

OSLCFest 2021

Navigating Versions in Configuration Management

Robert Baillargeon
rbailargeon@sodiuswillert.com

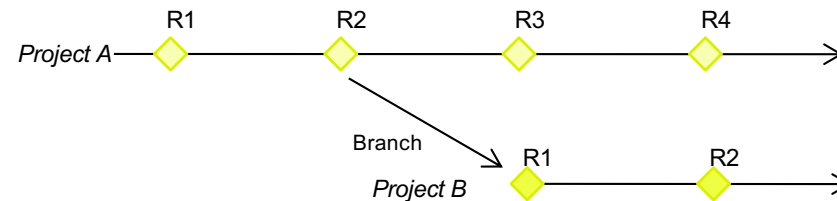


What we value in our products

- **Engineers working where they are productive**
 - Using tools of choice
 - Working simply and effectively
- **Tools that are flexible to the engineering needs**
 - Supporting Standards
 - Configurable to your workflow
- **Tools that work in the enterprise**
 - Secure systems of record (with no copies)
 - User authenticated access to data
 - Server-side integration to support deployment, support, availability

(Enterprise) Configurations

Configuration in a Domain



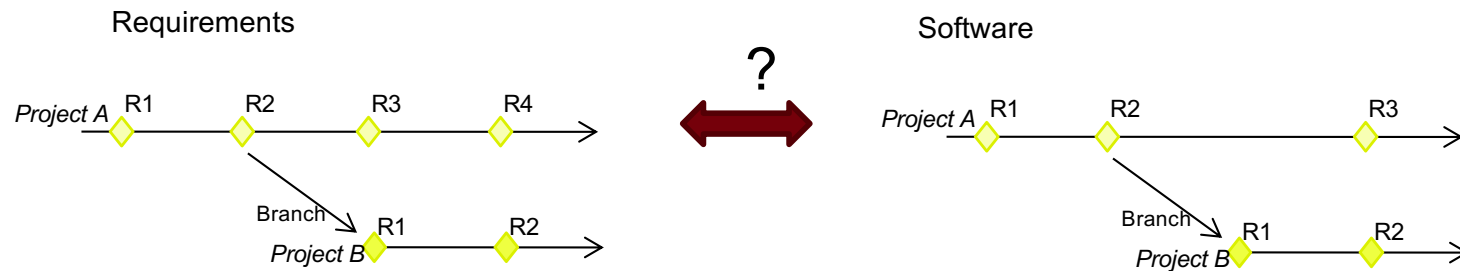
Domain configurations are logical. Streams branches of versioned artifacts centered around a project.

Most tools natively have this support.

And those that don't we usually leverage a version control system to version the files.



Configuration in a Domain



Challenges arise not as much on a single domain, but rather the combination of multiple domains.

Which version relates?

When do I update?

How do I link resources?

How do I understand change has occurred?



What is an Enterprise Configuration?

- It is a temporal context
- It is a purpose
- It is a scope
- It is a collection of artifacts
- It is a singular in versions of any particular artifact
- It spans repositories

Things that demand an Enterprise Configuration

- A product release
 - HW Revisions
 - SW Revisions
 - Approvals
 - Tooling
 - ...
- A gate review
 - Documents
 - Test Results
 - Risk Records
 - ...

Implementing Enterprise Configurations in Tools

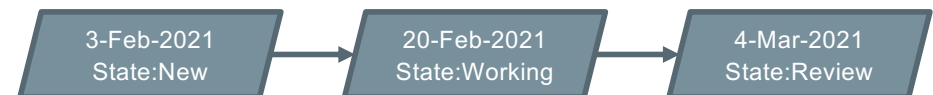
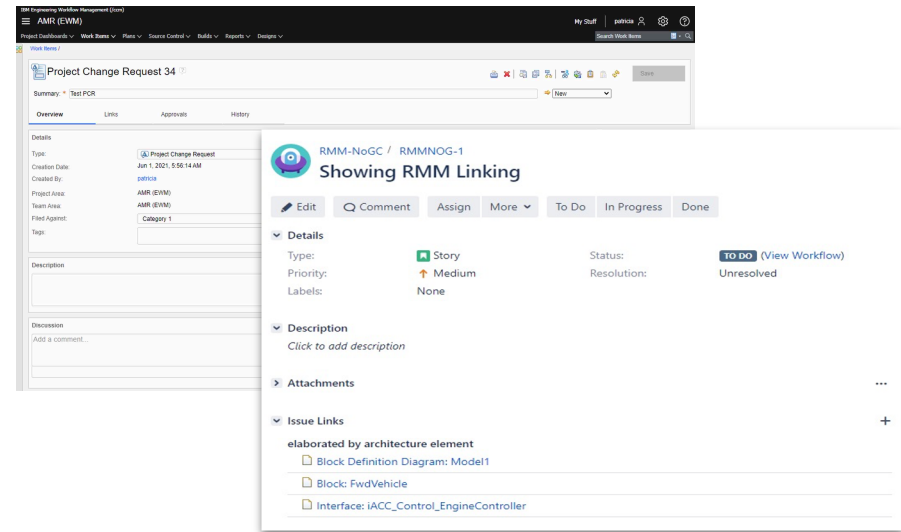
- We want configurations to be natural parts of our tools
 - They need to focus on the natural language of the current tool
 - *But provide a context of how they operate in the enterprise*
 - They must be intuitive to how teams use tools
- Simple Examples navigating the Change – Requirements Boundary
 - Moving a Story to a new Release should use the requirement versions for the release
 - Temporal nature of Defects.
 - *“Found in” describes a past version of where an issue was detected.*
 - *“Fixed in” describes a future version where it is fixed*

Example of Configurations in the Enterprise (Jira and DNG)

Enterprise Artifacts

The Workflow Artifacts in the Digital Thread

- **Workflow Artifacts**
 - **States and Sequences**
 - *They have an outcome*
 - **Examples**
 - *Change Request*
 - *Reviews*
 - *Tasks*
 - **History (but no Versions)**
 - *Time is our index*
 - **No Branching**
 - *Hierarchies common*
 - *Copies/Clones common*
 - **Owned by Projects (Teams)**



The Asset Artifacts in the Digital Thread

- **Asset Artifacts**

- **Entities**

- **Examples**

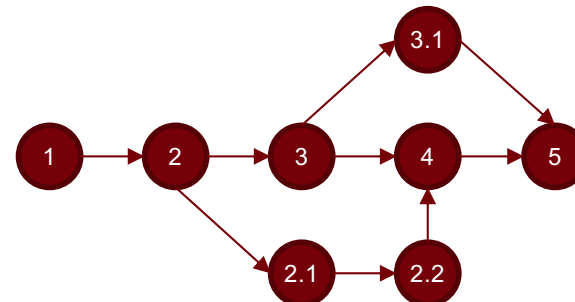
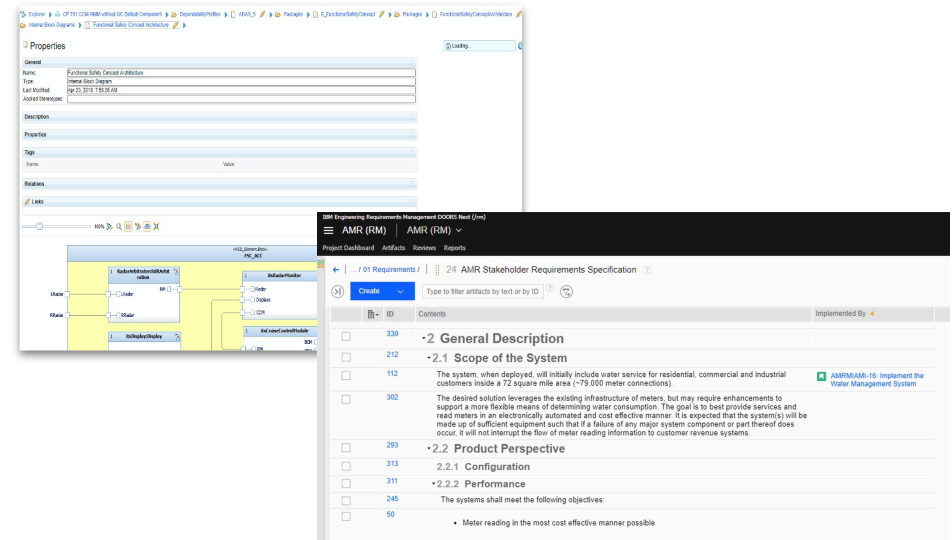
- Requirements
 - Test Cases
 - Model Elements
 - Code

- **Versions**

- Each version can be used in multiple contexts (reused)
 - Relate in branch and merges

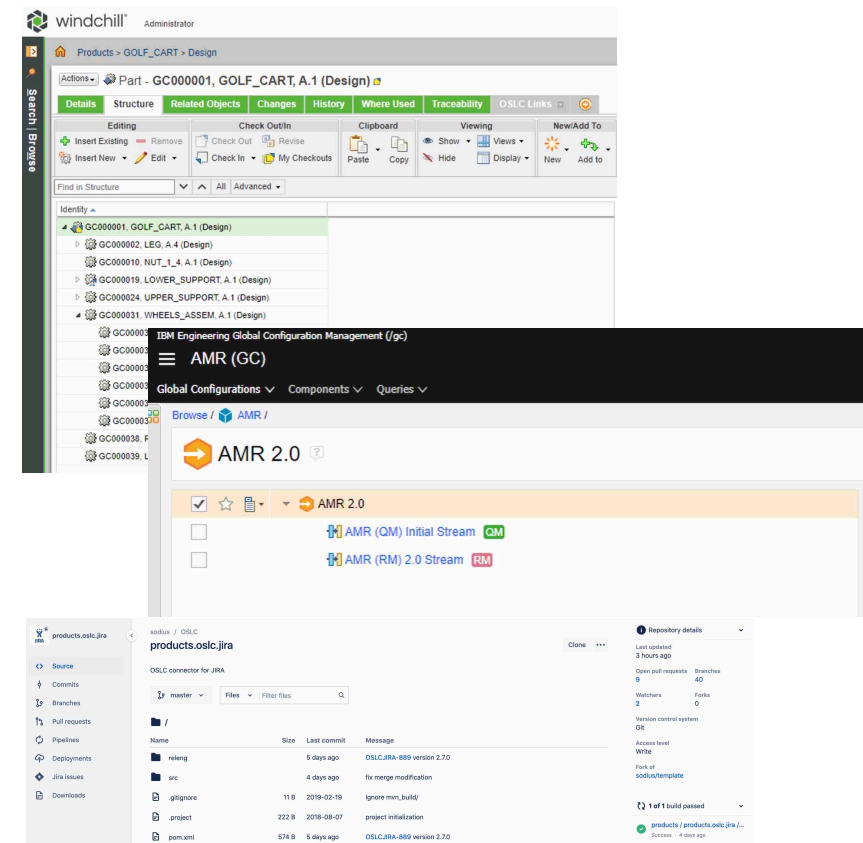
- **Owned by Collections**

- Instances in Streams and Baselines



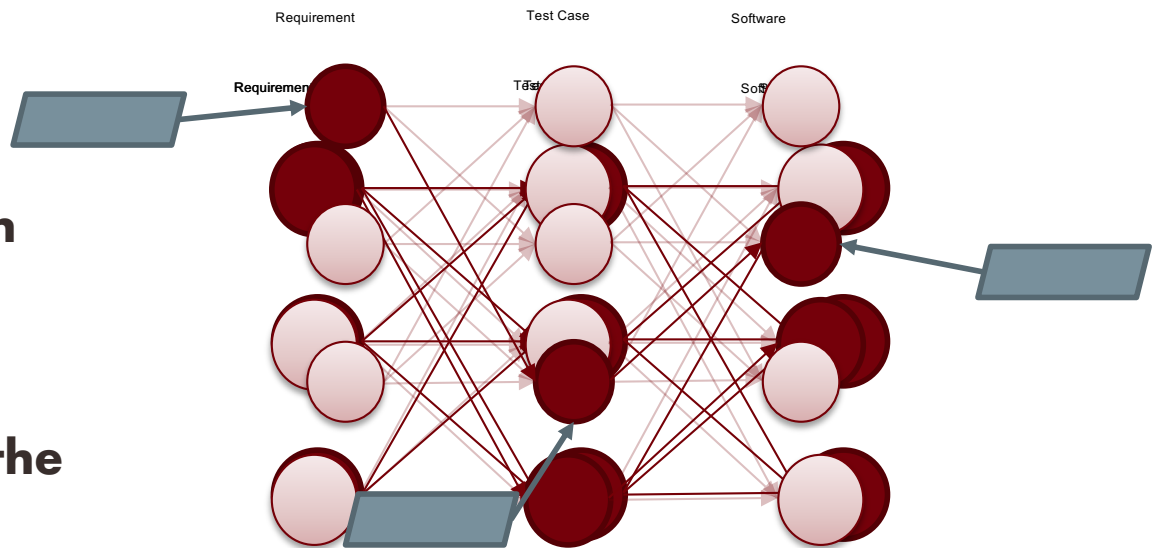
The Configuration in the Digital Thread

- **Asset Configurations**
 - **Instances of a Collection/Container**
 - **Examples**
 - *Git Branch*
 - *Released Requirements Document*
 - *Bill of Materials with Effectivity*
 - **Asset State**
 - *Baseline (static) or Stream (dynamic)*
 - **Scoped**
 - *Local -> Single Repository*
 - *Enterprise/Global -> Cross Repository*
- **Composite of Local Configurations**



Digital Twin from Web to Thread to Process

- Your digital assets are a web of relationships and versions
- An instance (configuration of artifacts) is a Digital Thread
- Creation of the thread is the application of your process with workflows
 - Creating artifacts
 - Assembling artifacts
 - Configuring artifacts



OSLC and Configurations

OSLC and Configurations

- OSLC Provides a Foundation for Enterprise Configuration Management
- Uniquely Identifiable OSLC Elements
 - Component – Unit of Configuration
 - Concept Resource – Unique Resource Independent of Version
 - Configuration – Set of versioned concept resources containing only one version of a resource
 - *Baseline – Static*
 - *Stream – Modifiable*
 - Local Configuration – Single Component
 - Global Configuration – Aggregate of Local Configurations and Composable Configurations



In Common Tools

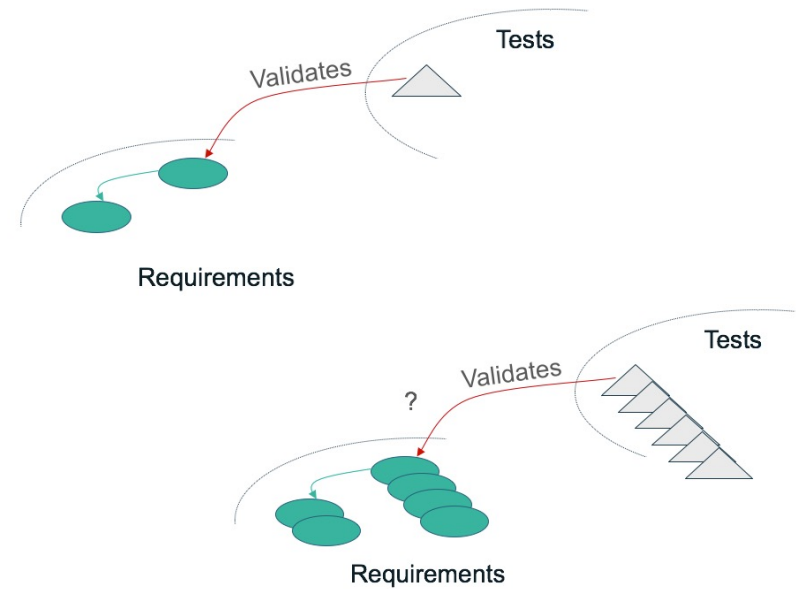
- Similar Concepts Exist in Most Repositories
 - Baselines
 - Change Sets
 - Branches
 - Bills of Materials
 - Document Bundles
- OSLC Focuses on the essential elements
 - Separation of a Resource from it's Version
 - Composability for Reuse
 - Demands for stable & unique identifiers
- OSLC Becomes an Overlay to Address the uniformity of the Enterprise

Links and Configurations

The Linking in the Digital Thread

- **Links**

- **Have a role (relationship)**
- **Are an attribute (written & deleted)**
- **Are owned by an endpoint (artifact)**
 - *Ownership is by the elaborating or referencing artifact*
- **Point to the basic artifact (unversioned)**
 - *Version is resolved by a configuration (context)*



Links in a Configuration Managed World

- **Basics from the Standard**
 - Links are owned (stored on a single artifact)
 - Links have a role (association type)
 - Links have a directionality (point to an artifact (source to target))
 - Links are contextualized to version with configuration information (Target GC)
- **What is stored in a link**
 - An association type (Implements Requirement)
 - An identifier/short title (519)
 - A title (My favorite requirement)
 - A resource URL (https://elm/rm/resources/BI_BkaxHJtwEeqNGNQYj3xGng)
- **What is unique**
 - No copies of content
 - No storage of the backlink
 - Ownership is based on type, not the location of creation



Ownership of links

- Ownership is based on the link type and the artifact types
- This is driven by the standard itself, and reflected in tools supporting ConfigManagement
- Workflow objects always store the link
- Versioned artifacts store by rule through the Software/Systems Lifecycle
- The links don't store the version
 - The version of the artifact and the link target is driven by your current context

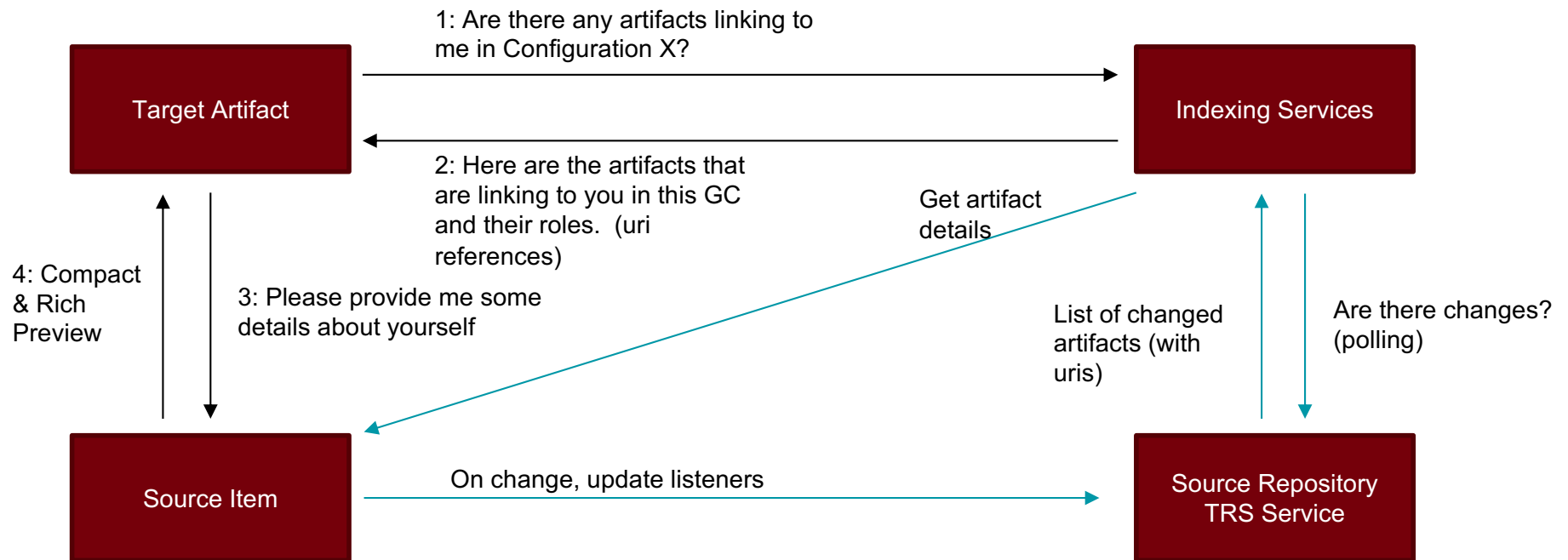
Link Relationship	"Owner" and how link displays	Target and how link displays
Change Management (CCM) - Design Management (DM) link types		
oslc_cm.relatedArchitecturalElement	CCM: Elaborated By Architectural Element	DM: Elaborates
Change Management (CCM) - Quality Management (QM) link types		
oslc_cm.affectsTestResult	CCM: Affects Test Result	QM: Affected By
oslc_cm.blocksTestExecutionRecord	CCM: Blocks Test Execution	QM: Blocked By
oslc_cm.relatedTestPlan oslc_cm.relatedTestCase oslc_cm.relatedTestScript oslc_cm.relatedTestExecutionRecord	CCM: Related	QM: Related
oslc_cm.testedByTestCase	CCM: Tested By Test Case	QM: Development Item
Change Management (CCM) - Requirements Management (RM) link types		
oslc_cm.affectsRequirement	CCM: Affects Requirement	RM: Affected By
oslc_cm.implementsRequirement	CCM: Implements Requirement	RM: Implemented By
oslc_cm.tracksRequirementTracks	CCM: Tracks Requirement	RM: Tracked By
oslc_cm.tracksChangeSet	CCM: Tracks Change Set	RM: Change Set Link
Design Management (DM) - Requirements Management (RM) link types		
dm.derives	DM: Derives From	RM: Derives Architectural Element
dm.refine	DM: Refines	RM: Refined by Architectural Element
dm.satisfy	DM: Satisfies	RM: Satisfied by Architectural Element
dm.trace	DM: Traces	RM: Traced By Architectural Element
Quality Management (QM) - Design Management (DM) link types		
oslc_qm.validatesRequirement	QM: Validates Requirement	DM: Validated By
Quality Management (QM) - Requirements Management (RM) link types		
oslc_qm.validatesRequirement	QM: Validates Requirement	RM: Validated By
oslc_qm.validatesRequirementCollection	QM: Validates Requirement Set	RM: Validated By
Requirements Management (RM) - Requirements Management (RM) link types		
oslc_rm.constrains	RM: Constrains	RM: Constrained By
oslc_rm.decomposes	RM: Decomposes	RM: Decomposed By
oslc_rm.elaborates	RM: Elaborates	RM: Elaborated By
oslc_rm.satisfies	RM: Satisfies	RM: Satisfied By
oslc_rm.specifies	RM: Specifies	RM: Specified By

Note: As a general rule of thumb, CCM owns all link types involving the CCM application, QM owns all link types involving QM, other than those owned by CCM. DM owns all links from DM to RM. RM only owns links within and across RM project areas.

<https://jazz.net/wiki/bin/view/Deployment/IntegratingWithConfigurationManagementEnabledCLMApplications>



Resolution of Backlinks on a Target



Example of Configurations in the Enterprise (Link Discovery)

Managing (Enterprise) Configurations

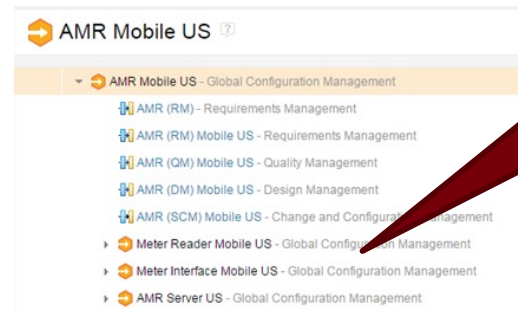
Enterprise Configurations

- Enterprise Configurations

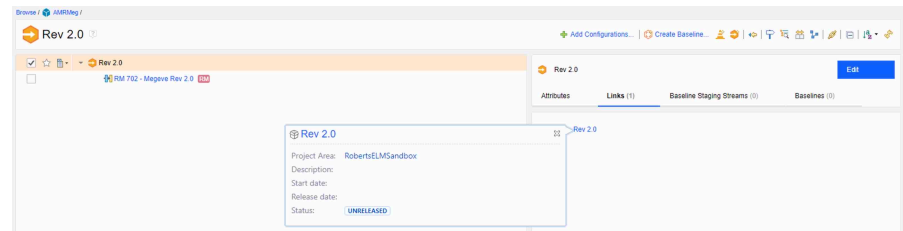
- Are cross repository
- Are hierarchical
- Are deterministic
- Are composable
- Do have a purpose
- Require tooling

- Enterprise Configurations

- Enable unifying silos
- Eliminate conflict & copies
- Preserve consistent work
- Enable reuse



Other repositories
need to contribute to
the Enterprise
Configuration Manager



Enterprise Configurations Are the Future of the Digital Thread

- In The Large
 - Product Releases
 - Production Gate Reviews
 - Safety Cases
 - PLE Practices
 - ALM-PLM Unifications
- In The Small
 - Subsystem Development
 - Gate Reviews
 - Asset Reviews

Process Orchestration in the Enterprise

Thank You



SODIUS SAS

34 Boulevard du Maréchal A. Juin
44100 Nantes
+33 (0)228 236 060

SODIUS CORP

418 N. Main Street 2nd Floor
Royal Oak, MI 48067
+1 (248) 270-2950

WILLERT SOFTWARE TOOLS

GmbH

Hannoversche Str. 21,
31675 Bückeburg
+49 5722 9678 60

For more information visit sodiuswillert.com