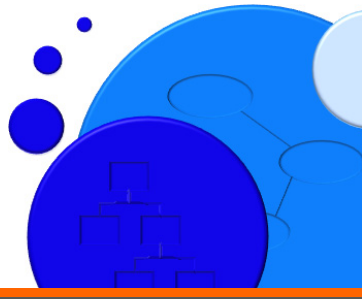
Cradle-7
From concept to creation...

3SL

Roles and Representation of User Requirements in Systems Engineering Using Cradle

RA007/07 June 2019



© June 2019 3SL. All rights reserved.
Cradle is a registered trademark of 3SL in the UK and other countries. All rights reserved.
All other trademarks are the property of their respective owners.

http://www.threesl.com
salesdetails@threesl.com
support@threesl.com

Structured Software Systems Ltd (3SL)
Suite 2, 22a Duke Street
Barrow-in-Furness
Cumbria LA14 1HH, UK
Tel: +44 (0) 1229 838867
Fax: +44 (0) 1229 870096
Regd: 2153654 VAT: GB 473 2757 28

Contents

List of Figures

List of Tables

Copyright © June 2019 Structured Software Systems Ltd
Cradle is a registered trademark of Structured Software Systems Limited. All other products or services in this document are identified by the trademarks or service marks of their respective organisations.

Introduction

This is one in a series of white papers that discuss the role and representation of types of systems engineering information in agile and phase-based processes in organisations that create products.

This paper discusses: ***user requirements***

Subject

The context of a project must be known and its boundaries clearly defined. This can only come from the project's ***stakeholders***.

Should stakeholders' statements be used as user requirements, or are user requirements a separate set of information?

Solution

We describe how to derive ***user requirements*** from stakeholders' statements and structure them in Cradle for further engineering.

Terminology

Table 1: Terminology

Term	Definition
Product	A physical / logical object that exhibits behaviour to change its environment
Need	Something that can be achieved by the realisation of requirements in products
User Requirement	An externally visible characteristic to be possessed by a product
System Requirement	Characteristic of a product component for the product to be able to satisfy its user requirements
Validation	Check that an externally visible product characteristic is as required
Verification	Check that an internal characteristic of a product component is as required
Test	Check to exercise a product component to confirm its behaviour or structure
Project	A set of activities that create or update products and their related information

Concept and Purpose 1

User requirements are statements of the characteristics to be possessed by a product that are visible externally.

User requirements are, by definition, the primary source of product information and so are the starting point for all product development. However, because the language needed in user requirements is not the natural language of your customers or your commercial and marketing colleagues, we recommend that the information you collect directly from stakeholders is stored as ***needs***, and that the user requirements are written from these needs.

There will be multiple sets of needs:

- A set generated internally in your business
- A set from each request from each customer

but only a single set of user requirements. Therefore, the transition from needs to user requirements is where you will resolve:

- Problems of ambiguity, contradiction, imprecision and duplication in each stakeholder's needs
- Complementary or contradictory wishes from all the stakeholders

Separating user requirements from stakeholders' needs helps you to resolve any needs that are complementary or conflicting, as:

- One user requirement encompassing all needs
- Several requirements when needs are not compatible

so that each need can be:

- ***Accepted***, and linked to user requirements
- ***Rejected***, and not linked to any user requirements
- ***Combined***, where several needs are linked to one user requirement
- ***Separated***, where one need is linked to several user requirements to allow more precision in the transition between the two sets of data

This is an easy way to categorise and report accepted and rejected needs, and derive metrics and KPIs from them.

User requirements fulfil the most important role of all database items. They are a re-expression of needs into a form from which system requirements can be created. User requirements specify characteristics that are desired from a product by someone who can only see, and is only concerned with, the external product, not its internals.

In agile processes, user requirements are the basis for user stories. Each user story should be linked to either one or more user requirements, or one or more features that, in turn, are each justified by being linked to one or more user requirements.

Content and Creation 2

These are the highest level requirements in the database. They will be written by you, not the stakeholders, and will probably be created manually, most likely by copying and rewording needs, and least likely by being captured from documents. The exception will be when existing or legacy data exists, when it should be loaded directly from the external documents into the database.

The purpose of the user requirements is to translate the language of needs (the only source material available) into the language of the functional and non-functional characteristics of your current and new products.

The authors of user requirements will re-express needs as the externally-observable characteristics required of new and existing products. This is a highly skilled task. As user requirements are the basis for everything that follows, the work must be performed, and reviewed, very carefully.

Most user requirements express the functional and non-functional characteristics required by a product's users:

- **"When the product is powered-on, pressing the emergency button will stop all current operation within 50msec."**
- **"All exterior surfaces must be painted blue, using paint reference ABC123, to a maximum thickness of 50 micron."**
- **"The product's mass in its operating condition must be no more than 350g."**
- **"The product must have an angular positioning accuracy not greater than 0.5°."**

but they can also be written from the perspective of other stakeholders, including:

- Purchaser
- Maintainer
- Disposer / recycler

It is often helpful to attempt to specify the ***scope***, or ***level***, of a user requirement. This is the ***largest proportion*** of the product whose externally-observable characteristic is being specified, such as:

1. A user requirement that states:
   "When the product is powered-on, pressing the emergency button will stop and all current operation within 50msec."
   refers to the product, so its scope would be: **Product**
2. A user requirement that states:
   "All parts of the casing that provide access to potentially hazardous electric currents (voltage > 12V or current > 250 mA) must be secured with Torx headed bolts."
   relates only to the casing, which will be an assembly or a part, so its scope would be: **Assembly**
3. A user requirement that states:
   "All user documentation must be provided in all official

languages of the European Union, Simplified Chinese and Korean."

relates to each document, so its scope would be: **Part**

Specifying the scope of a user requirement is intrinsically valuable, adding information that aids the understanding of its content, meaning and intent. It also helps when user requirements are to be satisfied by system requirements. Specifying scopes allows engineers to divide the user requirements into groups with the same scope, and then to consider each group, starting at the highest scope and progressing to the lowest, most detailed, scope.

The scope of a user requirement does not represent its position in a user requirement hierarchy. This hierarchy is likely to contain many levels of requirement, structured as described in the next section, in which the higher levels provide grouping in a manner that is logical and easy to understand. It is likely that only the ***leaves*** in this hierarchy will contain requirements, and it is only in these lowest-level requirements that the scope values will be valid and meaningful.

User requirements will be written using language that will:

- Only describe externally-visible characteristics
- Be either functional or non-functional
- Be singular (often termed ***atomic***), so words such as "**and**", "**both**" and "**also**" are typically not allowed
- Be precise
- Be measurable and testable

Although atomic statements are generally helpful for all information, it is vital for user requirements. More than any other set of information, user requirements are the subject of traceability and coverage analyses, often shown in ***compliance tables*** (such as user requirements to system requirements, or user requirements to validations) or matrices. These analyses use the cross references to and from each user requirement. If a user requirement contains a list of paragraphs, then cross references for each user requirement will refer to all of the statements in the list inside it, and not to an individual statement in this list. This will mean that the analyses have more links to and from each item, making the analysis more complex and more difficult to understand and to check. When requirements are atomic, there will be fewer links to and from each requirement which means that the analyses of the correctness of the links will be clearer and easier.

The user requirement statement that is finally agreed is, of course, of paramount importance. However, as time passes, it becomes equally important to understand why the requirement ***is as it is***.

This does not mean to understand the evolution of the

user requirement into its current form. A record of that evolution is always available in the item's ***change history***. Rather, this means to understand ***why*** the statement is as it is and why it is not ***something else***.

We strongly recommend that a ***rationale*** is recorded for each user requirement. This rationale could include many things but, as a minimum, we believe that it should record:

- All of the ***alternatives*** that were considered
- ***Why*** a particular alternative was chosen
- The ***assumptions*** that were used to make this decision
- For numerical values, sometimes termed ***measures of effectiveness*** (MoEs), why that particular value, or range, have been specified and why any tolerances applied to these value(s) are as they are

The assumptions are a particularly important part of the rationale. They are often based on current technology or techniques. Over time, these assumptions can become invalid, and a review of user requirements must include an appraisal of the assumptions to confirm their validity. Assumptions that have become invalid with advances in technology or techniques could easily change which of the alternatives should be the user requirement, or indeed introduce new alternatives for the user requirement.

We recommend that, wherever possible, the providers of the needs are consulted to approve the re-expression of their needs in the user requirements. We understand that this may not always be possible and that a careful decision will need to be taken whether it is commercially helpful to expose customers to this information.

Structure and Organisation 3

Since the user requirements are derived from the needs, and the needs are organised by project, there will be one hierarchy of user requirements for each project. Each hierarchy will contain a mixture of different types of user requirement.

It is your decision how to organise the user requirements in each hierarchy. User requirements could be organised by type, such as:

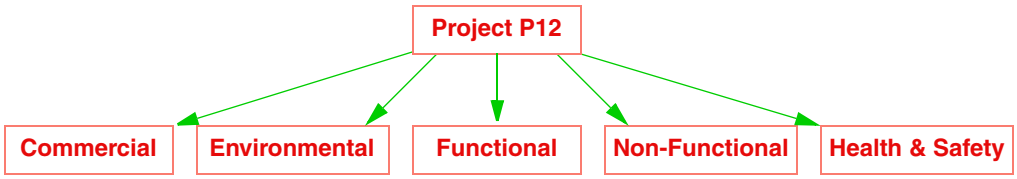


Figure 1: Organise User Reqs by Type

or by discipline, such as:

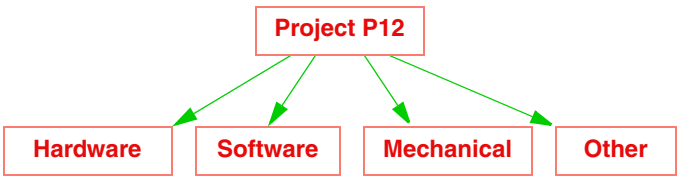


Figure 2: Organise User Reqs by Subject

or by the products' lifecycle (how it will be used), such as:

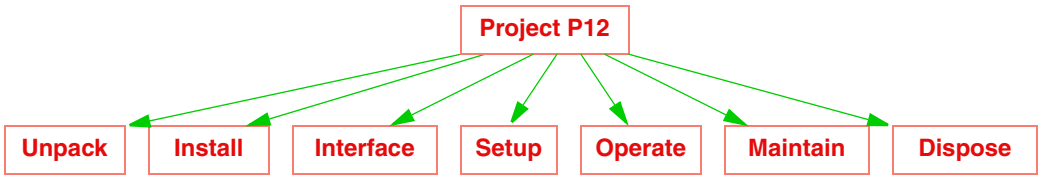


Figure 3: Organise User Reqs by Lifecycle

We recommend this last option, that user requirements are structured in hierarchies by the products' lifecycle. We believe that this is the easiest approach, because:

- It encourages user requirements to be created from the perspectives of all stakeholders
- These perspectives are closer to the perspective of the people who create the needs
- It is a smaller step from needs to user requirements than if the requirements were grouped any other way
- Such requirements are easier to write and review

This is a different structuring convention to that used for needs, but we believe that this is outweighed by the above advantages.

Traceability 4

Each user requirement will be linked to the need(s) from which it has been derived. We propose that only the bottom-level needs are linked to user requirements. It is likely that the user requirements that are linked to needs will have further decomposition, so the links between them are likely to be from bottom-level needs to middle-level user requirements.

This approach has several benefits:

- It is easy to do, Cradle has a built-in option to do it
- In one step, it finds all the needs whose traceability to user requirements is important
- It simplifies reporting the extent to which needs have been satisfied by user requirements
- It simplifies the calculation of metrics and KPIs for the completeness of the traceability

The top of each project's user requirement hierarchy will be linked from the **PROJECT** item for that project:

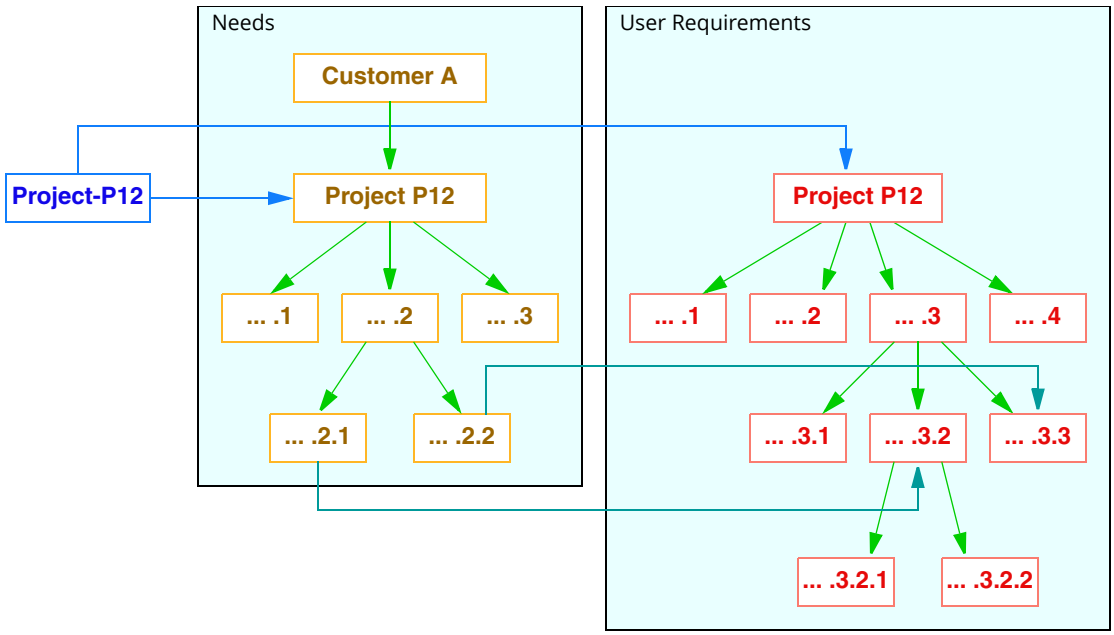


Figure 4: Projects, Needs and User Reqs

We recommend that traceability matrices are defined as views between the user requirements and the needs.

Lifecycle 5

An organisation will need to define its own process for the development of user requirements, particularly whether it is viable for customers and other stakeholders to be part of the user requirements' review. The end stage of such a process is that the user requirements are agreed by all parties involved, so the user requirements can be:

- Formally reviewed and baselined, if the project intends to use a formal configuration management (CM) approach, such as that provided by Cradle's Configuration Management System (CMS)
- Used as the basis to develop the system requirements

Several steps can be imagined in such a process. The path that each user requirement can take through these steps is called a ***workflow***. The position of each requirement in this workflow is recorded in an attribute in that item. We will use the name **Maturity** for this attribute.

Your organisation must decide what is to happen if a user requirement is no longer required. There are two choices:

1. Delete the user requirement from the database
2. Keep the user requirement in the database for the historical record (it may reappear in the future) with a **Maturity** value that you use to exclude such user requirements from the sets of items that you use for the rest of the project

We recommend the second of these approaches.

Assuming that customers and other stakeholders are to be involved in the review of user requirements, we suggest the following workflow (the **Maturity** attribute values are shown in parentheses):

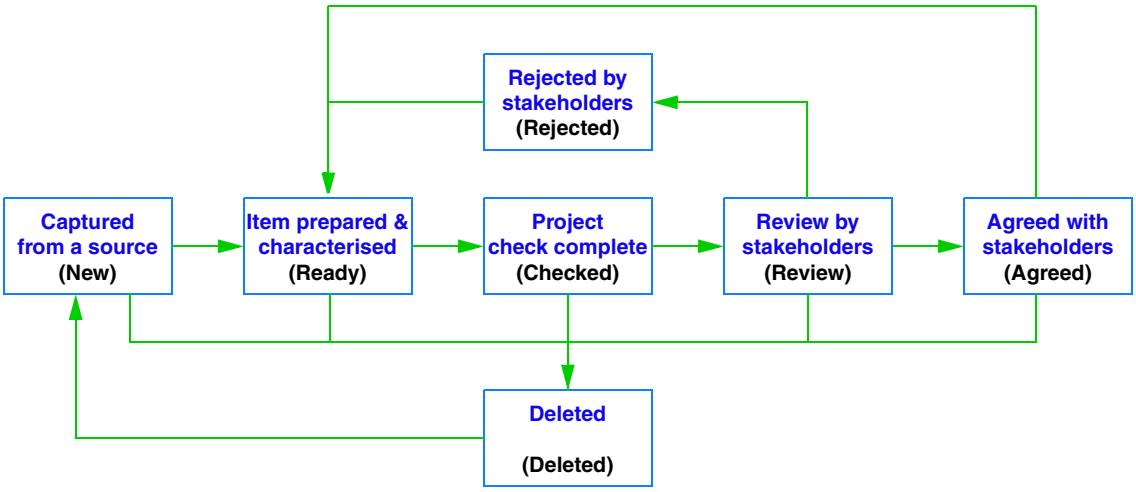


Figure 5: Workflow

Table 2: Workflow Stages	
Stage	**Description**
New	The item has been captured from an external document or entered manually
Ready	The item's statement has been cleaned of any compound or ambiguous statements and its attributes have been set to reflect its content and the structuring method used by the project
Checked	The item's content and attribute characterisation has been checked and is acceptable
Review	The item is being reviewed by stakeholders
Agreed	The item has been accepted by stakeholders
Rejected	The item has been rejected by stakeholders whose comments have been recorded in the item and it is to be reworked so as to become acceptable
Deleted	The item is no longer needed and will not be included in any further processing of items of this type

The workflow supports the idea that **Agreed** user requirements may be modified as the project progresses, perhaps after being baselined.

Database Representation 6

The same structure can be used for all user requirements, so they can be represented by one item type called **UR**. This item type will be hierarchical. The **UR** items will be auto-numbered to allow reordering of the hierarchies. The hierarchical numbers in these hierarchies will be stored in the **UR** items' **Key** attribute.

Since there will be one user requirement hierarchy for each project, the format of the **Key** attributes in the user requirements will be:

project.hierarchical-number

such as:

Project-P12.1.2 **P4-2016.2.3.4**

where the names of the **.1**, **.2**, **.3** ... items in the hierarchy will be based on the structuring criterion you chose. If this is lifecycle, the names of the **.1**, **.2**, **.3** ... items in each hierarchy will be **Unpack**, **Install**, **Interface** and so on.

User requirements will require attributes to:

1. Name the user requirement, acting as a summary
2. Specify the user requirement, as:
   a) Plain text, and/or rich text, with also
   b) An optional figure (JPEG) and table (RTF)
   c) Optional Excel®, Word, Visio® or PDF® documents
3. Provide the rationale for the user requirement as it is
4. Provide notes for the user requirement
5. Characterise the user requirement:

   - **Discipline** : as one of: **Hardware**, **Software**, **Sales**, **Mechanical**, **Marketing**, **Other**
   - **Maturity** : as one of: **New**, **Ready**, **Checked**, **Review**, **Agreed**, **Rejected**, **Deleted**
   - **Priority** : as one of: **High**, **Medium**, **Low**
   - **Scope** : as one of: **Product**, **Subsystem**, **Assembly**, **Part**
   - **Type** : as one of: **Commercial**, **Environmental**, **Functional**, **Non-Functional**, **Health & Safety**
   - **Validation**: as one of: **Inspection**, **Analysis**, **Demonstration**, **Test**
   - **Lifecycle:** as one of: **Unpack**, **Install**, **Interface**, **Setup**, **Operate**, **Maintain**, **Dispose**

   which could be set for user requirements at all levels, but must be set for bottom-level user requirements
6. Record external comments for the user requirement, such as from the review by stakeholders

The method chosen to structure user requirements could replace the **Lifecycle** attribute. Despite an overhead to maintain it, we keep the attribute in the item type to simplify searches.

Database Schema 7

User requirements are one set of database items. A simple ***schema*** for product development processes is (item type names in parentheses):

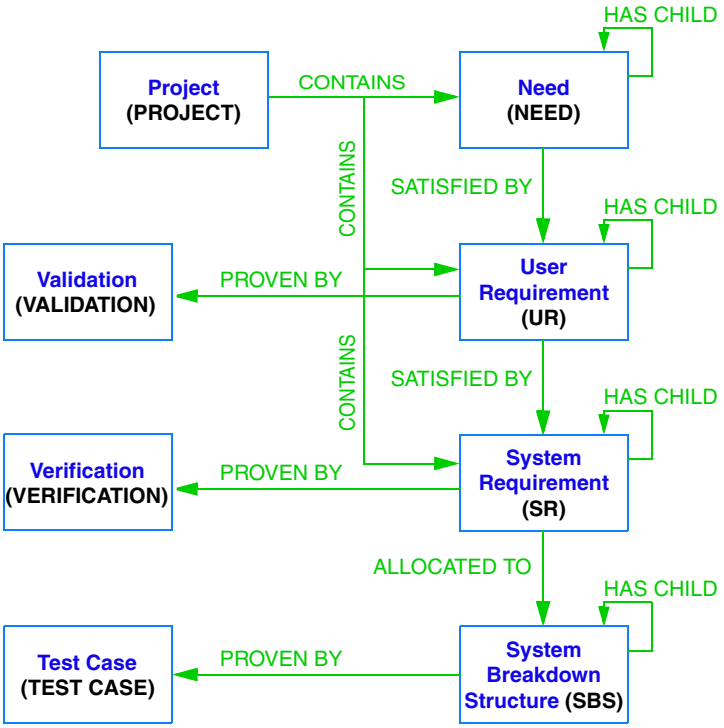


Figure 6: Database Schema

Verification 8

Validation is to check that "we built the correct product" while ***verification*** checks that "we built it correctly".

There are two distinct verification activities, one related to the product and described in the database, and one that confirms the correctness of the database itself:

- Verification items that describe the checks to be applied to the product components to ensure that system requirements have been met
- The act of checking all sets of **SATISFIED BY** cross references to ensure that the items at the **to** end of these cross references are a correct evolution of the items at the **from** end of the links

This second activity is a manual check by users using traceability and coverage views produced by Cradle and their engineering knowledge and experience.

A common use for cross reference attributes is to record the result of reviewing cross references. To do this, a link attribute such as **Status** could be defined with values **Approved**, **Rejected** and **TBD**. An approved cross reference is a cross reference whose **Status** attribute has the value **Approved**. One or more ***navigations*** would be defined that either only follow, or specifically exclude, cross references with this value in their **Status** attribute. These navigations

could be used to produce all the reports and documents that are part of the project's formal deliverables.

In this way, not do such deliverables only contain reviewed and approved items, but they will only contain links that have also been reviewed and approved.

Although we recommend this verification technique, it is not often used in practice and so has not been included in the schema definition described in the following sections.

Item Type 9

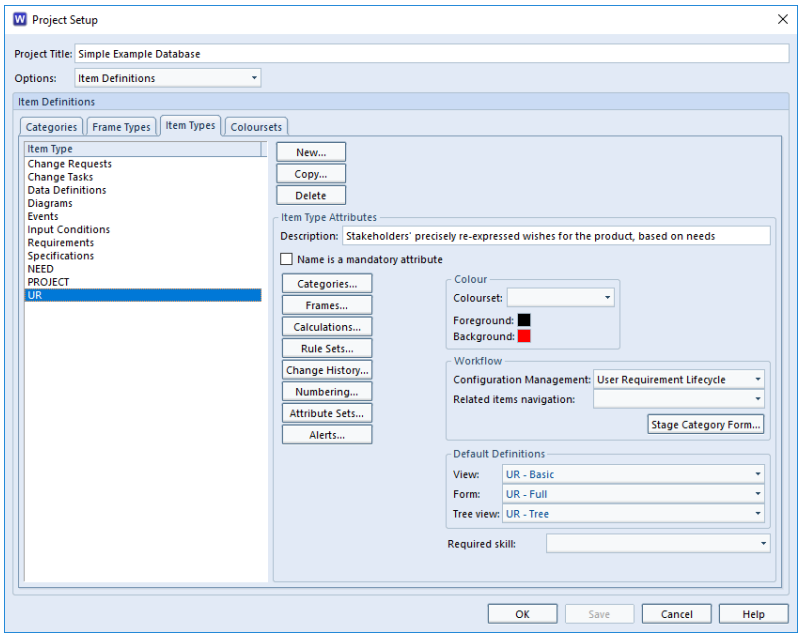
The ***item*** is the basic unit in a Cradle database. Each user requirement will be an item of type **UR**, defined as:

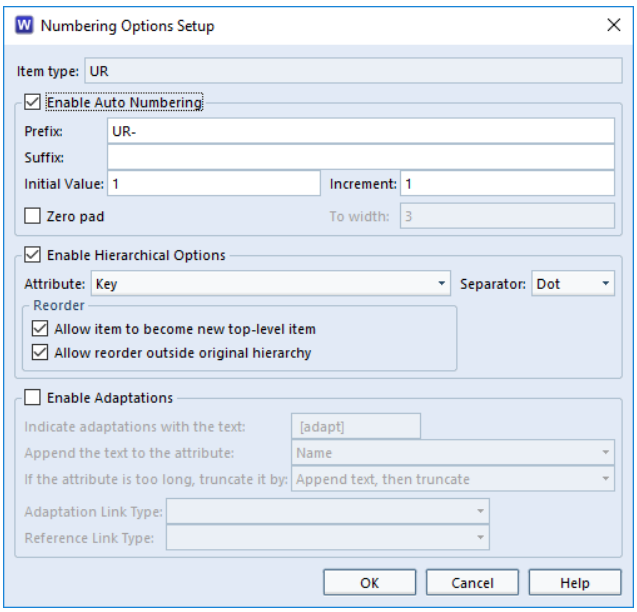
Table 3: Item Type Definition

Name	Categories	Frames			Hier-archical	Adap-tations	Change History
		Name	Type	Auto-Numbering			
RATIONALUR	**Discipline** **Lifecycle** **Maturity** **Priority** **Scope** **Type** **Validation**	**CALCS**	**EXCEL (XLSX)**	Yes **UR-***n* *n*= 1,2,3...	Yes, in the **Key**	No	Yes
		COMMENT	Plain text				
		DIAGRAM	**VISIO**				
		DOCUMENT	**WORD (DOCX)**				
		FIGURE	**JPEG**				
		NOTES	Plain text				
		RATIONALE	Plain text				
		RICH TEXT	**RTF**				
		SLIDES	**POWERPOINT (PPTX)**				
		TABLE	**RTF**				
		TEXT	Plain text				

In this item type definition:

1. ***Auto-numbering*** is enabled, so all user requirements have a unique identity that is fixed even if they are reorganised in their hierarchies. We usually recommend that items are auto-numbered.
2. ***Hierarchical*** is enabled, the hierarchical value is the **Key** attribute with a level separator dot (.) in the hierarchical values. Enabling this feature means that Cradle will provide the **New Child**, **New Sibling**, **New Hierarchy** and **Reorder** operations for user requirements, and other capabilities.
3. ***Adaptations*** is disabled.



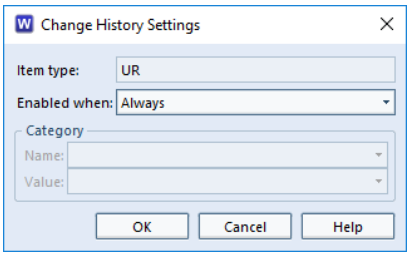


This feature is not needed for user requirements in this schema, but can be essential for user requirements in other contexts.

4. ***Change histories*** are enabled, and set to **Always** so that changes are always recorded in user requirements. Other options are available for recording change histories.
5. ***Categories*** are the attributes used to characterise the user requirements. They appear in the next section.
6. ***Frames*** are large attributes that contain the data inside the item. They are discussed in a later section.

There are other options for each item type, such as:

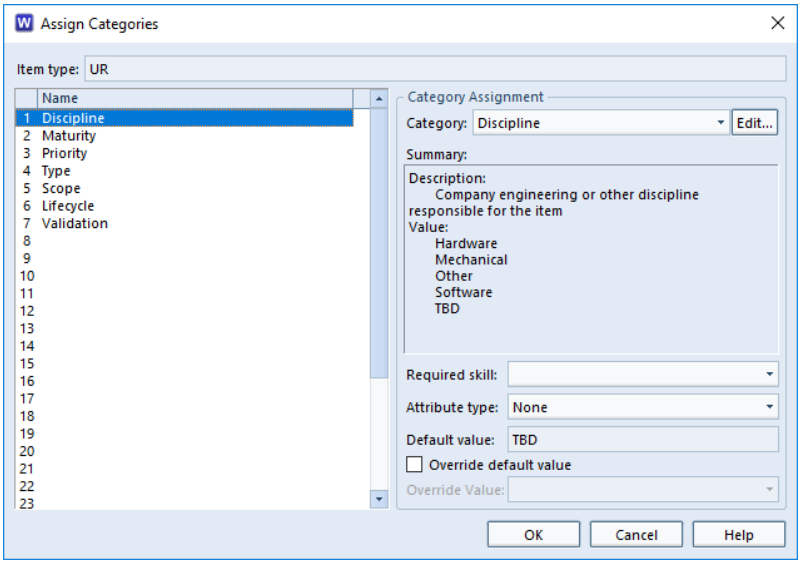
- The colours used to show items in trees and ***Hierarchy Diagrams*** (HIDs) for graphical views of traceability
- The ***form*** used to show items individually
- The ***views*** to display user requirements by default (if a specific view is not chosen), and to show them in trees



You can control users' access to items of the type with a ***skill***. We recommend that such access controls are ***only*** added when you know that your process and project ***organisation structure*** (***users*** and ***teams***) require it.

Category Codes 10

Category codes are small data values used to ***characterise*** database items. You can define any number of category codes and assign up to 32 of them to each item type. Category codes are defined once and used in many item types. Cradle is optimised to find items based on category code values, and stores all items pre-sorted by all categories so filtering and sorting by category value is very efficient.



User requirements can be characterised in different ways. We suggest that you use categories for each of the structuring methods (see "Structure and Organisation" on page 5) that were not used to group the user requirements, in this case type, importance,

engineering discipline, maturity and validation method.

The categories in the schema for user requirements are:

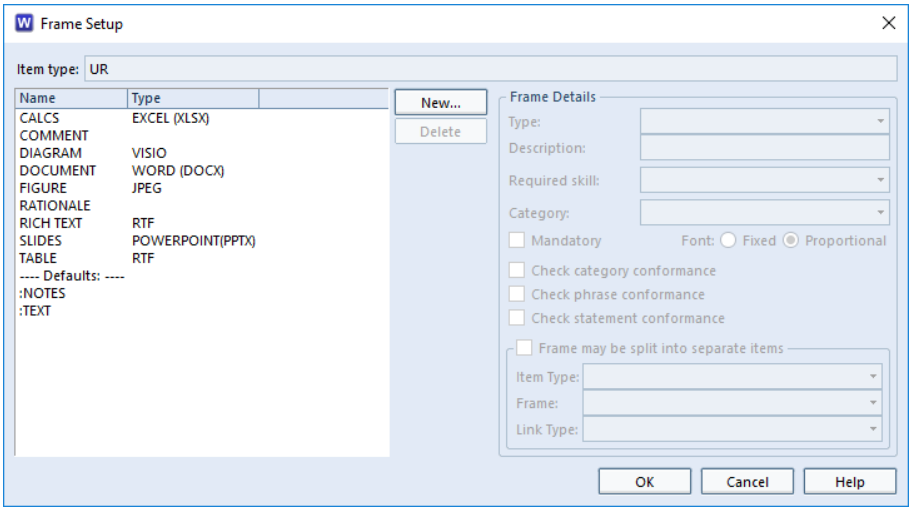
Table 4: Category Code Definitions

Name	Type	Values (* is default)	Description and Purpose
Discipline	Single-value pick-list	**Hardware** **Software** **Mechanical** **Sales** **Marketing** **Other** **TBD***	Which group or engineering discipline is responsible for re-expressing the need as user requirement(s)
Lifecycle	Single-value pick-list	**Dispose** **Install** **Interface** **Maintain** **Operate** **Setup** **TBD*** **Unpack**	The part of the product's lifecycle, as perceived by a user or other stakeholder in the product, to which a user requirement refers.
Maturity	Single-value pick-list	**Agreed** **Checked** **Deleted** **New*** **Ready** **Rejected** **Review**	The values of the lifecycle for a user requirement. It is also used to avoid deleting user requirements when their authors have removed them. Such requirements may be re-introduced, so having them available saves time. Such requirements may be linked to other items, keeping them makes it easy to find items that link to redundant requirements, so these items can be considered for removal. It also allows the justification for everything in the database to be traced back to requirements that are agreed.
Priority	Single-value pick-list	**High** **Medium** **Low** **TBD***	The importance of the user requirement to the user. In agile projects, this is the first determinant for ordering the feature backlog, so that the earliest sprints or iterations will implement those user stories that satisfy the highest priority requirements.
Scope	Single-value pick-list	**Product** **Subsystem** **Assembly** **Part** **TBD***	The level or scope of the user requirement. This helps to group the user requirements when system requirements are being derived from them.
Type	Single-value pick-list	**Commercial** **Environmental** **Functional** **Non-Functional** **Health & Safety** **TBD***	The type of the user requirement. This is the type of externally-observable aspect of the product that the user requirement describes. It helps to group user requirements that refer to the same type of product characteristic, and helps to determine how the user requirements will be satisfied, and by whom.
Validation	Single-value pick-list	**Inspection** **Analysis** **Demonstration** **Test** **TBD***	The validation method that will be applied to the product to demonstrate its compliance with the user requirement. The validation method is closely tied to the **Type**.

Frames 11

Frames are large attributes (up to 1 TByte each) that contain the majority of the data in an item.

Each frame has an underlying data type, called a ***frame type***, that defines the type of data, how and where it is stored (in Cradle, in a file, in another tool or a URL), and how to operate on the data (get, set, edit, check, print etc.).



We suggest some frames that should be sufficient to manage all of the information in your user requirements and to store comments from external groups. You can ignore any frames that are not relevant to you.

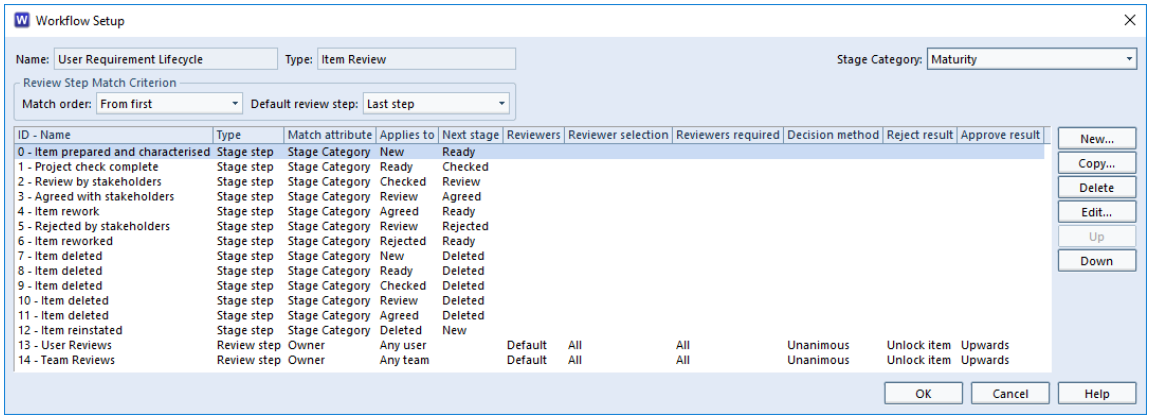
The frames defined in the schema are:

Table 5: Frame Definitions

Name	Type	Description and Purpose
CALCS	**EXCEL (XLSX)**	Store calculations or any other information in a spreadsheet
COMMENT	Plain Text	For comments from external groups, such as stakeholders
DIAGRAM	**VISIO**	Store any diagram or sketch that is a Visio drawing
DOCUMENT	**WORD (DOCX)**	Store any document held in Word
FIGURE	**JPEG**	Store any image
NOTES	Plain Text	Provide explanatory notes for the need
RATIONALE	Plain Text	Why the user requirement is as it is, and why it is not something else. It could include many things, but we recommend that, as a minimum, it contains: • All of the ***alternatives*** that were considered • ***Why*** a particular alternative was chosen • The ***assumptions*** that were used to make this decision
RICH TEXT	**RTF**	The user requirement statement, if it must be represented in rich text, using fonts, colours, formulae and so on
SLIDES	**POWERPOINT (PPTX)**	Store any information that is a presentation
TABLE	**RTF**	Store information held in a table for edit by WordPad, Write, Word
TEXT	**Plain Text**	The user requirement statement, stored here if it is not rich text

Workflow 12

Each type of item in the database has a workflow set for it. This workflow can be one of the defaults that simply



specifies the CMS reviews to baseline information. A specific workflow is required for user requirements to implement the lifecycle defined in “Lifecycle” on page 7. This workflow also includes the steps that are necessary to review user requirements into baselines, if required by your project.

Link Types 13

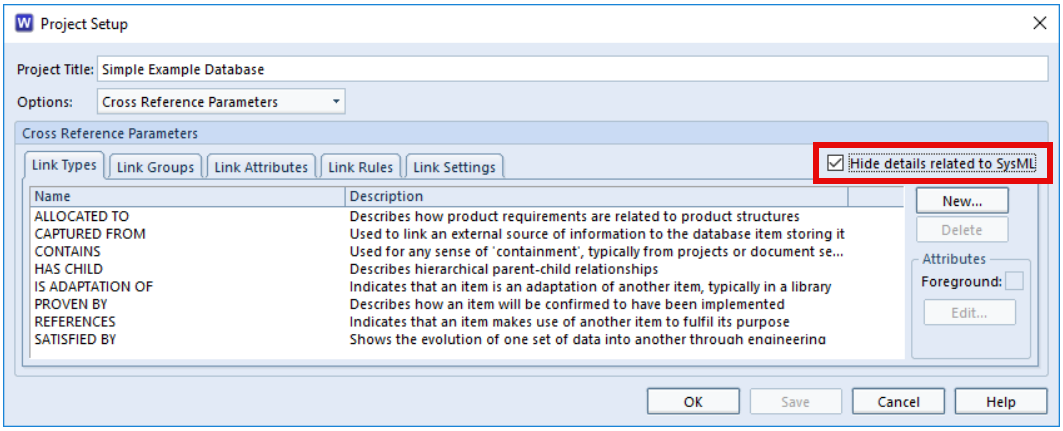
Cross references describe relationships or dependencies between database items. They can optionally have a ***link type***, chosen to describe the relationship. Items can be simultaneously linked by ***any*** number of cross references with M:N ***cardinality*** provided that these cross references have different link types.

Cross references can store data in attributes inside the links themselves. These attributes are sometimes used to characterise, parametise, justify or simply explain the existence of a cross reference. These attributes can be free-format text or pick-lists of possible values. They can be used to select the cross references to use to find linked items in situations such as:

- Produce a view or report
- Expand an item in a tree node, or a table row
- Show a list of linked items in the UI
- Show linked items in a form used to edit an item
- Show items in nested tables
- Display relationships graphically in a HID
- Publish a document

Cross reference attributes have not been defined in this schema.

There are many link types in the schema. The majority are needed for Cradle’s SysML® implementation. You can use these link types or define your own. You can hide the SysML-related definitions in all parts of the schema, as shown in the figure:



The types of cross reference used in the schema in Figure

6: ”Database Schema” on page 10 are:

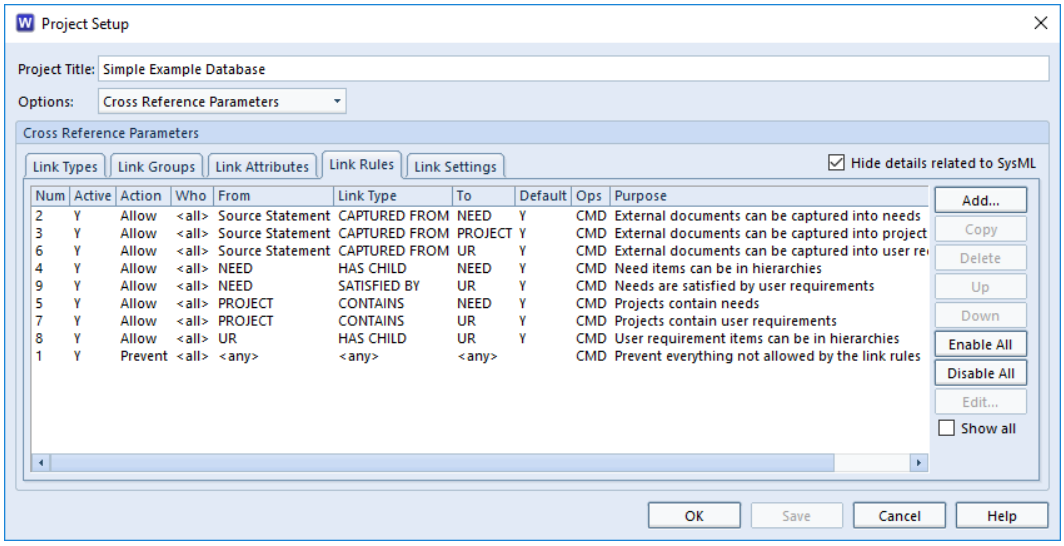
Table 6: Link Type Definitions	
Name	**Details**
ALLOCATED TO	Describes how system requirements are related to product structures
HAS CHILD	Describes hierarchical parent-child relationships
PROVEN BY	Describes how an item will be confirmed to have been implemented
SATISFIED BY	Shows the evolution of one set of data into another through engineering

Link Rules 14

The ***link rules*** in the schema either allow or prevent operations on cross references. Initially, the list of rules is empty, so it has no effect. When a user does anything with cross references, Cradle checks the link rules, from first to last, to find a rule that matches the operation. If a match is found, the rule allows or prevents the operation. If there is no match, the operation is allowed.

Link rules can be generic, or very specific. They can refer to:

- One or all item types
- For items with ***stereotypes***, part of Cradle's support for MBSE (model based systems engineering), one or all stereotypes, or a stereotype and its hierarchy
- All items of a type, or items matching a criterion
- All users, a group of users, or a single user
- One or more cross reference operations



To define the link rules for a schema, we recommend:

- Define a link rule that prevents all operations by all users on all cross references between all item types
- Precede this rule by one rule for each set of links shown in Figure 6: ”Database Schema” on page 10

This approach is easy to understand, as the link rules specify the only cross reference operations that are to be allowed.

It is also helpful to add a purpose to each link rule. If a user performs an operation that violates a link rule, then the purpose of that link rule is shown to the user, to explain why the operation is not allowed.

Navigations 15

Navigations are usually created to filter cross references by link type. If two sets of items **A** and **B** are linked by more than one type of cross reference, create navigations that filter by each link type, so users can see which **B**s are linked to each **A** by the first or second link type, and vice versa for **A**s linked to each **B**.

There is only one type of cross reference between each pair of item types in the schema, so extra navigations are not needed and you will only need some of the navigations supplied by 3SL.

Since the user requirements will be auto-numbered, the most useful of the as-supplied navigations are those that sort linked items by their **Key**. These navigations are:

Table 7: Useful As-Supplied Navigations

Name	Description
Bidirectional by Key	Follows all types of cross reference bi-directionally (upwards and downwards), sorting the linked items by their **Key**
Downwards by Key	Follows all types of cross reference downwards, sorting the linked items by their **Key**
Upwards by Key	Follows all types of cross reference upwards, sorting the linked items by their **Key**