Lessons Learned from 24 Years in Systems Engineering

Requirements Breakdown Structures, Traceability Schemas, Attributes Standardization, Version/Variant Management, …

Dr. Bernd GRAHLMANN
Bernd@Grahlmann.net
www.grahlmann.net

Agenda

1. Bio highlights;
2. Goals of the Presentation;
3. Requirements Breakdown Structures (RBS);
4. Traceability Schemas;
5. Attributes Standardization;
6. Version/Variant Management;
7. Questions & Answers (Q&A).

Dr. Bernd GRAHLMANN – Bio Highlights / Background …

1. Computer Science & Medicine background (Software for automatic diagnostic of the human hip based on 3D computed tomography data + tools for operation simulation);
2. Project Director – Tool for modelling, simulation and verification of parallel systems (30 programmers, 500.000 lines of code, distributed worldwide, SUSE Linux …);
3. 3 years Global Manager DOORS & Requirements Management GE Medical (2000+ engineers, process, guidelines, client/server installations, training, support, templates, project setup and migrations, coaching, evangelist, …);
4. 20+ years in various industries (such as medical devices, railway, automotive, space, aviation, aerospace, defense, energy, banking, pharma, semiconductors, software, elevators, building, gaming, ...) successfully setting up requirements engineering / management / development (incl. interfaces with verification & validation, change and configuration management, risk, (functional) safety assurance, ...) and, in particular, IBM Rational DOORS (ex QSS DOORS and then Telelogic DOORS) / IBM Rational DOORS Next Generation (DNG), Siemens Polarion, Visure Requirements ALM, … for a good number of companies worldwide; training and coaching thousands of engineers for hundreds of up-to multi-billion USD/EUR/CHF projects.

Goals of the Presentation

- ✓ Pass on the quintessence of lessons learned from 24 years in systems engineering;
- ✓ Raise awareness and interest of huge leverage potential for success of:
  1. Requirements Breakdown Structures (RBS);
  2. Traceability Schemas;
  3. Attributes Standardization;
  4. Version/Variant Management;
- ✓ Provide top-level directions on how to succeed;
- ✓ Provide mainly tool independent, but also some tool specific examples of artifacts which:
  1. have proven to be beneficial;
  2. have sufficient ‘meat’ to let you start a long, long, long … thinking journey with kick-starts towards success.

➔ PLEASE, don’t even think about trying to read the slides now – just listen and follow … !!!

Requirements Breakdown Structure (RBS) [I]

No matter what (systems of) systems you want to come up with, the ‚Requirements Breakdown Structure' (RBS) is THE core.

1. Start with the ‚System Decomposition(s)' of your (systems of) systems - I thought that I had been clear = PLEASE, don't try to read the slides now – just listen and follow … (e.g., *medical device, train, locomotive, aircraft, satellite, elevator, vehicle, power plant, ship, submarine, nuclear fusion reactor, ground control, flight inspection system, airborne surveillance system, defense system, cellular product package, GNSS receiver IC, TV, broadcast reception device customer support package, car entertainment system, smart card controller platform, eID, connected truck system, automotive supply, 3D video game, slot-machine, banking software, laboratory management system, fire protection system*, …);
2. Sub-divide complex (upper level) scopes into:
   a) ‚Non-Functional Areas' (such as, ‚*RAM – Truck level*', ‚*Acoustics and Vibrations – Car level*', …),
   b) ‚Functional Areas' (such as ‚*Manage Train Modes – Train level*', ‚*Device Management Functions – Cellular Module level*', …),
   c) (and potentially) Zones, Layout, and Design Guidelines;
3. Potentially add various 'Integrations';
4. Extend with ‚Management Areas' (such as ‚*Certification*', ‚*Documentation and Training*', …).

Requirements Breakdown Structure (RBS) [II]

Using the hierarchical structure of the RBS and the ‚Scopes' of the RBS **consistently during all disciplines of systems engineering for all specifications** is ‚key' to success:

1. Write Technical Requirements Specifications (TRS) for those scopes;
2. Do Architectures (ARCH) for those scopes and inline with the hierarchical structure of the RBS;
3. Write Verification & Validation Specifications (VnV-SPEC) for those scopes (taking those scopes as ‚Test Objects');
4. Base the configuration management plan on the hierarchical structure of the RBS;
5. Base the breakdown into work packages on the hierarchical structure of the RBS;
6. …

Traceability Schema (Pattern) [I]

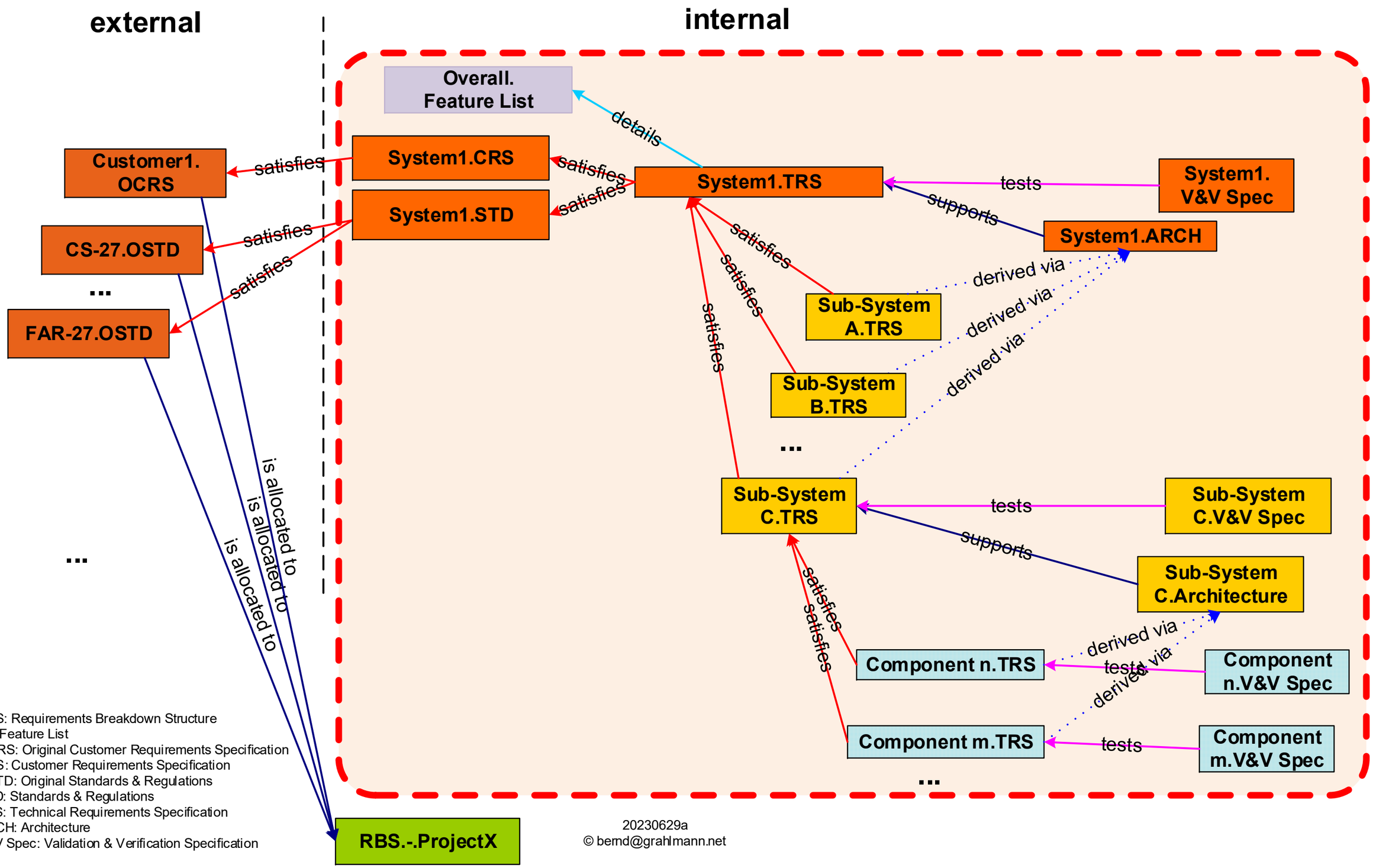
Make the ‚Traceability Schema‘ the second ‚corner stone‘ of the overall approach!

Base it on a ‘Traceability Schema Pattern’ which shows which types of specifications are done and what traceability is established between them.

Distinguish types of specifications:

1. External, such as:
   a) (Original) Customer Requirements Specification (OCRS);
   b) (Original) Standards and Regulations (OSTD);
2. Internal, such as:
   a) Customer Requirements Specification (CRS) giving parts of various OCRSs allocated to a scope of the RBS and annotated with internal ‘interpretations’ …;
   b) Standards and Regulations (STD) giving parts of various OSTDs allocated to a scope of the RBS and annotated with internal ‘interpretations’ …;
   c) Feature List (FL) giving top-level features on a typically upper-level scope of the RBS;
   d) Use Cases (UC) giving (goals of) use cases on a typically upper-level scope of the RBS;
   e) Technical Requirements Specification (TRS) giving black-box requirements on a scope of the RBS;
   f) Architecture (ARCH) giving hierarchical decomposition of a scope (explaining the main contributions of, and interactions between its sub-scopes);
   g) Verification & Validation Specification (VnV-SPEC) giving V&V Cases taking a scope of the RBS as test object (covering the TRS requirements).

Traceability Schema (Pattern) [II]



Traceability Schema (Pattern) [III]

Note that (importance of) Requirements Breakdown Structure and Traceability Schema (Pattern) are tool independent!

In IBM Rational DOORS you may:

1. use an RBS module to give the RBS and the traceability schema;
2. find a suitable DOORS project/folder hierarchy with DOORS folders for scopes;
3. use DOORS Formal Modules for specifications (such as OCRS, OSTD, FL, TRS, ARCH, VnV-SPEC, …);

Depending on your tool implementation, you want to give (also) tool specific guidance for the different systems engineering process steps, e.g.:

- Step 3:
  - System level DOORS TRS module (potentially multiple for different functional and non functional areas):
    - (potentially) Automatically pre-filled (and linked) with CRS and STD requirements
    - Re-organized into XYZ chapter structure
    - Requirements adapted/re-written for XYZ purposes …
    - Missing requirements added, duplications ‚removed' ... and consolidated
    - Target Sub-Systems on next lower level (multiple) selected

It is often opportune to give more detailed, lower-level traceability schemas (zooming in from the specification level to the object level – i.e., showing traceability, e.g., from V&V Cases to Requirements) focusing on different aspects (such as handling of standards and regulations, customer requirements, architecture, V&V, hazards, …) directly in a tool specific way.

Attributes Standardization [I]

Standardizing the attributes (and their types) is the ‘sine qua non’ on the way to efficiency, re-use, error/problem minimization, …

For the different types of objects in the different types of specifications, you want to standardize which properties / characteristics / … you manage and which values (and potentially transitions) you allow.

Carefully considering potentials of inheritance from more generic to more specific types of objects (e.g., from a general requirement to a technical requirement on component level) gives you additional advantages.

Examples of attributes (i.e. properties / characteristics / … to be managed) are:

1. Object Type
2. Applicability
3. Priority
4. Risk Level
5. Comment
6. Qualification Level
7. V&V Measures
8. Satisfaction Argument
9. Rationales
10. Architecture Decomposition
11. V&V Prerequisite
12. V&V Action

Attributes Standardization [II]

Specify and thus document the attributes (and their types).

Include information about naming, applicable level(s), applicable types of specifications, purpose, usage, ...

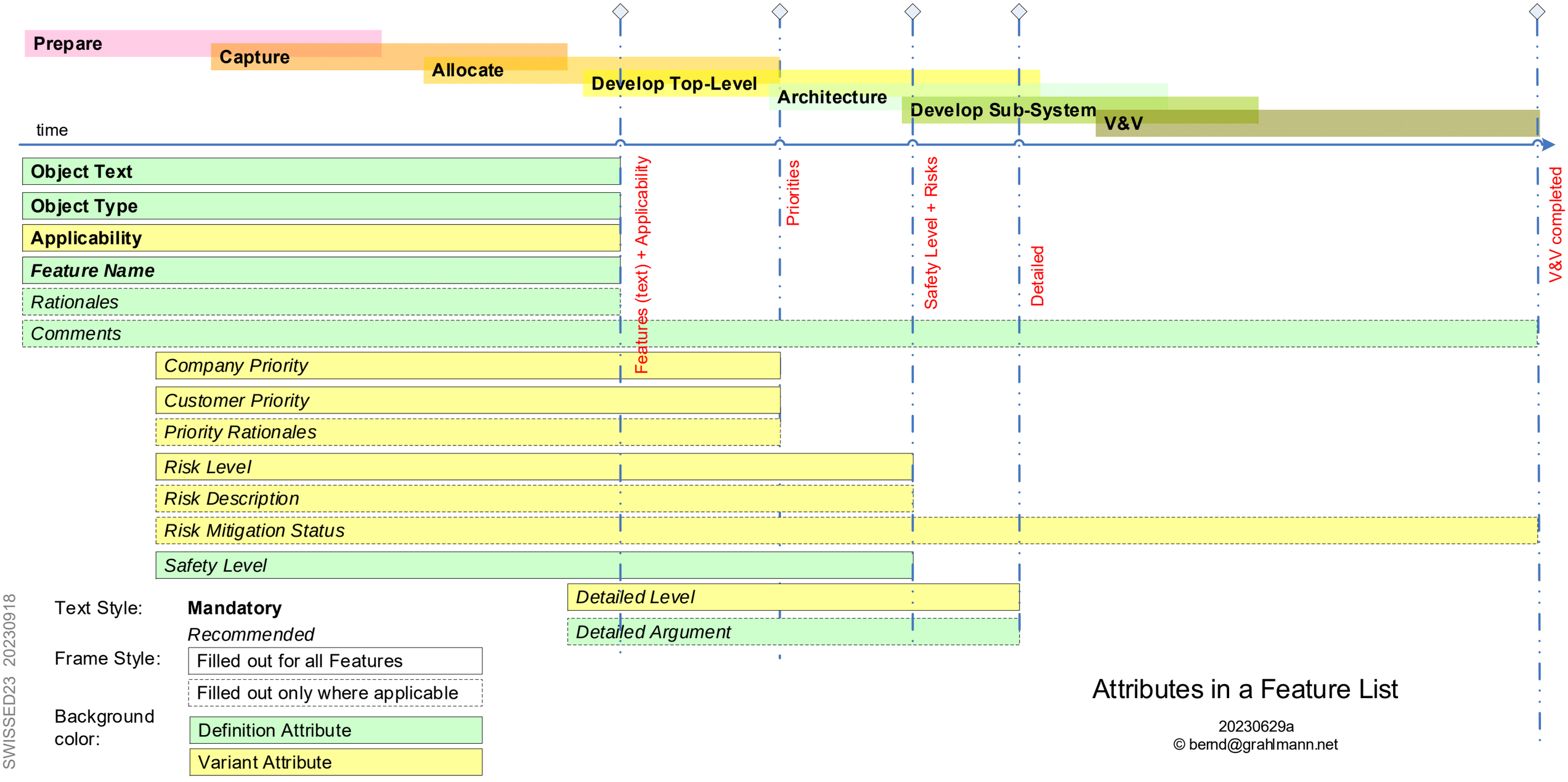
Best case, implement a way to setup and administer the attributes (and their types) automatically in the tooling from that specification.

Here is an example of how attributes and types can be specified and thus documented in / for IBM Rational DOORS:

ID	Doors Attribute Name	Module / Object Level?	Defintion / Variant specific?	Applicable Module Types	Presence of value	Presence Of Attribute	(c) bernd@grahlmann.net - iDARM Attributes & Types Documentation	Attribute Purpose	Type Details	Attribute Usage	DOORS Attribute Type Name
Attributes .DOC-13							**2 DOORS Attributes**				
Attributes .DOC-205							**2.2 Attributes for Requirements Specifications**				
Attributes .DOC-446							**2.2.3.2 'Qualification' related Attributes**				
Attributes .DOC-237	a_<VariantName>_ QualificationLevel	Object	Var	OSTD OCRS OINT TRS	Only when applicable TRS-Mandatory xRS-Mandatory	TRS-Mandatory xRS-Mandatory	**Qualification Level**	Specifies for a variant whether sufficient V&V activities (in particular, test cases) - to test the realisation of the requirement on this scope - have been written and linked to the requirement.	[Enumeration]: TBD N/A Not Qualified Partially Qualified Qualified	Note, that 'Qualification' is meant as being more general and covering Validation and Verification and Testing ... no matter which definition one uses (note that hardware guys often define verification/validation differently from software, ... guys). In 'Qualification Level' one important status of requirements is tracked - whether there are sufficient V&V activities (in particular, test cases) - to test the realisation of the requirement on this scope - have been written and linked to the requirement (such that 'passing' those V&V activities implies that the requirement has been sufficiently validated and verified). ...	aType_QualificationLevels

Attributes Standardization [III]

Specify and visualize expectations wrt. evolvement of attributes over time by mapping to phases and gates/milestones of your systems engineering process. Here is an example of how this can be done:



SWISSED23 20230918

Version/Variant Management [I]

No matter what (systems of) systems you develop, most often your challenge to do the systems engineering in an efficient, effective, … way becomes extremely more complex, complicated, … because you actually want to develop multiple versions/variants.

In addition to the usual challenge to properly manage changes and configurations of all your specifications (with all their requirements, architecture elements, V&V cases, V&V steps, …) one or even more dimensions are added if versions/variants of your systems of systems, systems, sub-systems, components, … require proper distinct management.

Different versions/variants imply that you need to manage (at least) certain properties / characteristics / … of all your requirements, architecture elements, V&V cases, V&V steps, … per version/variant.

Typical examples of such version/variant specific attributes are:

1. Applicability
2. Customer Priority
3. Risk Level
4. Qualification Level
5. V&V Measures
6. Realization Level

.

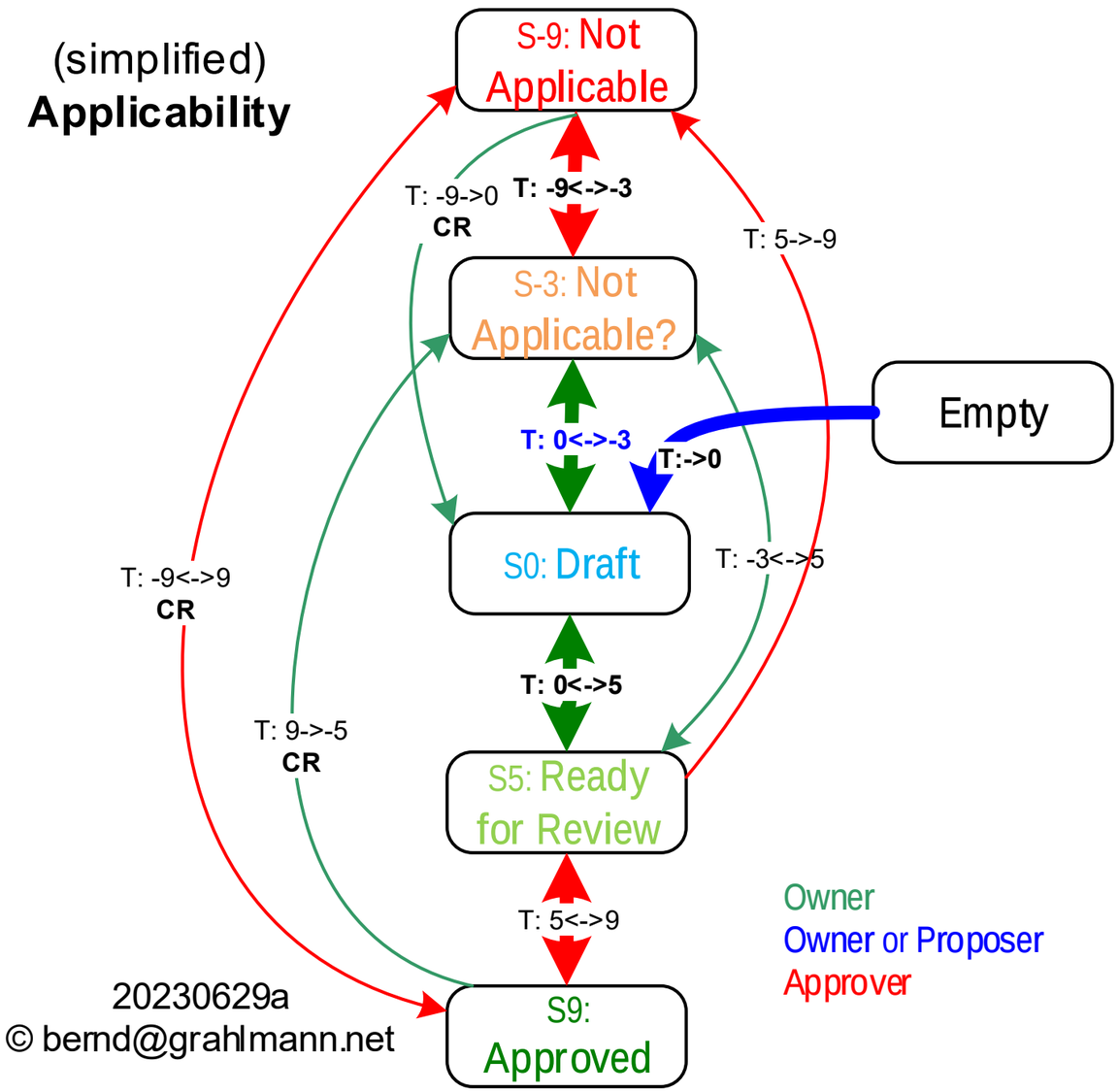
Version/Variant Management [II]

Manage the main ‘status’ of a feature, requirement, architecture element, V&V case, V&V step, … via applicability per variant / version:

- Applicability (per product version/variant)
  (whether or not the requirement or feature or architecture element or V&V case or … is applicable for the product version/variant)
  e.g., [{Not Applicable, Not Applicable?, Draft, Ready for Review, Approved}]

Typical ‚work-flows‘ are:

1. Draft -> Ready for Review -> Approved;
2. Draft -> Not Applicable? -> Not Applicable



Version/Variant Management [III]

Focus on version/variant management from the real beginning, don't postpone this - thinking that you can simply add this later.

Make sure that your solution covers all phases and disciplines of your systems engineering process and all specifications (with all their types of objects – i.e., features, requirements, architecture elements, V&V cases, …).

Ensure a powerful basis for efficient re-usability of features, requirements, architecture elements, V&V cases, … (AND their traceability!) across projects and product families (avoiding duplications, … as far as possible).

Cover, in particular:

1. Version/variant specific traceability (visualization);
2. Version/variant specific document generation;
3. Version/variant specific filtering;
4. Version/variant specific comparison;
5. 'Frozen version' creation incl. all version/variant specific 'information';
6. Error-proneness;
7. Performance.

Version/Variant Management - SSSE/INCOSE - Selected Event [IV]

If you want to hear more about efficient and effective versions/variants management and re-use …, an SSSE/INCOSE - Selected Event is planned on Tuesday October 31st, 2023 evening in Zürich:

- *Efficient and Effective Versions/Variants Management and Re-Use: Starting 'properly' with Requirements and Goals*

Questions & Answers

Contact me via email: Bernd@Grahlmann.net or phone +41 792967651

or check via https://www.grahlmann.net/doors_requirements_management_training_overview.htm
or LinkedIn: https://www.linkedin.com/in/grahlmanndoorstelelogic/
or Xing: https://www.xing.com/profile/Bernd_Grahlmann/

or join ‚my' LinkedIn groups:

- ‚Requirements Engineering Tools' https://www.linkedin.com/groups/12821233/
- ‚IBM Rational DOORS and DOORS Next Generation - DNG (ex Telelogic DOORS) User Group' https://www.linkedin.com/groups/769057/
- ‚Siemens Polarion' https://www.linkedin.com/groups/12004818/

Thanks a lot ☺