PTC®

Content Sharing and Reuse in PTC Integrity Lifecycle Manager

Author: **Scott Milton**

PTC®

Table of Contents

1. Abstract

This document details the various mechanisms that provide content "reuse" in PTC Integrity Lifecycle Manager. For the purposes of this document, content is defined as the individual line items in documents, such as requirements and test cases, and reuse is defined as the ability to use the same requirement in multiple documents. Both real examples and instructions on how to configure and use various Lifecycle Manager features are described and recommendations for implementation are made. The Document Model, Parameters and Trace Propagation are also briefly discussed.

2. Introduction

Generally, a set of requirements changes minimally from one release of a product to the next. If the changes are a small portion of the total requirements, a good requirements management solution makes it easy not only to introduce new requirements, but also to maintain and reuse the majority of unchanged requirements. A requirement that was introduced in one version or release of a product must continue in all subsequent versions until it is either modified or dropped. Each intervening version or release must be able to account for how the requirement was satisfied. In addition to reusing requirements from one release to the next, it is often desirable to reuse requirements by sharing them across projects that are distinct, but that share a common set of requirements. Non-functional requirements such as those detailing regulatory and compliance are a typical example of this. Reusing requirements allows the user to leverage the work products and activities of similar projects.

Note: A brief update on naming. The product formerly known as "PTC Integrity" is now named "PTC Integrity Lifecycle Manager", since PTC Integrity now refers to a family of software and systems engineering products. For brevity and clarity, this document uses "Lifecycle Manager" as an abbreviation for the full name, "PTC Integrity Lifecycle Manager".

Lifecycle Manager allows reuse of requirements by copying one or more individual requirements, or by selecting the root document and copying the entire hierarchy of requirements. However, this copying is not limited to a simple "clone and own" procedure which would lead to the newly copied version of the requirement becoming out of date when the original requirement is updated. By using Lifecycle Manager, the newly copied Requirements will reference the original versions in a variety of user defined modes which will either automate or facilitate updates. Where the Requirement came from, in other words its history or heritage, will remain readily available. The system manages the underlying items and artifacts to appropriately reuse and share those items between documents and projects; as a result, unnecessary duplication in the repository is prevented while providing revision history and usage information. It is also possible to create requirements that are specifically designed for reuse and make use of a publish-subscribe model to facilitate sharing.

Note: The rest of this document assumes you have at least a passing familiarity with Lifecycle Manager. For more information about integrity, please visit here. To schedule additional training for your location, please visit here.

3. Document Model

Before proceeding further, it is necessary to briefly describe a particular feature available in Lifecycle Manager, the Document Model. In Lifecycle Manager Terms, a document is a collection of related items. The document model describes the various types of items used by a document and the particular roles they play to form a cohesive unit. The Document view is the interface used to view and modify both documents, and their content, in a hierarchical tree structure. Documents are typically used in the Requirements Management and Test Management domains, although numerous other domains of analysis are supported by the Lifecycle Manager document model.

Lifecycle Manager uses specific terms to describe the component items of a document. Specifically:

- The Document item (or segment) is the holder of the document's metadata, lifecycle and container for all its content;
- The Content item (or nodes) is any line item within a document, such as individual requirements or test cases, as well as any sub-segments (i.e., included documents). Content items typically have both significant and non-significant fields, as defined by an administrator. Some fields are more meaningful than others and significant fields are those that would generally be the same on all instances of the content when it is reused. It would be expected that non-significant fields would often be set separately on each instance of the reused content and typically are used for metadata.

Note: There is an additional role to the two described above called the Shared Item. It is used "under the covers" to represent shared content in a document. Since the item that plays this role is transparent to end users during the creation or modification of content, it is not described in any detail this document. For administrators who require more information about the shared item, please refer to the Lifecycle Manager Administrator's guide.

Similar to other Lifecycle Manager items, relationships are the "glue" that ties document and content items together and also allows for reuse and persistence of artifacts over time. Within the document model there is a special subset of relationships called Trace Relationships, or Traces, that link Content Items to other Content Items, e.g., Requirements "Decompose To" other Requirements, or Requirements are "Verified By" Test Cases.

To learn about the relationships inherent in the document model, see “Relationships in the Document Model” of the Lifecycle Manager Administrator's guide or the "Traceability in Lifecycle Manager.docx" file provided.

3.1. Reference Modes

All content within a Lifecycle Manager document has specific permissions and access levels associated with it. In addition, reference modes dictate how the system behaves when content is created, copied and modified. When the reference modes available are:

- **Author**: The Author reference mode controls the evolution of the content item, i.e., it is the original or principal source of the item.
- **Reuse**: The Reuse reference mode allows users to reuse the content by exposing a fixed revision of the content item in one or more additional locations. If a user has the appropriate permissions and attempts a "significant edit" to an item whose reference mode is set to reuse, “branching" occurs behind the scenes, leaving the original content source unchanged. This effectively allows a user to share the content, up until the content is significantly modified in the new location. A significant edit is a modification made to one or more significant fields on the content item, as defined by the administrator. This concept allows for some fields to be edited in each of the reused copies without causing the branching to occur.

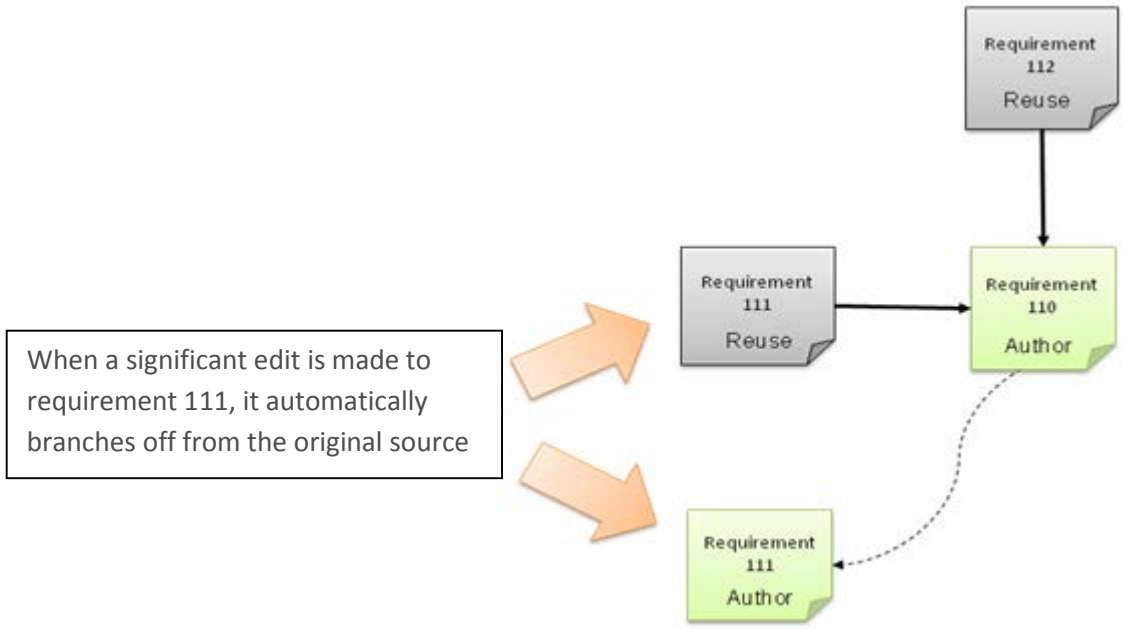

Fig A. Significant edit of a Reused Requirement

- **Share**: The Share reference mode allows users to share content by exposing the current version of the content item in an additional location. The shared item is not editable and it automatically receives any changes made by the node from which it was shared from. i.e., Users will see a read only copy of the content that always reflects the most up to date version of the content.

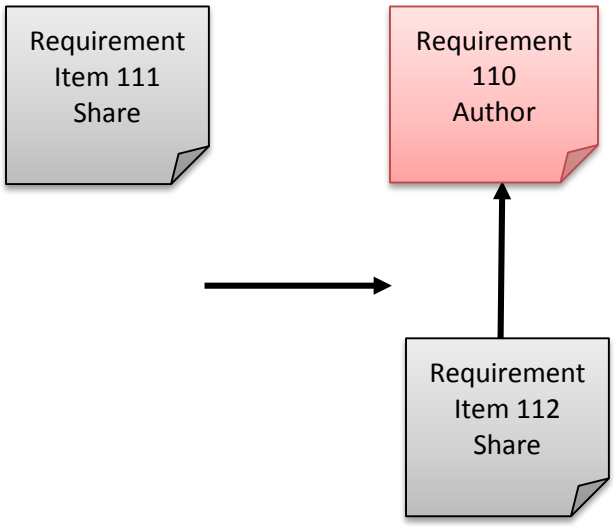

Fig B. Sharing of a Requirement

The user may change the reference mode on content by selecting the content and clicking on the Content > Toggle Share/Reuse menu option, and choosing between Reuse or Share. A user can change a reference mode from Share to Reuse and from Reuse to Share; however, an Author reference mode cannot be changed.

Note: If the reference mode of a content item is currently set to Reuse and the original content is modified, the reused content's reference mode can no longer be toggled to Share. If needed, the original content can be recopied and the new copy set to Share.

Further details about specific sharing scenarios, such as what happens when a user attempts to share content that is already being reused can be found in the Knowledge Base article entitled "Shares and Shared By relationship fields added in 2009" located here.

4. Reusing Content Items

To copy content within and between documents for the purposes of reuse/sharing.

- Open the applicable document in the Document View.
- Select a row or section in the Content panel of the document the user wants to copy from and select Content > Copy on the menu. The selected row or section is highlighted. Selecting a node to copy from the Content panel results in just the selected items being copied. Selecting a node to copy from the Outline panel results in all items in the node's structural relationship list (i.e., its children) also being copied.

Note: If desired, the user can configure their client to always or never copy child items by opening the document view, going to the View >Options menu and toggling the Operations mode to Section or Content respectively.

- Select an insertion point in the target document and select Content > Paste Special to paste row
- On the resulting Paste Special dialog:
- The user can choose the resulting reference mode of the newly copied content item(s); share, reuse, or just a regular, disconnected copy.
- The "Include Traces" checkbox allows you to specify to carry over all traces associated with the children of the copied content.

Fig C. Paste Special Dialog

Note: If a user is performing a simple Ctrl+C and Ctrl+V to copy and paste a content item, just like in standard Windows applications, within the same document boundary, a standard copy with no reuse/history will be performed. If the user performs the same keyboard sequence, but the target and source documents are not the same, the system will perform a “paste special” and the reference mode of the target item will be chosen depending on the "Default Reference Mode" set by the administrator on the Document Type definition the user is pasting into. The user can then toggle the reference mode as needed using the standard menu option of Content > Toggle Share/Reuse.

5. Reusing Entire Documents

Lifecycle Manager allows you to create nested documents which are subdocuments inside of documents. This allows users to partition content for logical, semantic, or performance reasons.

5.1. Differences between Included and Referenced Documents

Nested documents can be either inserted (as a reference) or included in other documents:

- Insert (reference): When a subdocument is inserted into a parent document, only the reference to the subdocument is exposed. You must open a subdocument in order to manage its contents.
- Include: When a subdocument is included into the parent, the entire contents of the subdocument are exposed as if they were a sequential part of the parent. However all editability rules are governed by the included document. e.g., you cannot make the contents of a “published” subdocument editable by including it in an open document.

In a component-based environment, where an application or product is made up of smaller, more purpose-built components managed by separate teams, the inserted option is common. This allows the parent document to reference the content, but it does not require direct management of the content. Inserted documents are not exposed as part of the parent document.

After you insert or include a subdocument into a document, a user can toggle the reference to either see the entire document expanded as a sequential part of the parent, or see only the reference, or link, to the subdocument in the content panel. This is a convenience feature for the user, but does not affect the document or its content.

5.2. Branching Documents

Branching is a method to reuse existing documents by creating a copy of the original document, rather than inserting it in another parent document. However, much like copying content items, this is not a simple "clone and own" copy of the document. All the content in the newly branched copy of the document has a Reference Mode of Reuse. This default value can be set by the administrator in the Document Model settings of the types. Individual content items can also be changed to have a Reference Mode of Share after the branch is completed, if desired.

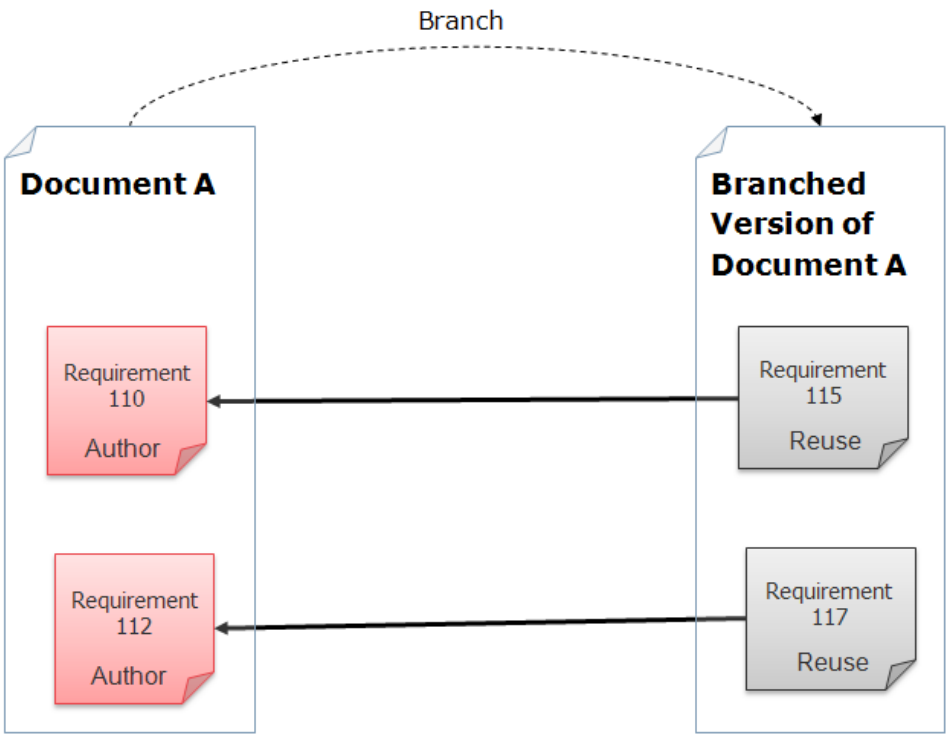

Fig D. Branching a Requirements Document

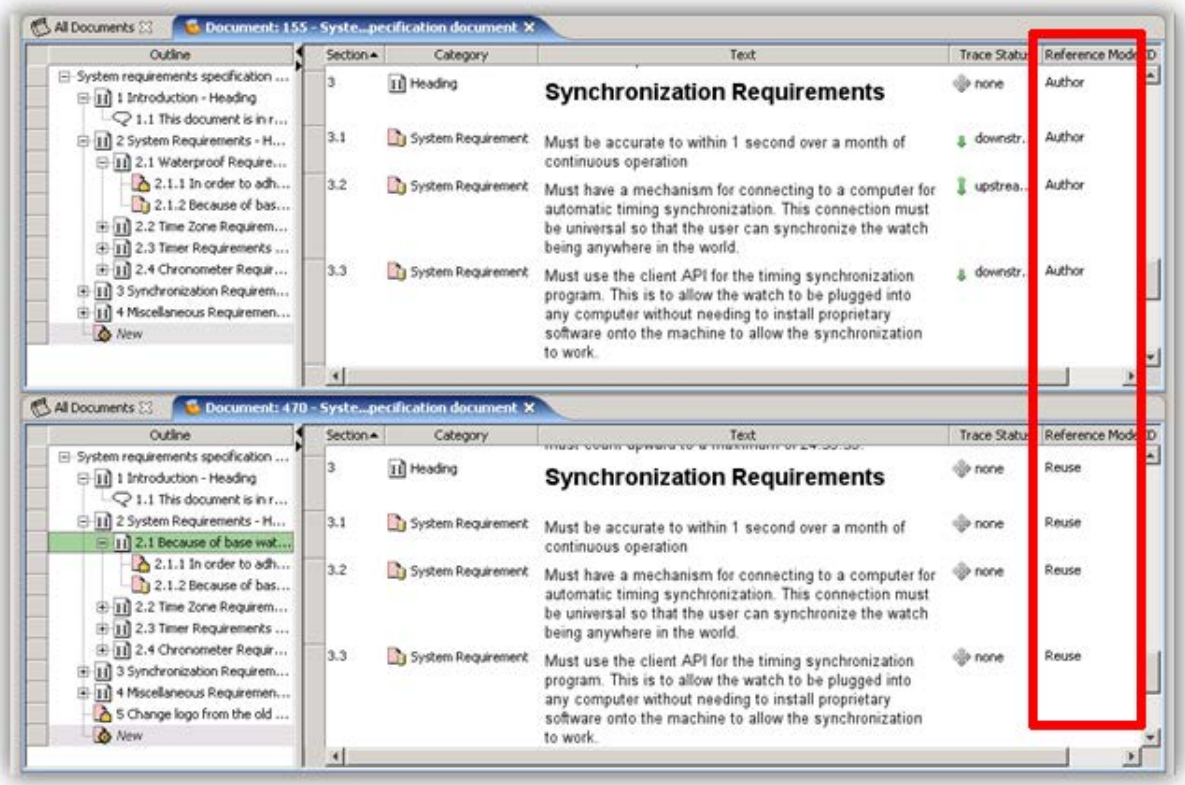

Fig E. Original and Branched Requirements Document

When the user initiates a branch command (**Document > Branch**), the system:

1. Asks for the project that the branch will be created in.
2. Copies the original document item and populates the fields listed in the item type's **Copy Fields** to the new item.
3. Traverses the Contains relationship field and both copies and relates all of the node items contained within the document.
4. Determines the reference mode (Share or Reuse) on the branched document's newly created content items by looking at the **Default Reference Mode** set on the document type.
5. Notes a branch in the branch tab of the original document.
6. Marks the new document as branched from in the history.

5.3. Trace Propagation for Branched Documents

To propagate traces means to copy trace relationships from one item to another. An example of this might be the situation where there is a Requirements Document (RD1) that has various traces to a Test Suite (TS1). When a new Release is being planned, both the Requirements Document and Test Suite are branched, creating a new Requirements Document (RD2) and Test Suite (TS2) respectively. The trace propagation wizard guides the user in copying the original set of traces in the original pair of documents into the newly created pair of branched documents.

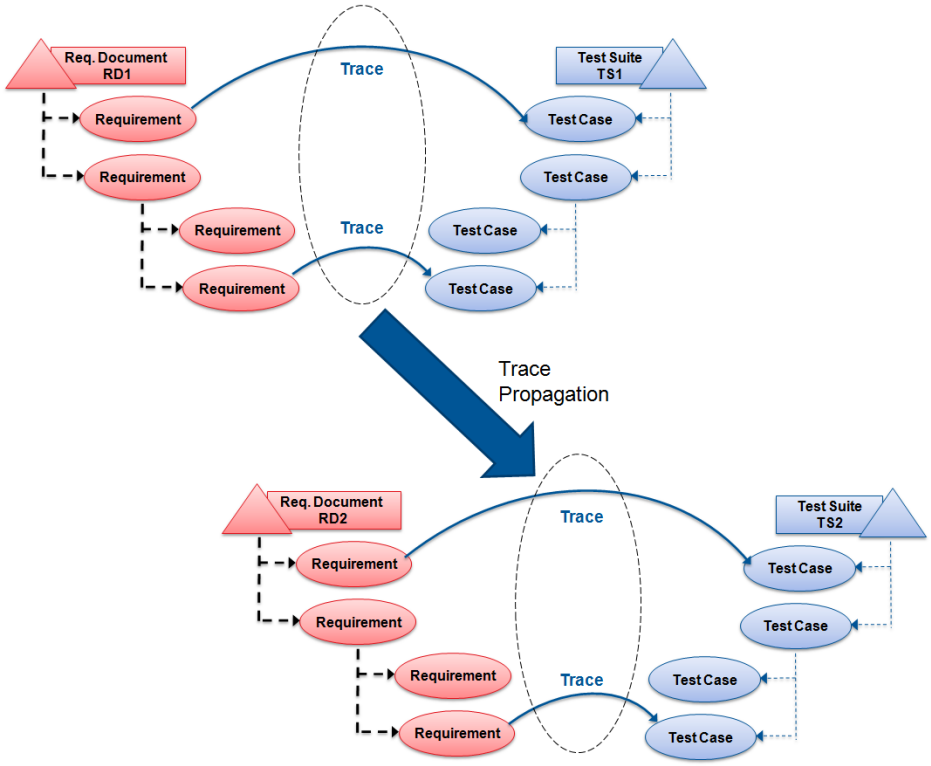

Fig F. Trace Propagation between two pairs of documents

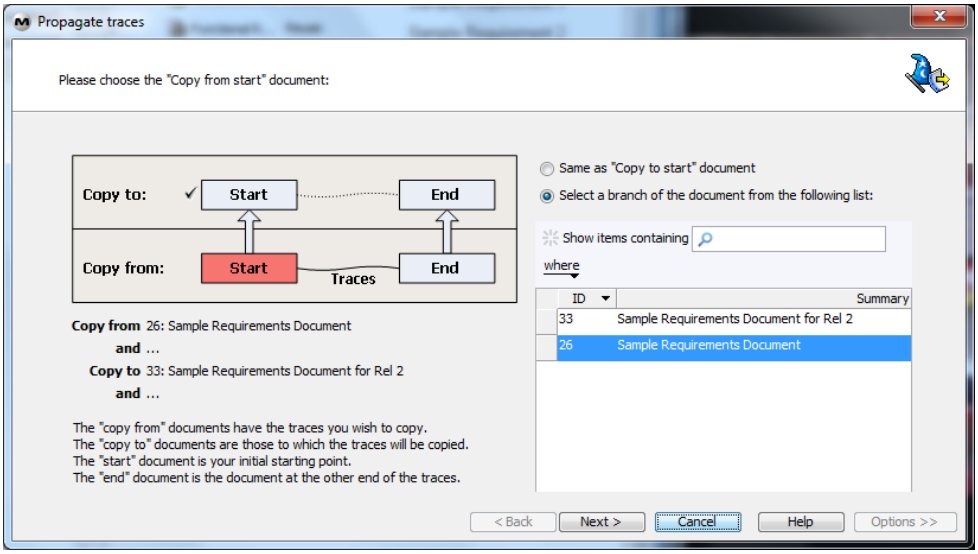

Fig G. Trace Propagation Wizard

Note: The fields that are copied when you branch a document are determined by your administrator. PTC recommends that trace relationship fields are not automatically copied during branching. This enables users to control which trace relationships get copied to the branched document using the propagate traces wizard.

Note: For proper operation of the wizard, PTC also recommends that no content items are allowed to be deleted from the database (i.e., by using the "im deleteissue" command), although the content can certainly be removed or "de-referenced" from documents.

6. Recommended Strategies for Reuse

6.1. Reuse Content Appropriately by Context

Whether content is being reused or shared for the purposes of a new Release of an existing Product or an entirely new Project, perhaps creating a related Product in the same Product Line, some general rules apply:

- Reuse is not for everyone. If content authors are heavily invested in single use word or excel documents, it may be difficult for them to transition to a new paradigm and the benefits may not outweigh the transition efforts. Also, if requirements, or other content, vary dramatically from release to release or from product to product, there may be little value in restructuring for reuse. It may be that only certain common components remain consistent enough over time to warrant reuse.
- Externally created regulatory requirements, especially those needed for compliance to various industry specific laws, are generally a good candidate for reuse. These tend to not change frequently, but it can be of critical importance to an organization that they are always followed and that updates are carefully monitored and enforced.

- Where appropriate, create a Lifecycle Manager document that holds a common set of Base Content, such as Requirements, to facilitate Sharing. This is especially appropriate for a Product Line Scenario where many similar products, i.e., variants, are possible. For each of the variants, reuse an appropriate amount of content from the base set of requirements, and then supplement with the additional variant specific requirements. To easily do this after the base set of Requirements is created, create a new document for each Variant, and then use Copy + Paste Special > Share the new Variant Documents. Each Variant may then author new requirements as needed. See figure below for an example.

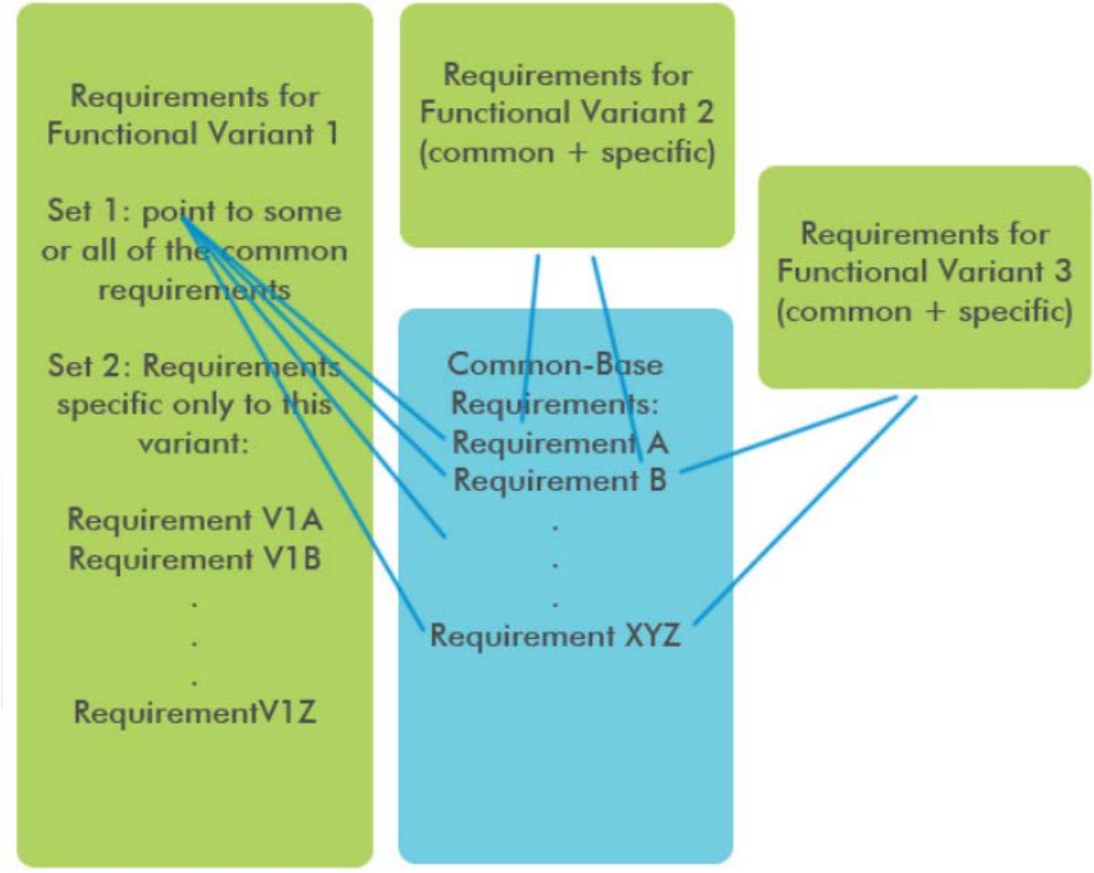

Fig H. Reusing from a base set of requirements

Once the common base of requirements is approved or validated, it need not be re-validated every time a new variant points to it; only the requirements unique to the variant need to be examined and validated. When changes are made to the common base, of course revalidation will need to be performed for all product variants pointing to the changed requirements in the common base, but the effort needed to determine which variants are affected and how, as well as which downstream shared development assets may also need to be recertified for each variant is straightforward and precise because Lifecycle Manager maintains the appropriate relationships between shared assets across the product line.

- If Documents are needed almost verbatim in a new project, such as for a supplemental release, branch the document and make modifications to the individual content items as appropriate. Branching is the most appropriate mechanism in this case, as the content is nearly guaranteed to be the same and each of the branched documents can be easily accessed from the Branches tab on the source document that it was branched from.

6.2. Use Parameters to Write Requirements intended for Reuse (and Sharing)

Parameters are typically used in the Requirements Management and Test Management domains to facilitate reuse or sharing of a content item in different contexts by being able to specify unique item values for each context. For example, a requirement item with parameters that display different values based upon the project the current requirements document is linked to.

Example: A company produces a number of models of widgets in its Product Line. Each widget is tested to different internal pressures, but the pressure requirement is different for each model in the line (e.g., the deluxe model can withstand more pressure for a greater length of time). If the maximum pressure is a parameter value, then the same requirement or test can be used for each model, but the value of the pressure parameter will change based on the model being tested at the time. If there are multiple tests for the different amounts of time the widget should work at the pressure, making the pressure a parameter also ensures that if the pressure requirement for a model changes, all the requirements and tests for that model would use the new value.

Lifecycle Manager determines what parameters can be specified and what parameter values are substituted in text fields based on items that are related to the item being edited or viewed. See Appendix 3. Hierarchy of Related Items Considered When Determining Parameter Values for more details. Parameter substitution is particular appropriate for managing Product Lines, as Products in the same Product Line typically vary from each other by a strict set of easily defined variables. The tests often must follow the same testing process and steps, but the steps just have slightly different values being set or slightly different expected results due to the Product variation.

6.2.1. Defining Parameters and Allowable Parameter Values

There are two portions of parameter configuration, one to be done by administrators and the other by users with appropriate permissions, such as Project or Product Managers.

Administrators must:

- Make the Parameters and/or Parameter Values fields visible on the content item type. These fields will allow users to define the Parameters and/or Parameter Values for the specific instance of the item. The Parameters field essentially allows end users to define a variable. The Parameter Values field allows end users to set a value to an already defined parameter. It is also possible to make FVA fields (i.e., pointer fields) that will retrieve Parameters and Parameter Values from a remote item. E.g., Projects may have FVA fields that retrieve the parameters defined on the related Product item.
- Set a long text field that is visible on the content item type to have a "Substitute Parameters" value of true. Users will be able to refer to the defined parameters in this field.

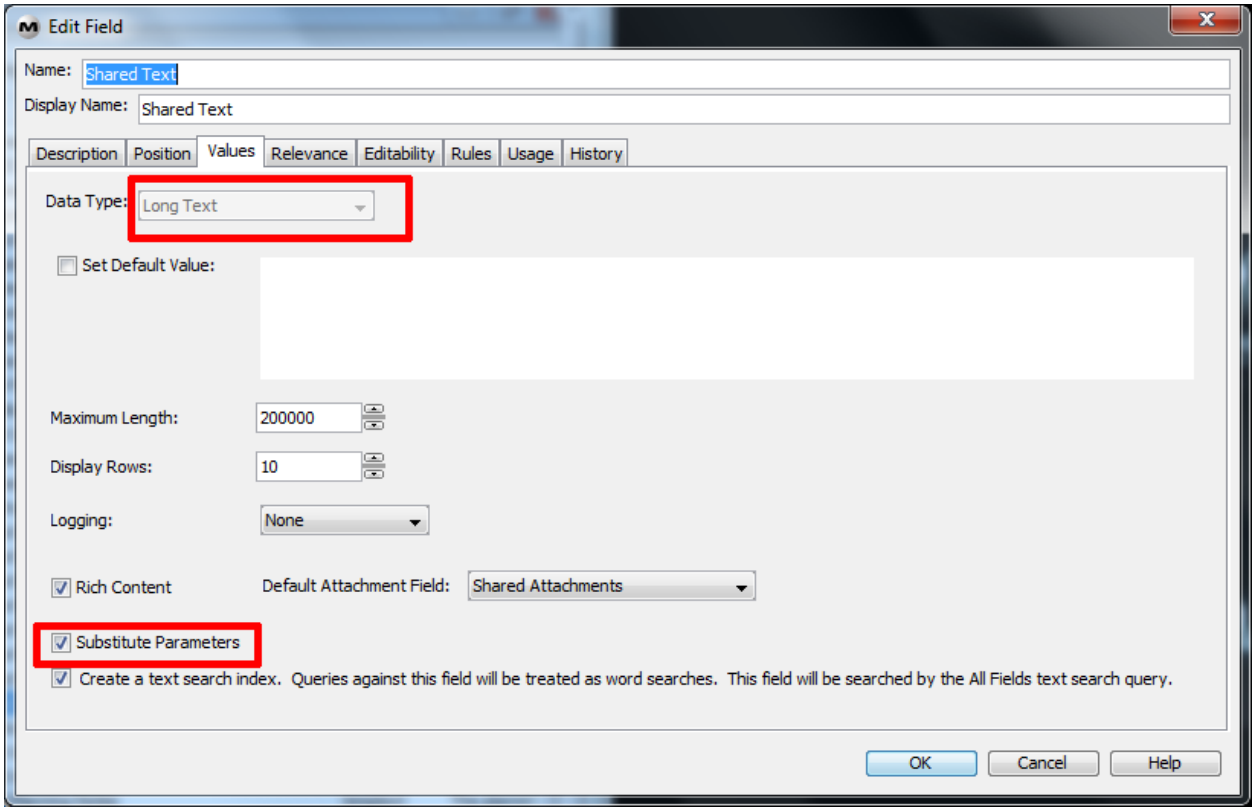

Fig I. Setting Substitute Parameters on the Edit Field dialog

After the administrators have made the proper configuration, users with the appropriate permissions may:

- Edit the Parameters field that is visible on the item, if any. The Parameters field essentially allows users to define a variable for that item, or other items that may inherit parameters from it. Apart from the name, users may add Description and choose whether it is a String or Pick parameter. If the latter, potential values may also be entered.

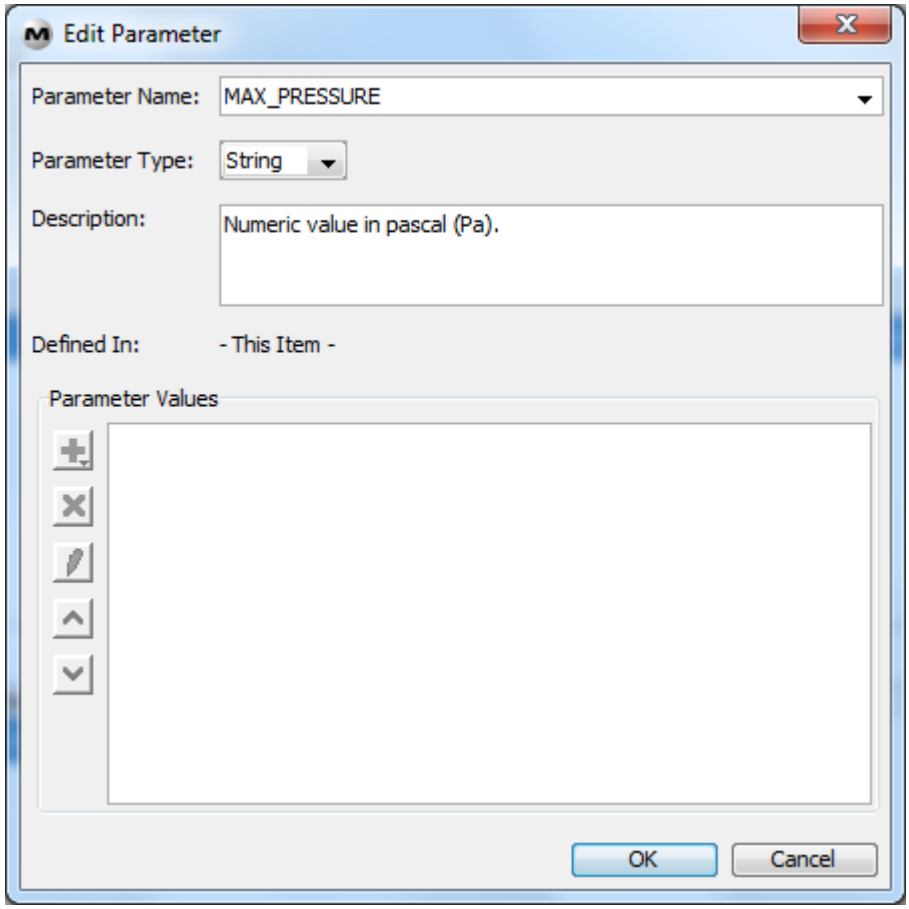

Fig J. Creating a Parameter

- Edit the Parameter Values field visible on the item, if any. The Parameter Values field allows users to set a value to an already defined parameter. The Parameters themselves might be defined on the same item, but are often inherited from elsewhere in the hierarchy. If a parameter value is locked, by clicking on the lock checkbox, it cannot be overridden by items further down in the hierarchy. If there have been no parameters defined on the current item, or in its hierarchy, it will not be possible to define a Parameter Value.

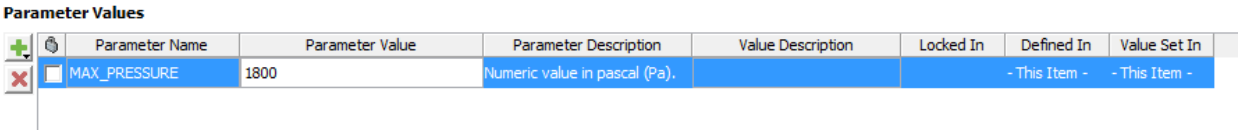

Fig K. Creating a String Parameter Value

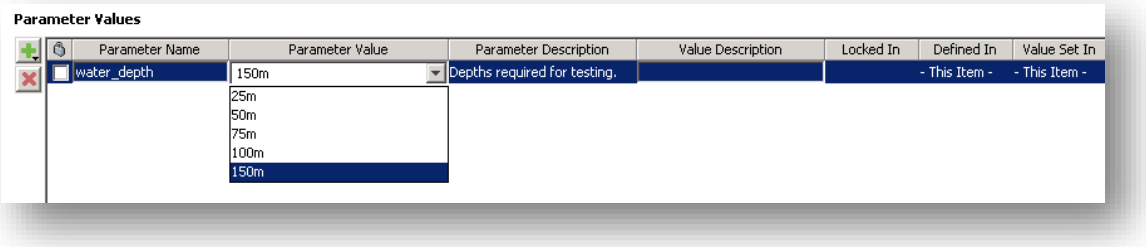

Fig L. Creating a Pick List Parameter Value

The project item is a good example of an Item that uses Parameters and Parameter Values, as well as potentially using inherited Parameter Values.

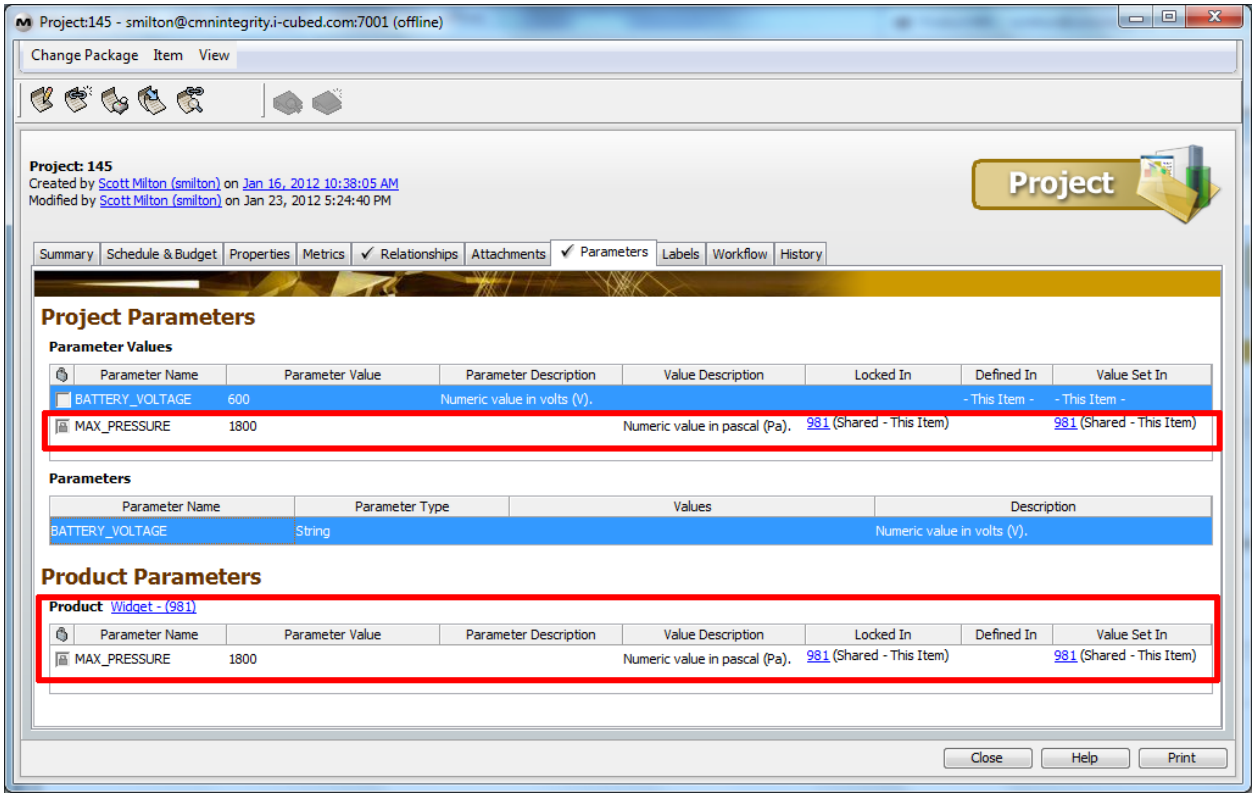

Fig J. Sample Project Parameters

In the figure above, there is a Product called "Widget" selected on the Project, and a Parameter called MAX_PRESSURE has been inherited from the Product. This particular Parameter has been locked, so cannot be overridden on this project. Any other Project items that select this Product will also inherit this value. There is also a project specific Parameter called BATTERY_VOLTAGE defined on this project in Particular and a value for 600 has been set on this Project. Since this is a project specific parameter, other projects will not know of this parameter definition, let alone its value of 600.

6.2.2. Inserting Parameters into Content

In any long text field that supports parameter substitution, the user simply types the parameter name in this field using the following format {{< parameter name>}}.

Example: If a parameter of MAX_PRESSURE has been defined, the text of a requirement may be written as "The widget shall withstand a pressure of {{MAX_PRESSURE}} without visible or functional damage".

6.2.3. Viewing Parameter Values Inline

If an item has parameters specified in a text field and the field is set up to support parameter substitution, selecting the "Substitute parameters" option will replace the parameter name with the correct parameter value. When parameter substitution is not enabled for a field, users see parameter entries in the following format: {{< parameter name >}} . To configure the Items view in the GUI:

- Select View > Options. The Options dialog box displays.
- On the General tab, select the "View layout > Substitute parameters" option.

Note: If a user changes a parameter value in the document view, the user must refresh the view in order to see the impact of the change on related document items.

For more information on how users work with Parameters and Parameter Values fields, see the PTC Integrity Lifecycle Manager 10.0 User Guide.

7. Strategies on Deferring Updates on Shared Content

Although immediate updates to shared content are desirable in many situations, this may not always be the case. Occasionally, a project manager may need to defer the updates to a future point in time, or may wish to defer them indefinitely. These are not primary use cases resolved by "out-of-the-box" configurations of Lifecycle Manager, but they can be addressed in a variety of ways.

First and foremost is the need for proper notification of updates. Users that will be selectively adopting updates on a case by case basis will need to know when new content is added, and when existing content is updated or dropped. This can easily be configured through the use of a "Stakeholders" or "Watchers" field, on documents and/or content items where users may add one or more names, essentially subscribing them to notifications of changes. Depending upon the prevalence of updates, a scheduled digest of emails may be more appropriate than immediate notifications.

In situations where project managers only have certain windows of time to accepts updates, notifications may not be needed at all. When a new window for updates is about to begin, documents containing the common content can be differenced between the current date and the date the last update window closed. The project manager can then decide how to configure their current content based upon the delta.

The easiest way to defer updates is when entire documents are inserted as the method of reuse. In this case, a branched copy of the common document can be inserted, rather than the original. The branched copy can be treated as a static copy of the common document at a certain point in time. It will remain unchanged as the common document continues to develop. When a new version of the common document is available and desired for inclusion, the existing branched copy can be dropped and a new branched copy of the current common document inserted. Traces would need to be propagated to the newly branched document using the Trace Propagation Wizard.

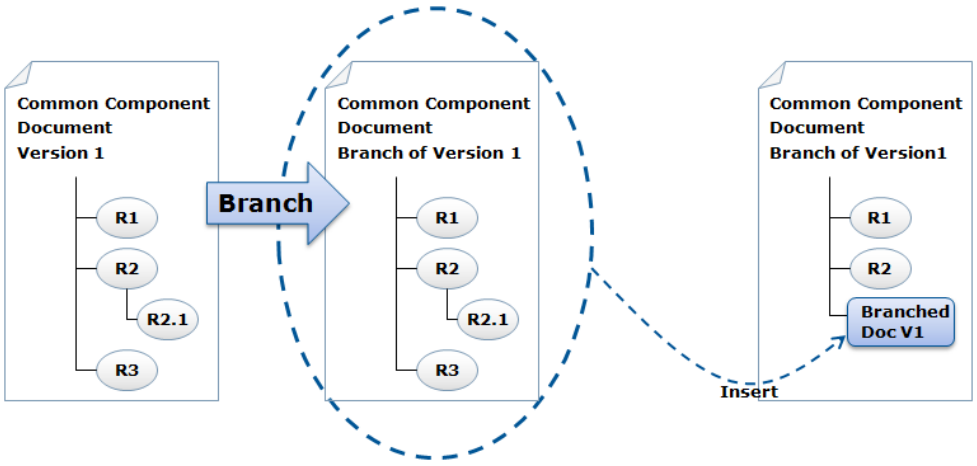

Fig K. Inserting Branched Document

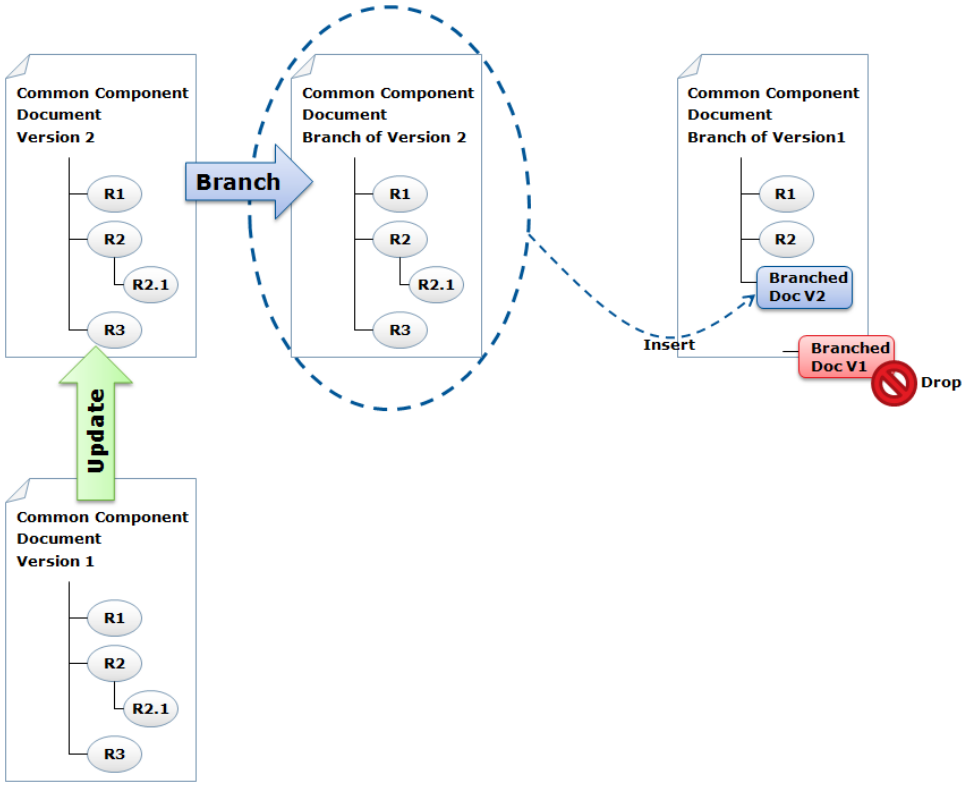

Fig L. Updating Inserted Branched Document

In situations where only individual content items, rather than entire documents are being reused, a more sophisticated method may be needed. A typical strategy would be to add additional fields to shared content, e.g., one that will reflect the accepted version of the text in the current document and another the current version of the text from the common source document. After appropriate notifications that the common document has been updated, the project manager would set a flag,

either on individual content or recursively throughout the document, causing a trigger script to fire and copy the current common value(s) into the accepted value(s). Relevance rules would be used to hide the current common value(s) from unauthorized users to prevent any possible confusion. Depending upon the exact needs, the trigger script could be configured to copy multiple fields, rather than just the primary text field of the content (e.g., priority, owner, etc.). It would also need to be decided whether this trigger script should create any nested requirements that may have been added in the common document, as well as how to process content that had been dropped from the current document.

In this scenario, manual notification and updates would still be necessary for entirely new content added to the common document. This should not be of great concern, because the origin of this use case is that the document author had decided to selectively reuse content.

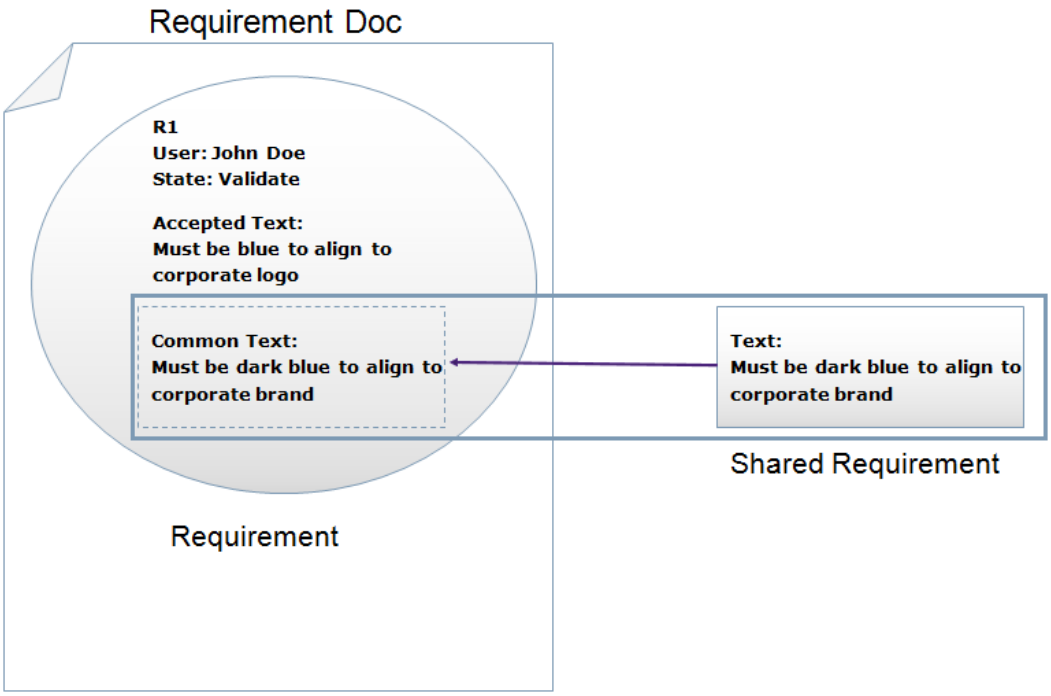

Fig M. Accepted Text and Shared Common Text on Requirement

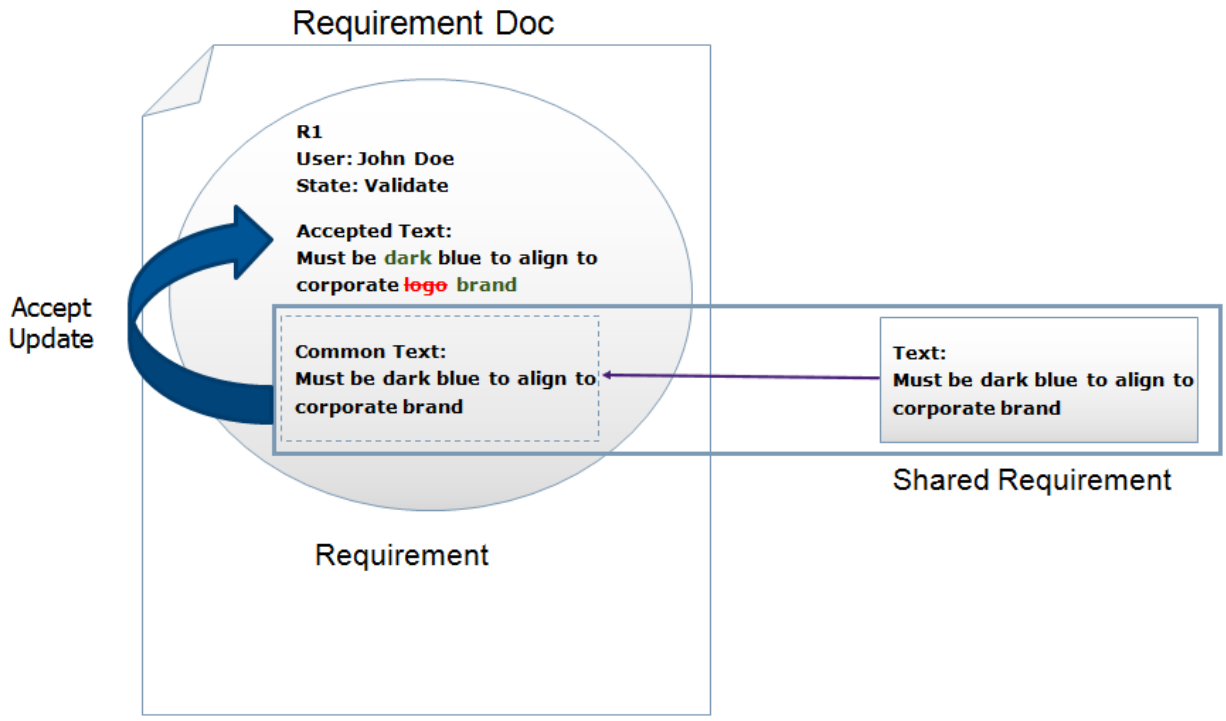

Fig N. Accepting Updates will copy Common Text over Accepted text

There are future plans to add core functionality to further enhance deferred update capabilities of shared content, but these have not yet been scheduled for a release.

8. Conclusion

Lifecycle Manager provides the most powerful and flexible mechanisms that provide real content "reuse" and "sharing". The capabilities were built from the ground up to be flexible and capable of handling real world scenarios. By leveraging these capabilities organizations can achieve real benefits to traceability, auditability and impact analysis of change management to such critical areas of development as Requirements and Test Management.

9. Appendix 1. Online Learning Video

A Lifecycle Manager Online Learning Video is available on this topic: Reusing Requirements.

10. Appendix 2. Theoretical Discussion of Requirements Reuse

A white paper describing additional theory around requirements sharing and reuse is provided here: Real Reuse for Requirements

11. Appendix 3. Hierarchy of Related Items Considered When Determining Parameter Values

The following hierarchy of related items is considered when determining parameters and parameter values for an item:

- Shared parameters from a related project item. For example, the related project item has an FVA field that displays a parameter value from the Product item.
- Parameters defined in the related project item.
- Shared parameters from a related test session item.
- Parameters defined in the related test session item.
- Shared parameters from a related document content item linked through a Shares relationship.
- Parameters defined in a related document content item linked through a Shares relationship.
- Shared parameters from the closest related document root item, for example, a test suite.
- Parameters defined in the closest related document root item.
- Shared parameters from the item containing the backing field or from the item represented by the table row.
- Parameters defined in the item containing the backing field for an FVA field or in the item represented by the table row.
- Shared parameters on the current item.
- Parameters defined in the current item.
- Parameter values higher in the hierarchy are overridden by parameter values lower in the hierarchy, unless the parameter value is locked, in which case the value locked at the highest point in the hierarchy is used. If the relationship does not exist, Lifecycle Manager skips that step and continues down the hierarchy.

If a parameter value is changed in an item, and an item that is lower in the hierarchy is open, the user must refresh the view of the lower level item in order to see the updates.

Note: Steps 9 and 10 only apply when substituting parameters in FVA fields backed by a text field with parameter substitution, or when substituting parameters in text fields for items in a relationship table field.

12. Appendix 4. Identifying where Content has been Shared or Reused

To truly understand where content has been shared or reused, it is necessary to explain couplet concept of the Document Model in more detail. Each time content is newly authored in Lifecycle Manager, two items are actually created; the content item (aka node) and the shared item. The end user generally does not need to be aware of the shared item, but its main purpose is to act a common location to store shared metadata from the content. As a practical example, if a requirement is shared 20 times, there will be 20 requirement items that all "point" to or reference the common Shared Requirement item. When the original, or author Requirement is updated, it will automatically pass the update on to the Shared Requirement item and each of the other 19 Requirement items will now display the updated data housed on the Shared Requirement.

It is easy to find where content has been reused/shared via two different mechanisms:

- For a quick summary, simply looking at the Branches tab on the content item is sufficient. This will generally be used to determine if there are any cases where the content item has been shared or reused and allow the user quickly navigate to them.
- In cases where more detail is required, especially if there is no desire to navigate to each of them individually, traversing the References relationship on the content item, then looking at the value of the Referenced By field on the Shared item may be a more desirable option. This view helps show the potential impact of updating the content item across documents and/or projects by showing what other locations are referencing the same version of the content. E.g., in the diagram on the next page, we can traverse the References relationship to the Shared Requirement and see that Requirement 132 is being "Referenced By" Requirement 441 with a Reference Mode of Reuse. Requirement 441 is contained by Document 408, which in turn could easily be opened and reviewed for more context.

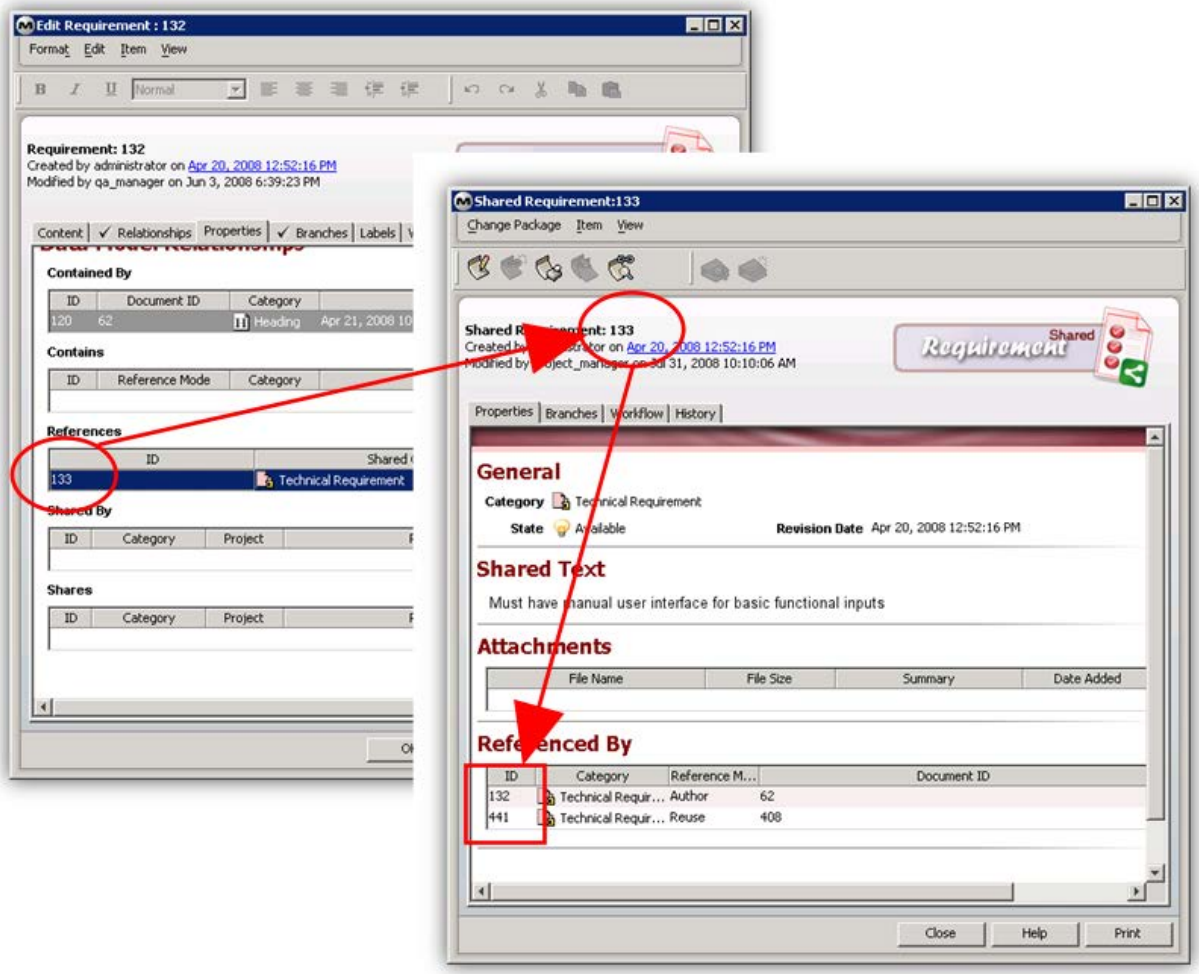

Fig O. References Relationship

Note: In some circumstances, the Shared Item that is displayed after following the "Referenced By" relationship might have been branched. The typical reason for this to have occurred is the original content item was reused in another location, but a significant edit was performed on the item. After a significant edit is performed on reused content, the reused content no longer "subscribes" to updates from its original source item and will have its reference mode set to Author. It is still easy to locate the original content that is was reused from however. The user can just look in the header of the shared item and look for the line "Branched from <ID #> by <User> on date <date>". The ID # displayed will be that of the original shared item.

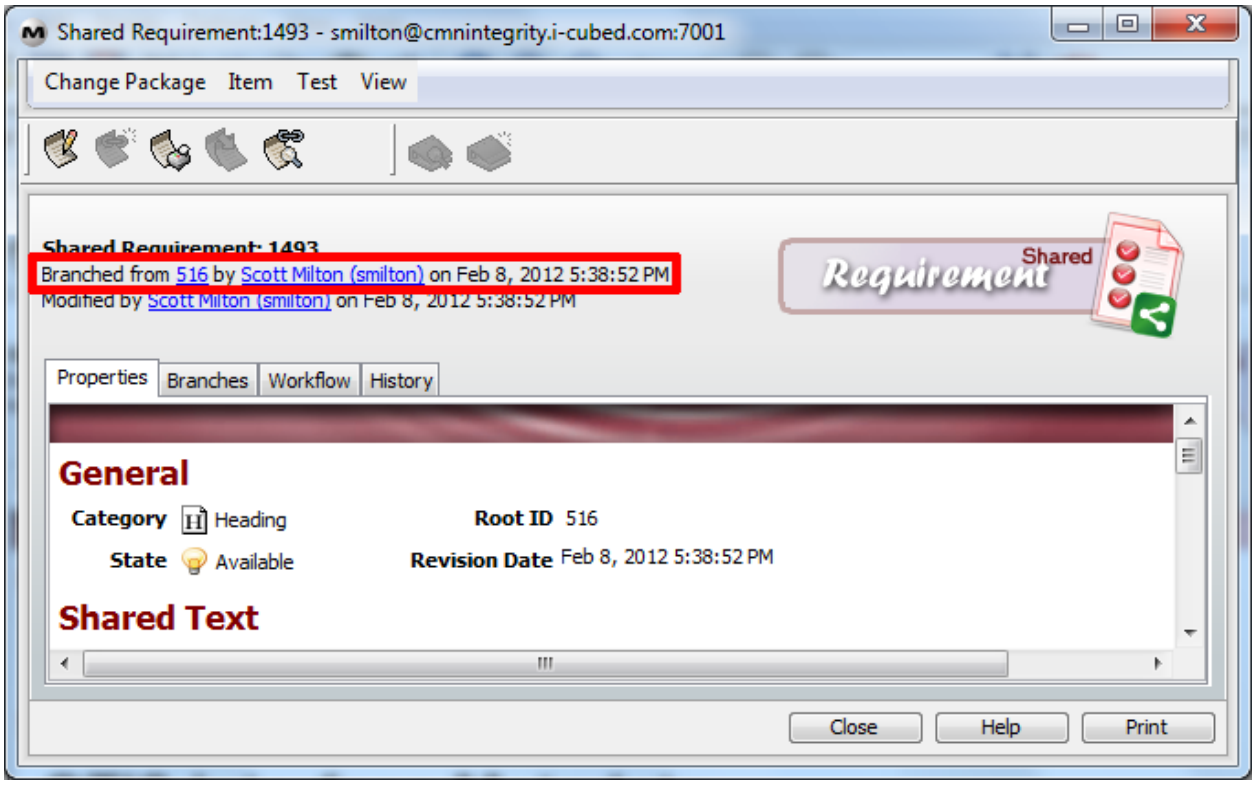

Fig P. Branching of a shared item

Additional details can obtained from the "Shares" and "Shared by" relationship fields on the original content item, although by default these fields are only visible to Administrators. "Shared By" will list instances where the content item has been shared to and "Shares" will list where it has been shared from. Neither field lists content that is being reused, only shared.

All of this information can be easily displayed in reports or the hierarchical relationships view as well.

13. Appendix 5. Theoretical Discussion of Software Product Lines

A white paper describing additional theory around managing Software Product Lines is provided here: Managing Variants in a Software Product Line (SPL) With Lifecycle Manager, A PTC Product