be created, edited, and taken into favorite.



The used query is:

```
project = "XYZ"
   AND type = PRO
   AND (fixVersion = XYZ_4 OR fixVersion = XYZ_5 OR fixVersion = XYZ_6)
   AND (labels not in (Sprint, Flow)
      OR labels not in (Shirt-S, Shirt-M, Shirt-L, Shirt-XL)
      OR labels = Sprint AND labels not in (Shirt-L, Shirt-XL)
      OR labels = Flow AND labels not in (Shirt-S, Shirt-M)
      OR "PaR Priority" is EMPTY
      OR "PaR Complexity" is EMPTY
      OR "PaR Volatility" is EMPTY
      OR "PaR Risk" is EMPTY)
```

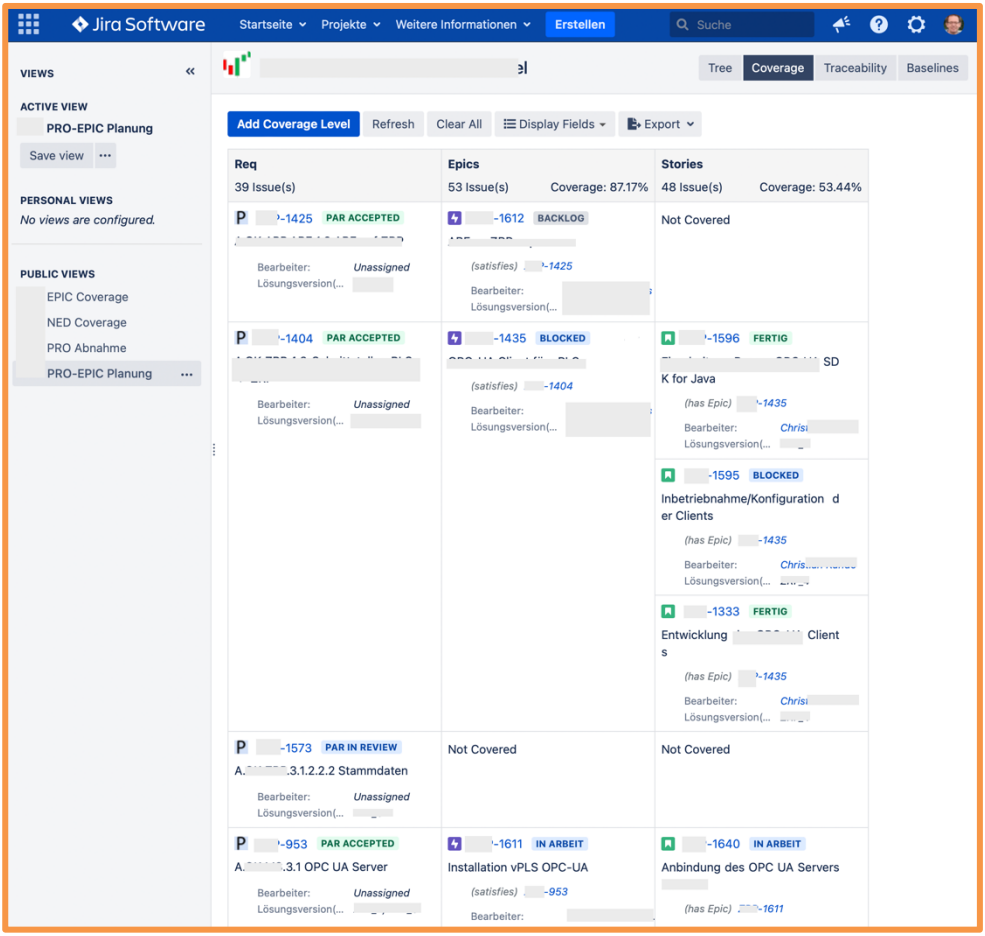
The query shows that in this project we do effort estimations simply by T-*Shirt*-Sizes in the *Labels* field. We also separate larger *Sprint* topics (L and XL) from smaller *Flow* topics (S and M). The former ones are organized as *fixVersions*, the latter ones can be shifted and may be worked off for a longer period, even if they are smaller in size, because they don’t get big attention. Often the *Flow* topics are improvements or silent reworks. We also use the **REGERE** *PaR Risk* evaluation for PRO as described in **PaR – The Book** to figure out whether the requirements are already good enough for a GO with Epics and Stories to be done.

Currently no other tool set can merge the process and product requirements with the doing so efficiently. And such plausibility checks can be used to figure out the formal maturity of the project (and of the team).

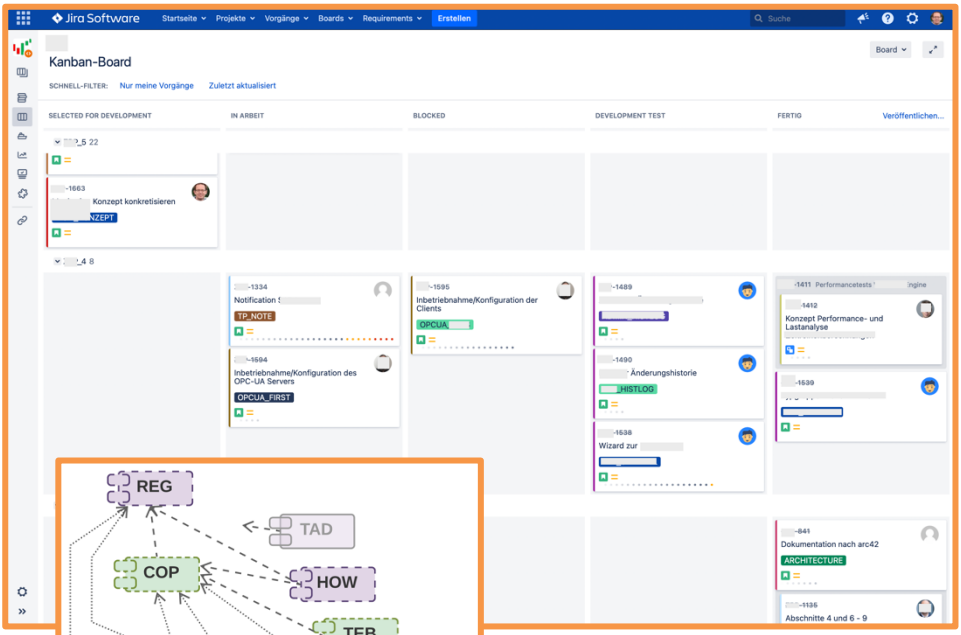
With the Jira R4J coverage view a configurable hierarchy can be displayed. Fields can be added, and the relation of the columns can be configured, and the columns can be filtered as well.

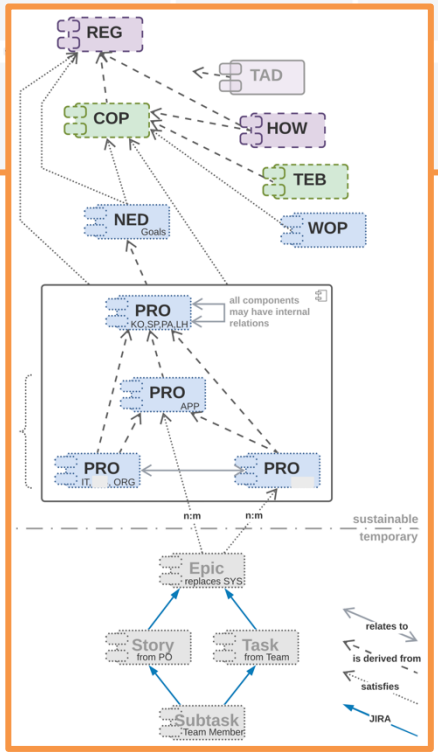
Unfortunately, in some screenshots from a huge critical infrastructure project lots of project specific information must be hidden, but this project is still a very good showcase.

Anyway, this view gives a deeper insight to the progress of the project and links the information from the requirements tree with the information from the boards. Again, we see how efficiently Jira R4J can merge process and product requirements with the project doing.



The Jira boards of the work items inform about the progress in the project according to the item status that is used in the board as columns. In this project we use the fixVersion to organize the board swim-lanes.





Finally for a transparent evaluation of the real project and product maturity, work results should be noted in the items or attached to them (or linked), so that real evidence of results is given and can be checked anytime by quality guys or stakeholders.

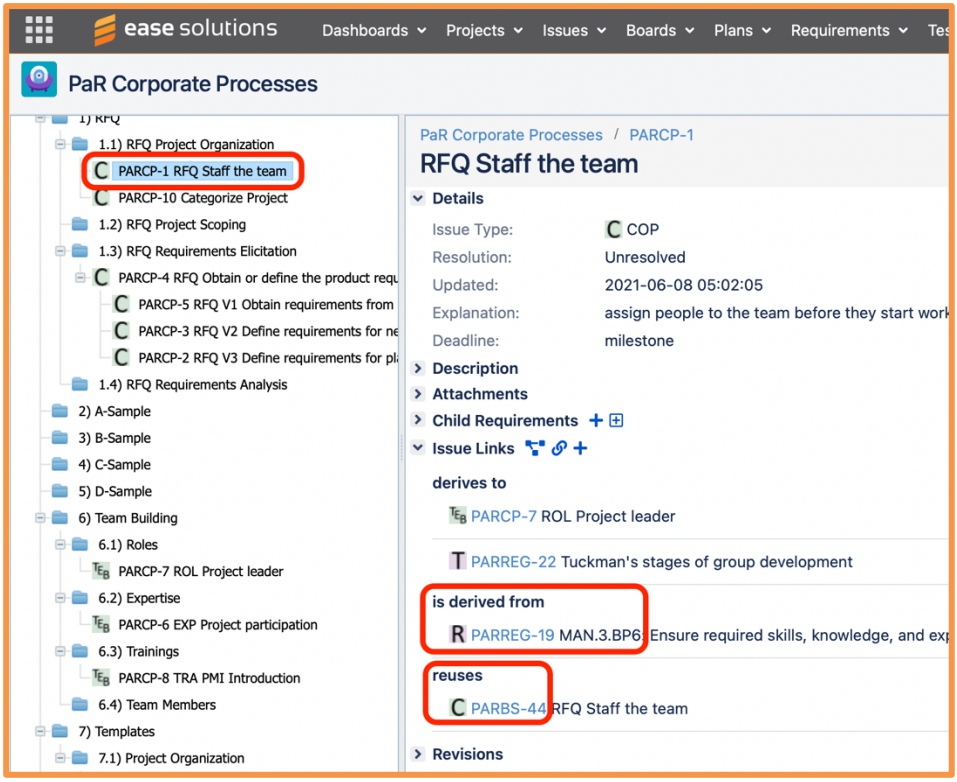
For the project referenced here as example, even the PaRis has been tailored as described in PaR – The Book. The tailored relations can also be checked with Jira R4J in detail for coverage and completeness as described above and below.

Feature 4: Compliance checks by standards coverage

The previous chapter already showed how views, lists and diagrams can be used in Jira R4J for compliance checks also.

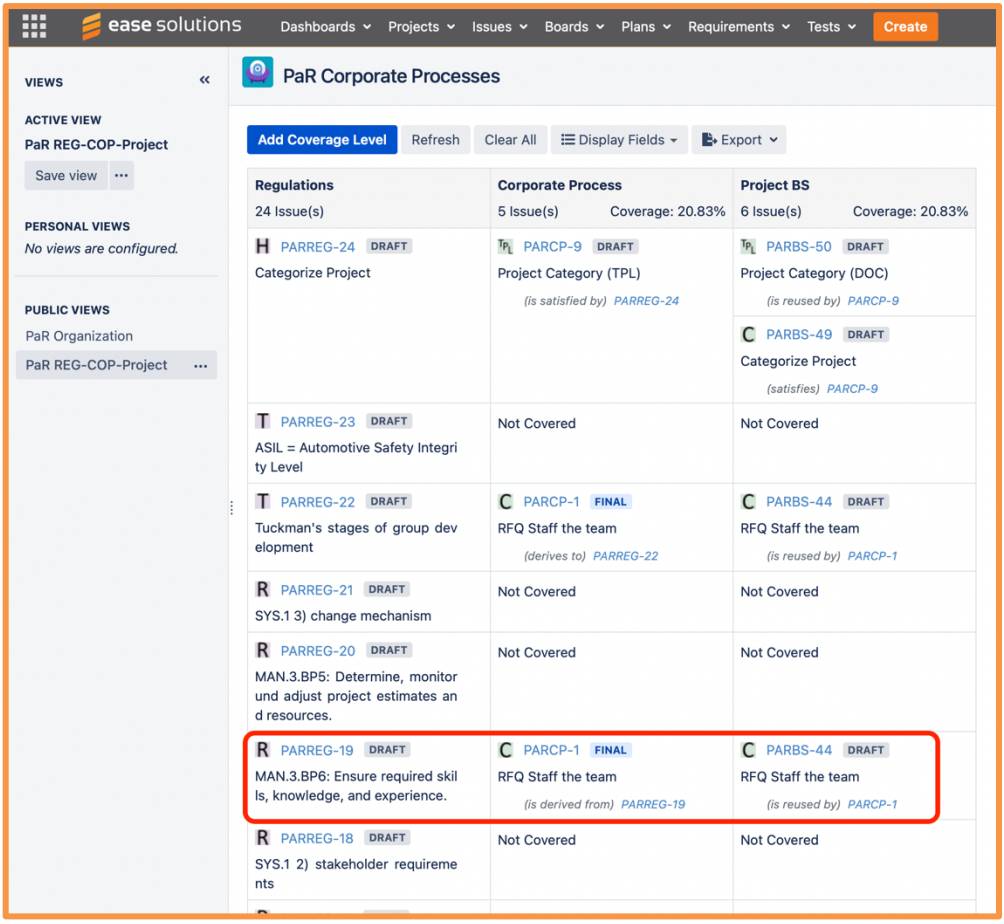
Each single requirement can be checked for compliance to the applied regulatory standards. Also, each single requirement can be checked for reusing the standard corporate -development- process. Usually, the reused corporate process should bring the standards compliance into the project automatically.

In the screenshot below the corporate process is shown, having the regulatory standards “above” and the projects “below”. The selected corporate development process requirement (PARCP-1) is derived from the regulatory standard element (PARREG-19) MAN.3.BP6 of Automotive SPICE, and it is reused in the battery sensor project as PARBS-44.



In the Coverage View shown on page 20 we saw the hierarchy from product requirement PRO items (we don't use SYS items in that project) down to the teamwork with EPIC items and STORY items to check for progress and maturity of the project: PRO → EPIC → STORY.

Now we trace a regulatory standards level REG to a corporate process level COP and reuse down to a project level COP to check the process compliance of the project as done in assessments: REG → COP → COP



The highlighted area shows the single compliance trace of the previous screenshot again: PARREG-19 → PARCP-1 → PARBS-44 (the COP color in the PARBS-44 is green because it is the same item type within the tool).

In addition, **Jira R4J** provides a Traceability View as a two-dimensional matrix of a selected item type over another selected item type. For **PaR** this may be REG over COP.

The term "View" sometimes is misleading because in most views of **Jira R4J** items can be added directly, missing links can be added, and wrong links can be deleted.

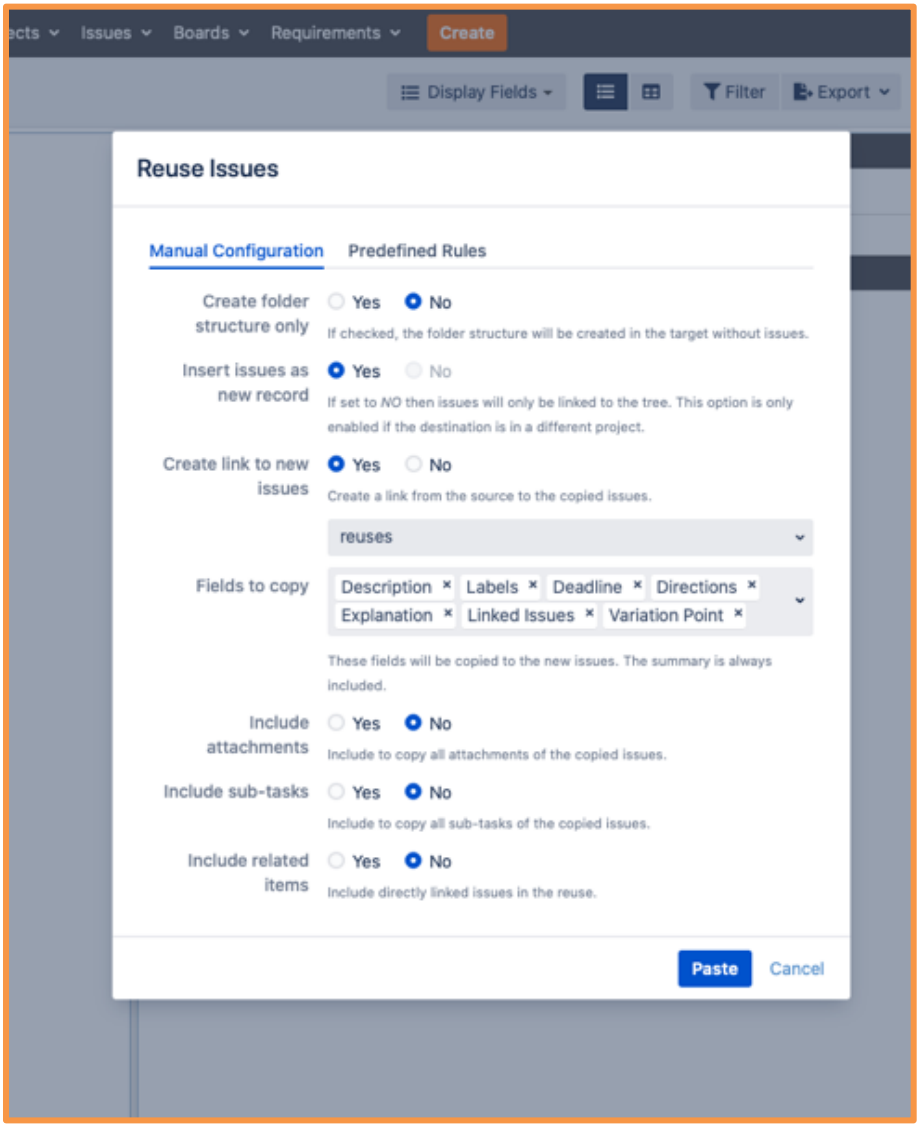
Feature 5: Support for process versions

Baselining a tree of requirements is possible in **Jira R4J**. The difference between baselines can be evaluated, but not with the current version. Also, no update from a baseline into the current version is possible. Furthermore, no real usage of a baseline is possible because it cannot be edited, and it also cannot become a current valid version. To be more precise: Each item has a version and can be rolled back, but not a complete baseline or parts of it.

The suggestion from **ease solutions** is to go with the **R4J** reuse function to create copies of current sets of items, all linked by the "versioned to" link type.

Feature 6: Reuse of requirements sets

Jira R4J supports reuse of requirements sets. It typically creates a copy of the requirements as new items and can optionally keep the relation to the sources. Check the following screenshot for more details of this feature.



Feature 7: Synchronization of requirements sets

Jira R4J does not allow synchronizing reused requirements or requirements sets, but a comparison of requirements is possible. Also, reusing them back is possible manually (same as forward).

Feature 8: Definition and management of variability

Within **Jira R4J** tags can be defined (e.g., entries in the labels-field) and filters for selections of tags can be applied. These selections can then be reused (see Feature 7). Thereby, when requirements are defined multiple times with variation for different variants and are tagged accordingly, then the requirements belonging to a variant can be filtered and reused.

True implicit variability management for a process platform that creates variants of process requirements in a derived project process is not given by **Jira R4J**, but currently also not by any other tool.

PaRtial Import, Export, Backup

For migrating processes from existing process design tools to the requirements engineering tool it is essential to have some import options. Then all processes and sub-processes can be migrated step by step, and finally maybe the expensive licenses for the process design tools can be saved (the requirements engineering tools are needed anyway).

Some architectural design tools can work quite good with requirements. Therefore, it makes sense to also transfer process requirements to those tools. This requires functionalities for partial exports.

At least it makes sense to perform partial backups from time to time. Of course, the complete databases and clouds are saved regularly by central tool administrators of the IT departments, but saving a process release now and then should be possible.

In Jira R4J default exports to XML, Microsoft Word and Microsoft Excel are possible. Also export templates can be defined using a formatting language.

This is an example Microsoft Word export template:

[[$selectedProject.name]]

[[\each $selectedProject]][[\recurse $.folders]]

[[$.numbering]] [[$.]]

[[\recurse $.folderissues]]**[[$.summary]]**
[[$.key]] – [[$.issuetype]] - [[$.status]] [[\if $.fixversions.count > 0]]- [[\each $.fixversions]][[$.]] [[/each]][[/if]][[\if $."PaR Risk" <> ''']]- **(R[[$."PaR Risk"]]** = P[[$."PaR Priority"]] V[[$."PaR Volatility"]] K[[$."PaR Complexity"]])[[/if]]

[[\wiki $.description]] [[/wiki]]

[[\if $.linkedissues.count > 0]]Links to: [[\each $.linkedissues]][[$.]] [[/each]][[/if]]

[[/recurse]]

[[/recurse]]

[[/each]]

When applied for export, then it creates a Microsoft Word document as follows:

PaR Corporate Processes

1 RFQ

1.1 RFQ Project Organization

RFQ Staff the team
PARCP-1 – COP - Final

Links to: PARREG-22 PARCP-7 PARBS-44 PARREG-19

Categorize Project
PARCP-10 – COP - Rejected

Categorize the project into A, B or C class type project.

For RFQ phase the project category is relevant for feasibility studies and risk management.

Later process activities will decide differently depending on this project category.

Links to: PARBS-49 PARCP-9 PARREG-10

1.2 RFQ Project Scoping

1.3 RFQ Requirements Elicitation

RFQ Obtain or define the product requirements
PARCP-4 – COP - In Review

Links to: PARBS-45

RFQ V1 Obtain requirements from customer
PARCP-5 – COP - Final

Links to: PARBS-46 PARREG-1

RFQ V2 Define requirements for new product
PARCP-3 – COP - Final

Links to: PARBS-47

RFQ V3 Define requirements for platform improvement
PARCP-2 – COP - Final

Links to: PARBS-48

…

or in …

… another project:

…

A.GK.XYZ.5 Systemarchitektur - XYZ-150 – Final (**R4** = P4 V1 K1)

Die Systemarchitektur soll für alle identifizierten Komponenten möglichst flexible Schnittstellen vorsehen, um für die Zukunft Austauschbarkeit und Aktualität der Module zu gewährleisten.

Links to: XYZ-45

A.GK.XYZ.5.1 Architektur/Schnittstellen - XYZ-167 – Draft (**R19** = P3 V1 K2)

Offene Schnittstellen sollen in der Architektur definiert werden, um durch individuelle Adapter Auf- und Abwärtskompatibilität und damit Langlebigkeit des Gesamtsystems zu gewährleisten.

Links to: XYZ-27 XYZ-30

A.GK.XYZ.5.2 Schnittstelle Anwendungen/Plattform - XYZ-30 – Draft (**R4** = P4 V1 K1)

Eine Schnittstelle der XYZ zu den Apps soll ausgebildet werden, die sich der Funktionalitäten und Daten der XYZ bedient, um diese systematisch und zukunftskompatibel den Apps zur Verfügung zu stellen.

Links to: XYZ-167 XYZ-164 XYZ-640

A.GK.XYZ.5.2.1 Funktions-API - XYZ-627 – Draft

Ein API (Application Programming Interface) soll den Fachanwendungen Funktionen der XYZ gemäß OpenAPI per Swagger zur Verfügung stehen.

https://de.wikipedia.org/wiki/OpenAPI

https://www.openapis.org

https://swagger.io

…

Here we also see the requirements risk evaluation according to the **REGERE** as described in **PaR – The Book**. Unfortunately, the risk cannot be calculated because **Jira R4J** does not support calculating fields (no formulas). But the user defined project specific fields can be exported according to the template.

Export to Microsoft Excel can also be used for transferring requirements from one **Jira R4J** installation to another one, but better consult **ease solutions** for doing so, because – for example – there might be tricks needed for keeping the original IDs.

In newer versions of **Jira R4J** the export and import in ReqIF format is supported. But take care, dealing with ReqIF can be tricky.

Backups can be created by using the native **Jira** functions because at the end **R4J** requirements items are stored as native items - just like all other **Jira** items - within the **Jira** database. Therefore, at the end it's a backup of the **Jira** database that requires administrative access rights and recommends further exports before doing so.

… From the graphical representation in 3 levels we can derive an organizational responsibility that is not that visible today in many of our departments. Setting up the regulatory standards – i.e. the boundary conditions – separately from the processes and then relating it to each other is something we don't do that clear today. But that would be much more efficient also for sure. …

Central process department, a German automotive supplier

… I'm convinced that PaR is the next step to be more efficient and agile in project even though you have to fulfil A-SPICE, ISO 26262 and ISO 21434. …

Sascha Kobus, CEO KoDeCs GmbH

… Very promising approach, which exploits the reuse potential for product and process aspects in a unified manner. …

Dr. Martin Becker, Department Head Embedded Systems Engineering at Fraunhofer Institute for Experimental Software Engineering (IESE)

… The variant development process for architecture is among the best we have seen. We consider this approach to be state of the art and benchmark. Especially the strong link between platform and project architecture …

Feedback from an expert discussion of a process for platform-based product development, that I created over some years for a German automotive supplier. My basic platform ideas of that process also made their way into the **PaR** approach for process platforms.

Did your processes become a heavyweight backpack to be carried by the projects, rather than a lightweight intrinsic approach that really helps the teams to navigate through the storms of the projects?

It gets better when you design regulatory standards and Processes as Requirements that are reused in and improved by the projects.

Ralf.Buerger@ ProcessesAsRequirements.info
https://ProcessesAsRequirements.info