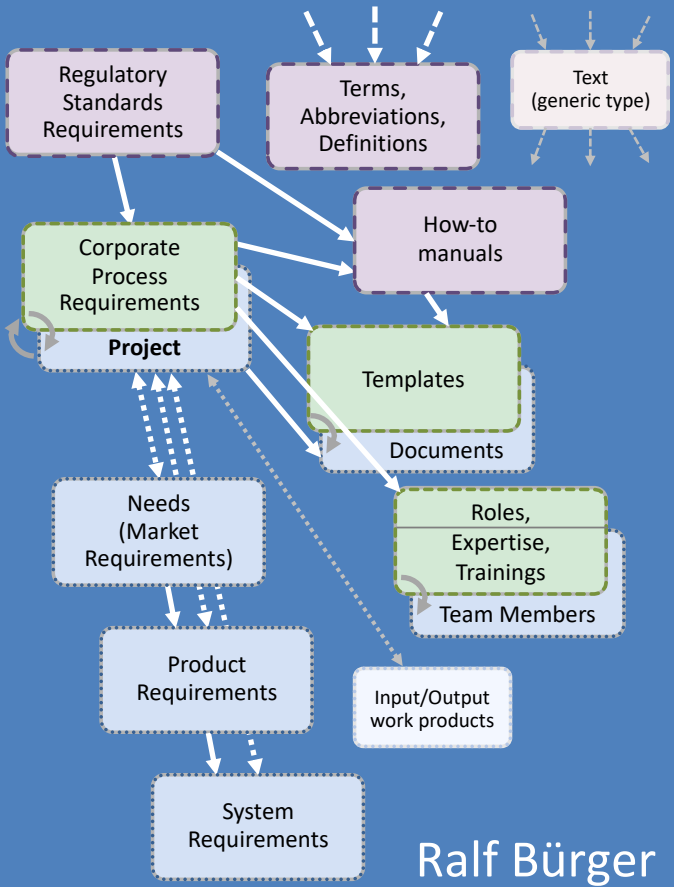
# PaR methodical framework

## Processes as Requirements

# The Book




Ralf Bürger

Systematic Software Engineering

# Contents

# 3 PaRade of Challenges Showing the Needs

## 3.1 Many projects coached – 5 Challenges identified

1. When a project team claims that following the processes causes extra effort without any ***help*** this should be heard and analyzed and changed, but it quickly turns out to become a real challenge.
2. Nowadays we often perform projects or organize departments in an agile fashion to stay focused, to inspect and adapt frequently, and to enable teams to do the right things. We give responsibility to the people doing the work and let them organize themselves.[2] Then we may figure out that the teams forget to ***merge*** their doing with the standards to ensure corporate sustainability. Motivating teams to apply regulatory standards can be a real challenge after passing the responsibility to them.[3]
3. We see innovations spreading faster, new products emerging in variants, windows of opportunities closing quicker, paradigm shifts like "from oil to Tesla", and even the "first new" pandemic. We have to ***learn*** quickly, but the processes to be applied in the projects often are the same as a decade ago, getting more and more heavy-weight and almost frozen like being defined once and forever[4]. This needs to be challenged and changed.
4. While I am writing this, I am also helping a new project team on a highly innovative product. They have just been audited and assessed by officials how they ***comply*** to regulatory standards. It made the key experts feel like kids back in school who actually don't get it - not good! Better catch people doing something right – and do it often to lead the people.[5]
5. The same team has to ***monitor*** and report its progress on funny colored standardized presentation slides to managers who don't know what the project really is about, not to even talk about the projects' obstacles.

◎ Analyzing challenges to figure out the root cause often leads to violation of key values. For the challenges above, the leaders should trust their teams, the teams should respect the corporate responsibility of their leaders, and both should be open minded. And yes, I'm also talking about attitude.

---

R-2 Read "Two Types of Authority Leaders Must Give to Self-Organizing Teams" by Mike Cohn on https://www.mountaingoatsoftware.com/blog/preview/1680

B-3 Read "The Dilbert Principle" about ISO 9000 ☺ (Scott Adams) – ISBN 978-0-7522-7220-7

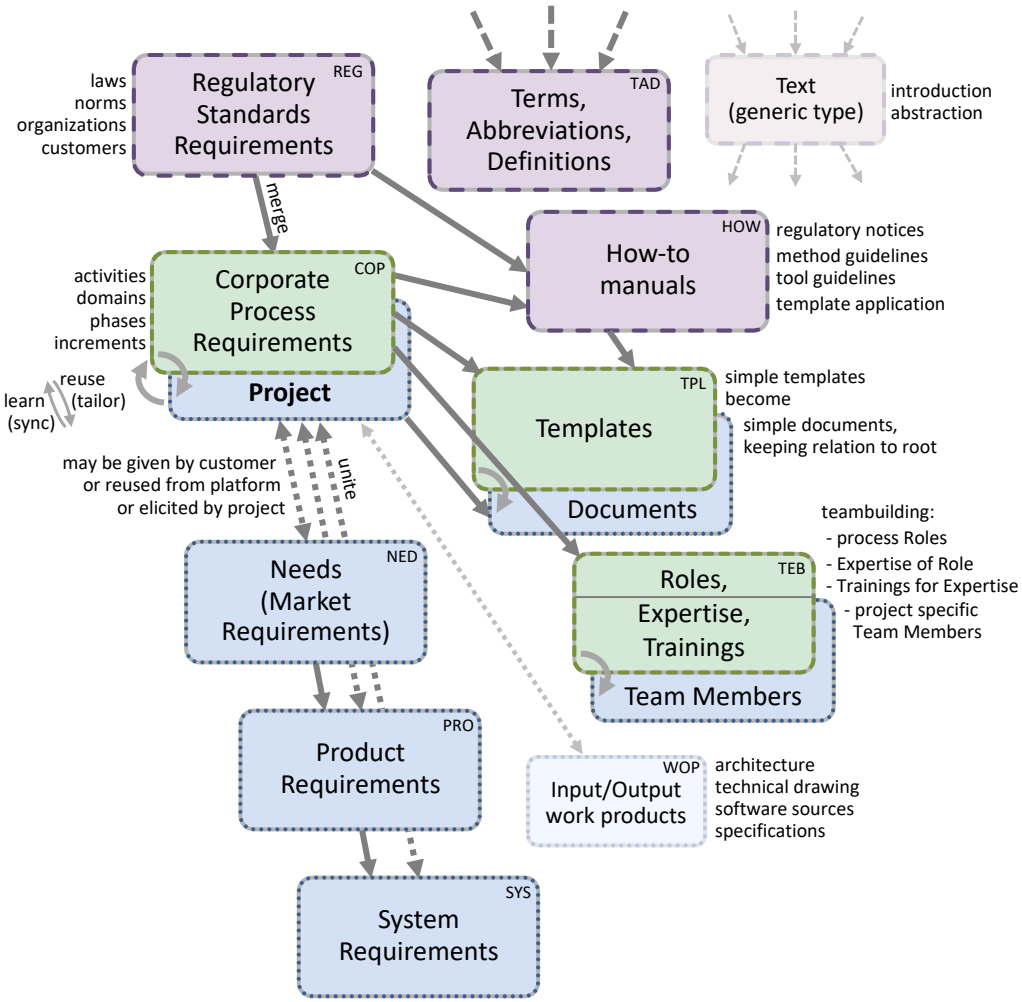
R-4 Read "The New New Product Development Game" by Takeuchi and Nonaka - Jan.1986 on https://hbr.org/1986/01/the-new-new-product-development-game

B-5 Read "The One Minute Manager" (Blanchard, Johnson) – ISBN 0-688-01429-1

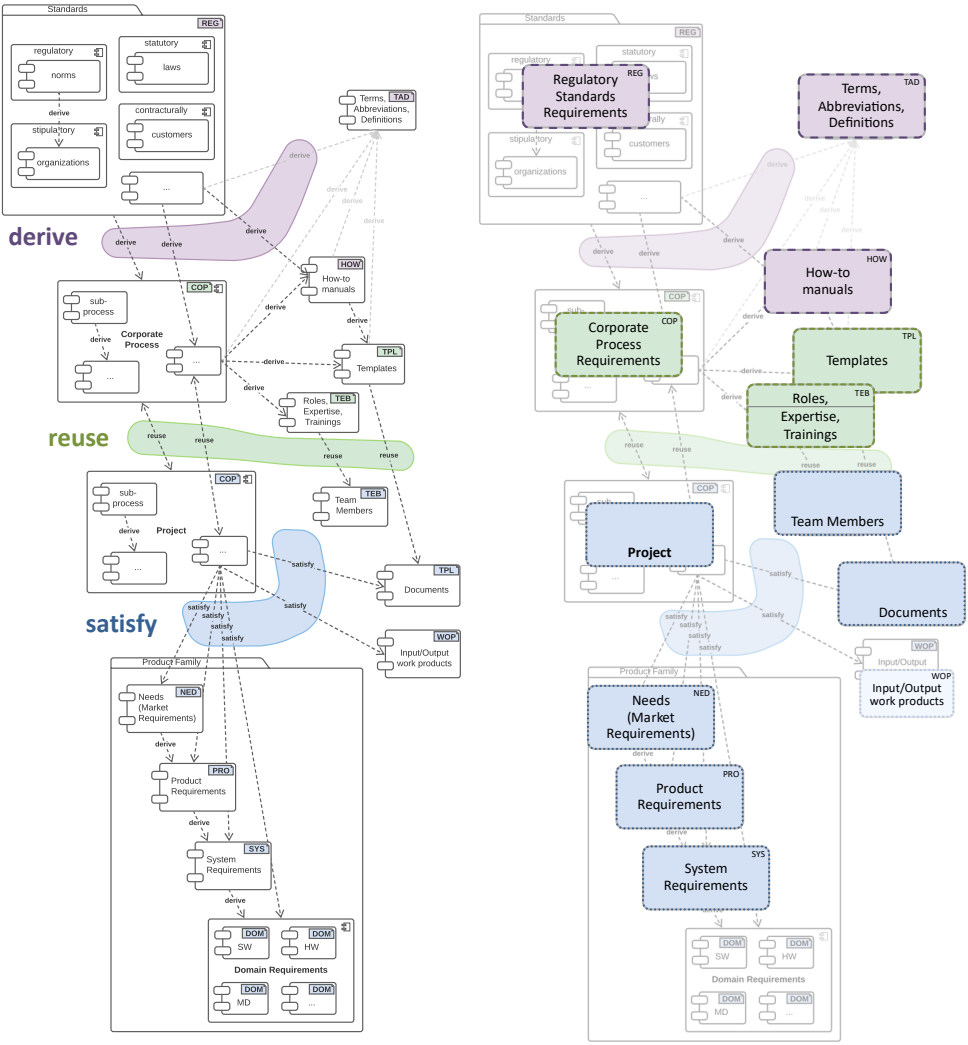
1. Corporate processes ***shall be*** defined as good requirements, ***to*** speak the formal language of the development project teams.
2. Corporate process requirements ***shall be*** clustered to sets, ***to*** support sub-processes for different disciplines and aspects.
3. Sets of process requirements ***shall be*** reusable in development projects, ***to*** pull them as ***help*** into the projects' tools.
4. The development projects ***shall be*** in the focus, ***to*** emphasize that the company sells products and not process or project documents.
5. The possibility of different types of requirement items in the tool ***shall be*** presumed, ***to*** support valued automized transparency.
6. Variability of process requirements sets ***shall be*** considered, ***to*** support flexibility, self-organization and tailoring in projects.
7. The sets of corporate process requirements ***shall be*** organized as a process platform, ***to*** support improvements from projects also.
8. The reused sets of corporate process requirements ***shall be*** related to all types of product requirements, ***to*** map "how" and "what".
9. Reusing corporate processes multiple times ***shall be*** possible, ***to*** support planning of project iterations and product features.
10. Regulatory standards ***shall be*** centrally defined as requirements, ***to*** be able to ***merge*** the corporate processes with the standards.
11. The corporate processes ***shall be*** merged with the regulatory standards, ***to*** ensure compliance and offer help to the projects.
12. Terms, Abbreviations and Definitions ***shall be*** provided and related centrally, ***to*** ensure that all people talk the same language.
13. The process requirements ***shall be*** related to the standards requirements bi-directionally, ***to*** measure coverage up and down.
14. The project teams ***shall be*** enabled to pull the corporate process requirements as needed, ***to*** support modern (agile) teamwork.
15. The relation of process and product requirements ***shall be*** uniting but flexible, ***to*** support different views and ***learn*** with the teams.
16. Each reuse in the projects ***shall be*** bi-directional traceable, ***to*** continuously ***comply*** and automatically measure deviation.
17. Teambuilding aspects like roles, expertise and trainings ***shall be*** considered, ***to*** systematically support the people who do the job.

The core idea of PaR is that the predefined sets of Corporate Process Requirements COP are reused (pulled with traceability) in the projects to make them become Project COP activities, planned with parameters.

At least for the reused COP→COP we should check regularly for tailoring and then ***learn*** from the projects to bring improvements back into the processes by synchronizing COP←COP.



With these **action relationships** it is now possible to design a UML-style diagram (the left one, see enlarged views on the next pages), based on the detailed PaRis diagram on p.30 (mapped to the UML diagram on the right side):



# 5 Principles

## 5.1 Complicate, complex, chaotic

Something becomes **complicate** when it gets many aspects or details and becomes more difficult as a result. But it still has a clear structure and defined rules with clear cause-and-effect relationship. Architecture can help to describe the structure and interaction of the parts, and the behavior.

A good example is a mechanical watch. It already looks complicated when it only shows hours and minutes. Adding calendar, moon phase, and alarm makes it even more complicated. But it works exactly as it has been defined for a very long time, it can be perfectly described, and it works very predictable.



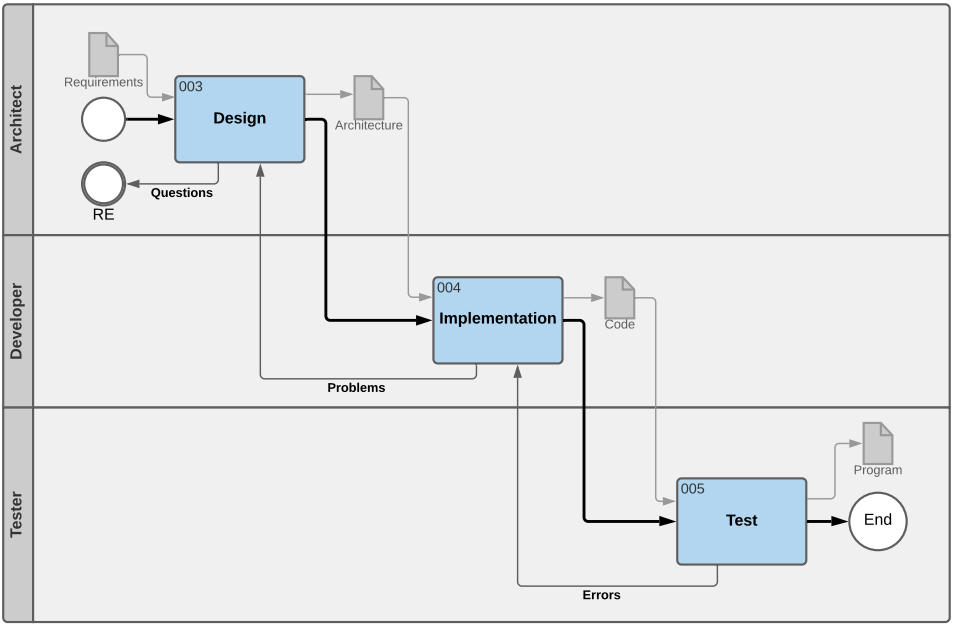
It becomes **complex** when it has too many rules **or** too many elements with a relation of cause and effect only in retrospect.

In a swarm of birds, everyone follows only three rules: keep your speed, keep your distance, and keep your direction. Any bird can understand this, but each parameter varies during flight. Due to the high number of elements (birds) it is such a dynamic system that no one can predict what this swarm will look like just a minute later.

The key aspect of complexity is uncertainty by the lack of predictability.[30] Complexity can be reduced by reducing the number of elements or rules.

W-30 Read VUCA "Volatility, uncertainty, complexity and ambiguity" on Wikipedia: https://en.wikipedia.org/wiki/Volatility,_uncertainty,_complexity_and_ambiguity

**Roles** are added with so-called swimlanes[59], and this also defines a role as a summary of **activities**. Team members taking a process role in the project can check what they need to do and which **work-products** they shall deal with on input and output side. They can also check interfaces to other roles that usually will be taken by other team members.




Even simple drawing tools usually offer shapes for drawing process diagrams according to the standard BPMN[60/61] for BPML[62]. Better tools support reuse of activities, roles and work-products from catalogues, and more sophisticated tools allow modelling these three element types with their relations in a database and automatically visualize the model with shapes in swimlanes.

---

W-59 Read "Swimlane" on Wikipedia: https://en.wikipedia.org/wiki/Swimlane

W-60 Read "Business Process Model and Notation" on Wikipedia: https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation

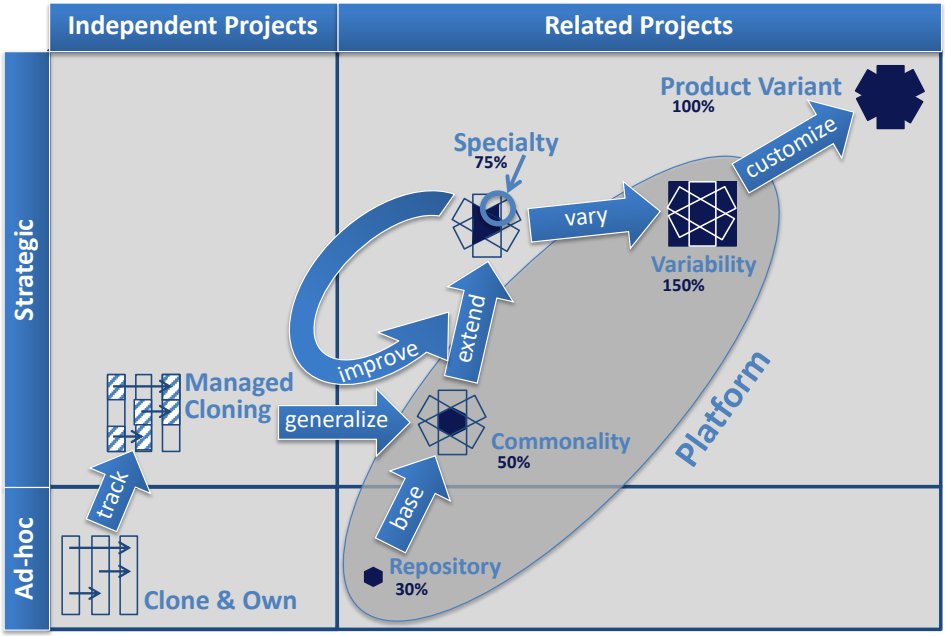
R-61 Read the BPMN standard at OMG (Object Management Group): https://www.omg.org/bpmn/

W-62 Read "Business Process Modeling Language" on Wikipedia: https://en.wikipedia.org/wiki/Business_Process_Modeling_Language

Making the first product of its kind never scales directly and automatically.

Managing the “clone & own” is a good first step towards a platform. That means, if for a new product the archeology finds things in the older versions and variants that can be cloned and owned, it is a good idea to track in a change-management tool that this has happened. This gives the opportunity to trace bugs back to their source or to generalize things to enable reuse, because one can only reuse what has been made reusable before by purpose.

These generalized reusable things can be collected centrally as the commonality of all products of the product family that evolves from the first product of its kind. Another good start is to store the basic stuff, that has always been reused “as is”, in a central repository.




This diagram is based on one created by Martin Becker (Fraunhofer IESE) & Danilo Beuche (pure-systems).

A basic platform like that will quickly lead to reuse as much as possible in new customer projects or new product variants or versions, only extending it with some specialty or modifying a few components. These additions or modifications maybe can also be generalized later on to improve the basic platform.

## 7.2 Teamwork

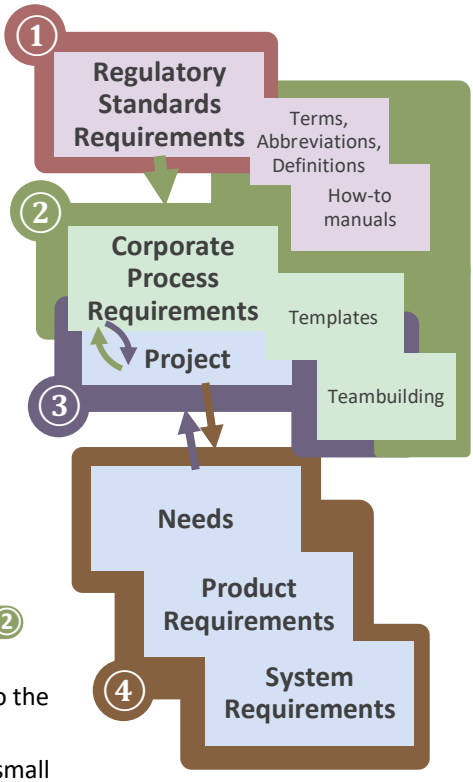
18. The methodical framework PaR supports organizing process teams that efficiently collaborate. While each team focuses on their objectives with high transparency, the members can switch from team to team to work in a cross-functional fashion. This also helps to prevent “social loafing”.[83]

knows all about standards ①

Team 1 may be a single person in a small organization, or even multiple teams joining the ISO/IEEE committees in a large organization.

Anyway, Team 1 drills down all the regulatory standards as sets of requirements, which is a lot of detailed work with many reviews, together with assessors and auditors also, always checking the Copyright clauses and licenses of the standards carefully.

Team 1 also actively coaches the changes on the other teams, mainly Team 2.




knows the company inside out ②

Team 2 has good connections to stakeholders, to Team 1 as well as to the developers that work in Team 3.

Team 2 may be a single person in a small organization starting from scratch, or even multiple teams in a large organization that already has large processes up and running, maintaining them strategically with templates, forms, tools and how-to manuals.

Anyway, Team 2 brings all this into the requirements management tool with all the relations, figuring out the details of the PaRis (information system, or: relationship model), also for the platform approaches.

---

B-83 Read “Social Loafing” in “The Art of Thinking Clearly” (Dobelli, Rolf) – ISBN 978-0-0-06-221969-5

## 7.4 Projects like wildfire

Introducing PaR requires organizations to change, and even more it requires people to change. How can this change be motivated at colleagues and higher management levels?

### We need real arguments and strategies.

If your projects are like wildfire that burns everything, budget, time, motivation, just everything, then the standards and processes are certainly not the cause. On the other hand, following and applying them does not automatically save any project, because *"no matter how smart you are, you spend much of your day being an idiot"* [88].

Think of standards and processes as reusing base practices and best practices. Follow them and apply them to iterate towards a project goal. This may help make project success more likely. And when the going gets tough: *"You don't have enough time in a death march project to do everything the users are asking for. If you build your processes and methods around that sobering fact, you have a chance of succeeding."* [89]

✓ The PaR methodical framework may be ***the cheapest, simplest, and most efficient*** to systematically implement standards and processes and help projects applying them under all conditions.

✓ The PaR methodical framework can be implemented ***step by step*** and also ***partially***, so that the business can go on and the risk of giving it a try is low.

✓ The PaR methodical framework provides the standards and processes for the projects in a ***most flexible way*** to support them in their organization as much as possible, regardless of what else they have to do to withstand *business* wildfires.

---

B-88 Read "The Dilbert Principle" (Scott Adams) and enjoy it ☺ – ISBN 978-0-7522-7220-7

B-89 Read "Death march: managing 'mission impossible' projects" (Edward Yourdon) – ISBN 0-13-014659-5

## 8.5 Finding a compromise

Don't go directly full house into PaR. You may end up with thousands of standards and process requirements in a hierarchy with a dozen of levels. Then you might realize that you just created thousands of requirements instead of thousands of sentences or instead of thousands of diagram items. As described before, think whether you're heading towards an extreme position (it could be a nightmarish giant textual process), think about the other extreme (it might look like a sunny lightweight Process as Requirements), and then think about a compromise position.

A PaRadigm Shift does not mean adaption, but rethinking (see chapter 7.1 on p.94). Think of PaR as a new tool in the box that you can use along with all the good ones already in place to build your product.

I still like diagrams for a process house (see chapter 5.6 on p.80) and process diagrams with swimlanes (see chapter 5.4 on p.70) for high-level process views that show the whole process landscape. Remember the old saying: *"A picture is worth a thousand words"*. A poster that combines the process house with core high-level process diagrams looks great in the hallway.

The process house should last a long time, as if set in stone. The high-level process diagrams without any details will also rarely change. Both provide long-term orientation, and that is good for the agile living at the lower levels. Therefore, those diagrams should only show the commonality and no variability; if they do so the level of detail might be too low.

Project documentation and product specification shall be done as appropriate, but not with requirements. Classic documents or wikis or special tools can help here. Only really small and simple stuff should be written as requirements if they are closely related to real requirements.

Finding good compromises can ***help*** to keep PaR on the bright side of life!

The main focus of PaR is on platforms for **complex** projects of **complicated** products of large companies (see chapter 5.4, 9.1 and 11.1). It can also be applied to single huge projects that last a generation (see chapter 4.7 on p.44). Currently there is no visible need to apply PaR to production processes or standard business processes.

## 9.2 Tool features to satisfy the needs of PaR

### 9.2.1 4 basic features and 4 advanced features

These features define what a requirements engineering tool should bring in to be able to fully implement the PaR methodical framework with the described PaRis map.

The 4 basic features deal with the **entirety** of the standards, processes, and projects, by defining item types, implementing the PaRis, evaluating the maturity, and checking compliance by coverage.

**Basic features**

1: Definition of requirement item types

2: Implementation of the PaRis map

3: Evaluation of project maturity

4: Compliance checks by standards coverage

The 4 advanced features bring in the **structure**, by thinking about versions, reusing parts when needed (instead of all upfront), updating those parts in both directions, and managing variability to reuse variants.

**Advanced features**

5: Support for process versions

6: Reuse of requirements sets

7: Synchronization of requirements sets

8: Definition and management of variability

Somehow in this list later features depend on earlier ones, e.g., implementing the PaRis requires different item types that can be related to each other.

Inversely it shows the purposes also, e.g., the definition of different item types is needed to define an information system.

The following table shows some requirements engineering tools and their capability for PaR (at late 2020). But this is not at all a recommendation or warning, and sometimes it depends on the experience with the tool or on the version of the tool. Some experts of the PaR community can give detailed advice (see PaR website).

| Feature / Tool | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Jama | good | good | good | good | fair | good | good | lack |
| Jira with R4J | good | fair | good | good | pale | fair | lack | lack |
| Polarion | good | fair | fair | good | lack | fair | lack | lack |
| PTC Integrity | good | pale | fair | fair | pale | fair | lack | lack |
| IBM DOORS 9 | pale | lack | pale | pale | pale | pale | lack | Lack |
| IBM ELM | good | good | good | good | good | good | good | good |

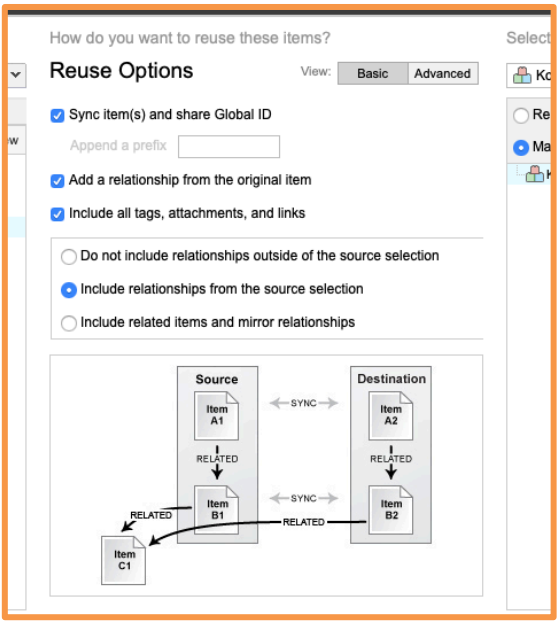
### 9.2.7 Feature 6: Reuse of requirements sets

To be able to apply the **Processes as Requirements** to multiple different projects, a reuse functionality must be available. The worst case is "copy & paste", better is sharing, the best case may be reusing a dedicated version, maybe based on branching/merging concepts or on binding variability to variants.

No company establishes processes only for one project. Therefore, the processes should be a project independent standard that can be reused by multiple projects. When defining the **Processes as Requirements** the requirements engineering tool should provide a reuse functionality to support this. Some options may further help to optimize the reuse, e.g. including relations or attachments. It should also be possible to define for each item the attributes or fields that shall be reusable.

The worst case is only having a simple "copy & paste" functionality because it doesn't cause a sustainable trace. Better is a "sharing" functionality that is made for distributing items because it often keeps a trace to the source.

The best case is a real "reuse" functionality for a dedicated version because it creates a sustainable bi-directional traceability between the items.




Such a function may also keep the relation to its source with the option of synchronizing changes later in either direction (upstream or downstream), also with some options.

Keeping the original additional external relationships ensures that e.g., the REG requirements exist only once but are still related to the reused COP – perfect!

## S