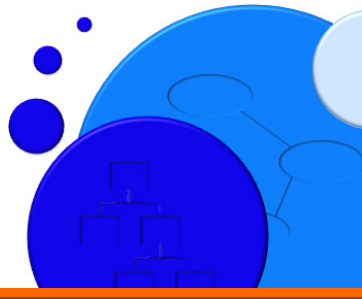
Cradle-7
From concept to creation...

3SL

Role and Representation of Needs in Systems Engineering Using Cradle

RA006/07 June 2019



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

Introduction 1
Subject 1
Solution 1
Terminology 1
1 Concept and Purpose 2
2 Content and Creation 2
3 Structure and Organisation 3
4 Traceability 4
5 Lifecycle 4
6 Database Representation 6
7 Database Schema 6
8 Verification 7
9 Item Type 8
10 Category Codes 9
11 Frames 10
12 Workflow 11
13 Link Types 11
14 Link Rules 12
15 Navigations 13

List of Figures

Figure 1:Overall Structure of Needs 3
Figure 2:Organise Needs by Type 3
Figure 3:Organise Needs by Discipline . . . 4
Figure 4:Projects and Needs 4
Figure 5:Workflow 5
Figure 6:Database Schema 7

List of Tables

Table 1:Terminology 1
Table 2:Workflow Stages 5
Table 3:Item Type Definition 8
Table 4:Category Code Definitions 10
Table 5:Frame Definitions 11
Table 6:Link Type Definitions 12
Table 7:Useful As-Supplied Navigations . 14

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: ***needs***

Subject

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

Are stakeholders' statements used directly, or should they be linked to a set of items that form user stories or requirements?

Solution

We describe how to capture stakeholders' statements as needs, and structure them in a Cradle database 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

We propose the starting point for any project is ***needs***, not requirements. Needs are expressions of outcomes to be achieved by new or existing products. The term ***objective*** is used in business analysis to mean something achieved by following one or more steps. In a systems engineering context, objectives and needs are considered equivalent. A ***goal*** is intangible, a situation which an organisation seeks to achieve by completing a series of objectives, that is, by satisfying a set of needs.

There are multiple sets of needs:

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

We propose that these are not ***requirements*** as they are not in the language normally used for requirements, particularly for statements from customers. Customers will state their wishes in their terms, and not as desired changes in your products, and not as requirements for new products that you might produce.

In agile processes, needs are the basis for the derivation of ***user stories***.

Content and Creation 2

Needs are not written in the language of your products, nor in the language of the technologies used in your products. Rather, they will be written in the language of the customers' products and business, or (for internal needs), written in the language of business or commercial objectives.

Examples of internal needs are:

- ***"Reduce the number of types of component that we buy-in, to increase our order volumes and purchase discounts."***
- ***"Reduce software maintenance by standardising drivers across all products, disabling features where needed."***
- ***"Achieve compliance with standard IEC 12345."***

Needs will be created by:

- Capturing from documents or spreadsheets
- Structured interviews, normally only for clarifications

It is important that needs are expressed in the natural language of the group that provides them, so these providers (customers and internal) will have confidence that their needs have been properly documented in the database.

Therefore, the validity of the database is primarily determined by the providers' acceptance of the needs inside it.

Using the providers' statements will inevitably mean that needs will be ambiguous and duplicated. It may mean that some needs are expressed in high-level terms whilst others are extremely detailed. There may be a need to decompose some needs into simpler statements, to achieve a consistency in the level of detail amongst the lowest-level needs. Since this means changing the information originally provided, each decomposition must be formally accepted by its provider.

We recommend that decomposition is the only manipulation of the needs that will occur. Identifying and resolving the other problems, such as ambiguity, imprecision and duplication will occur in the user requirements.

Structure and Organisation 3

Needs are hierarchical. As their main characteristic is the group that produces them, we propose a hierarchy of internal needs, and one hierarchy for each customer. Further, since needs are the basis for projects, we propose a sub-hierarchy of needs for each project. Hence the overall structure of needs will be:

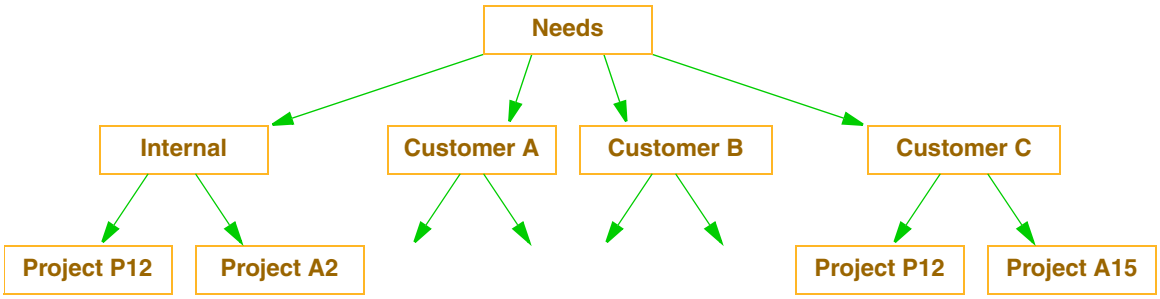


Figure 1: Overall Structure of Needs

If a project has needs from several groups, such as ***Project P12*** above, it will have several needs hierarchies, all linked from the project, as described in "Traceability" on page 4.

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

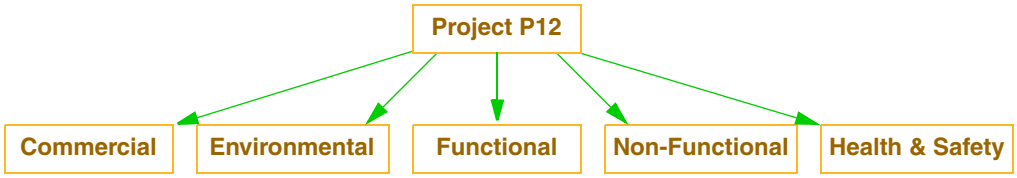


Figure 2: Organise Needs by Type

or they could be organised by engineering discipline:

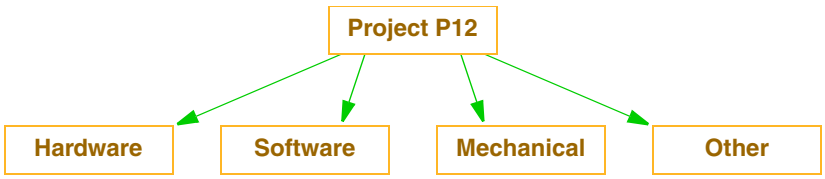


Figure 3: Organise Needs by Discipline

We recommend that needs are organised by type since we expect your customers and internal groups will naturally express needs in groups based on these types.

Traceability 4

Each need will be linked back to its project and will link to the user requirements created from it. For example:

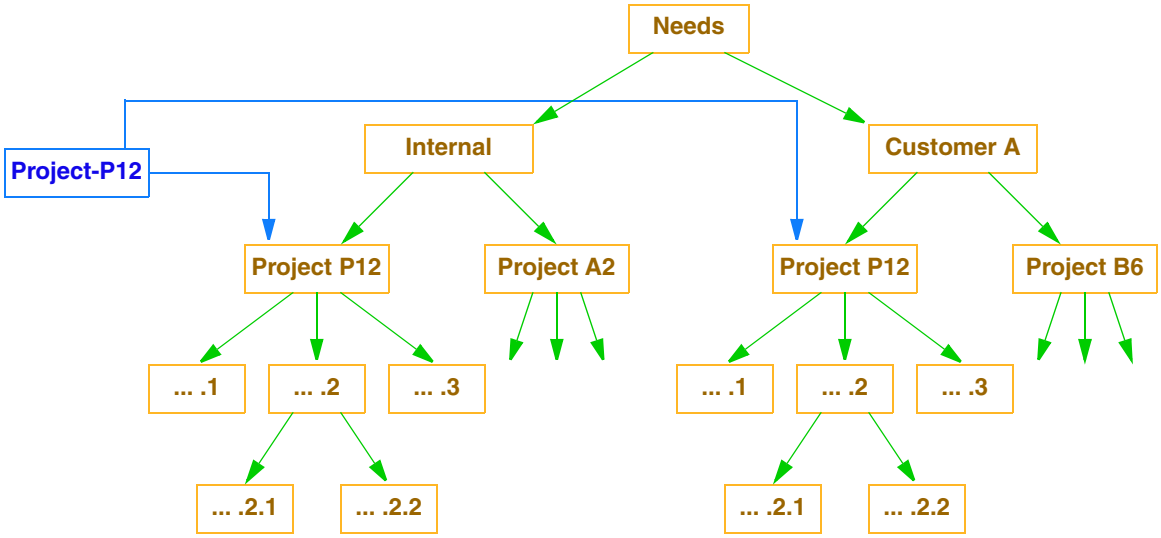


Figure 4: Projects and Needs

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

Lifecycle 5

An organisation will need to define its own process for the development of needs between the project team and the stakeholders. The end stage of such a process is that the needs are agreed by both groups, so the needs 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 for the development of the user requirements

Several steps can be imagined in such a process. The path that each need can take through these steps is called a

workflow. The position of each need in this workflow is recorded in an attribute in the need. We will use the name **Maturity** for this attribute.

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

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

We recommend the second of these approaches.

We suggest the following workflow (the **Maturity** attribute values are shown in parentheses):

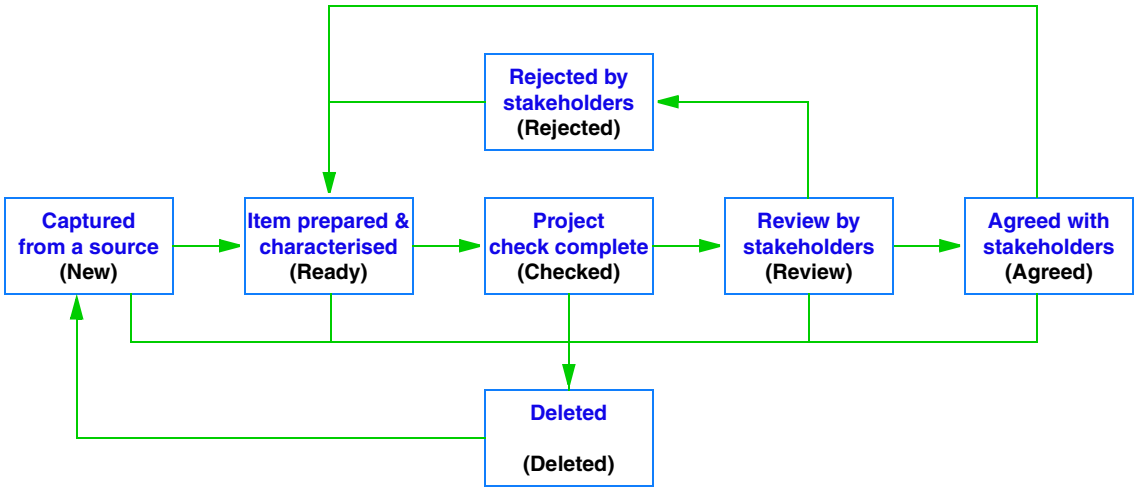


Figure 5: Workflow

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** needs may be

modified as the project progresses, perhaps after being baselined.

Database Representation 6

The same structure can be used for customer and internal needs, so we use one ***item type*** called **NEED**. This type will be hierarchical and auto-numbered to allow reordering of the hierarchies. The hierarchical numbers will be stored in the items' **Key** attribute, and will use a prefix **Internal** for internal needs and a suitable abbreviation for each customer's needs. The **Key** values' format is therefore:

organisation.project.hierarchical-number

such as:

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

where the names of the **.1**, **.2**, **.3** ... items in the hierarchy are based on the structuring criterion you chose. If this is type, the names of the **.1**, **.2**, **.3** ... items in every hierarchy of needs will be **Commercial**, **Environmental** and so on.

Needs will require attributes to:

1. Name the need, a convenient summary or shorthand
2. Specify the need, 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 notes for the need
4. Characterise the need:

   - **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**
   - **Type** : as one of: **Commercial**, **Environmental**, **Functional**, **Non-Functional**, **Health & Safety**

   which can be set at all levels, but must be set for the bottom-level needs
5. Record external comments for the need, such as from the review by stakeholders

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

Database Schema 7

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

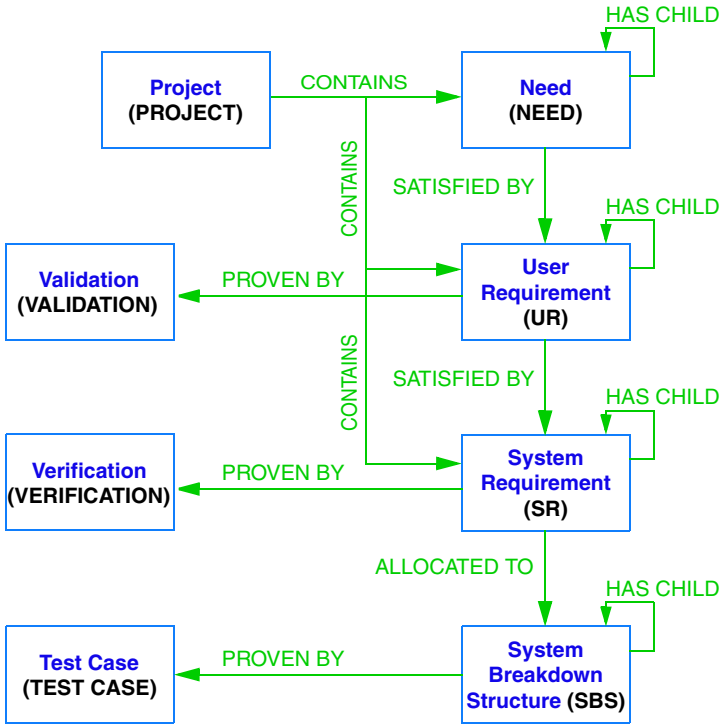


Figure 6: Database Schema

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

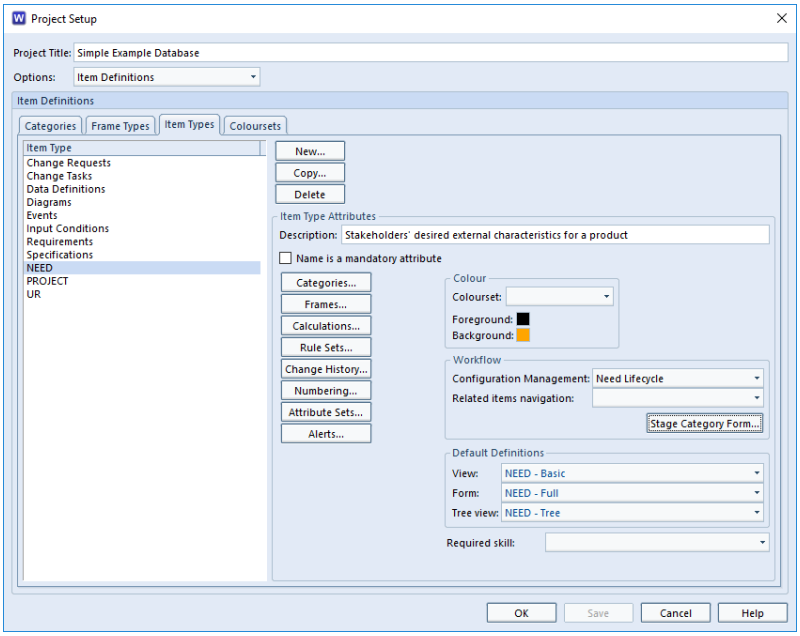
The ***item*** is the basic unit in a Cradle database. Each need will be an item of type **NEED**, defined as:

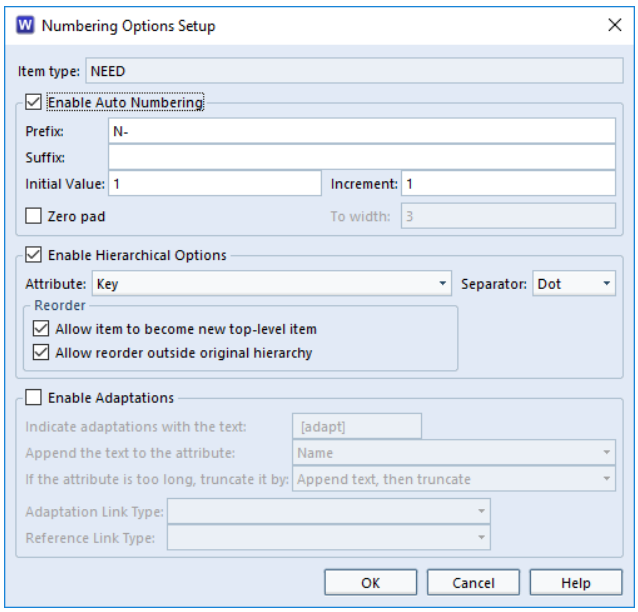
Table 3: Item Type Definition

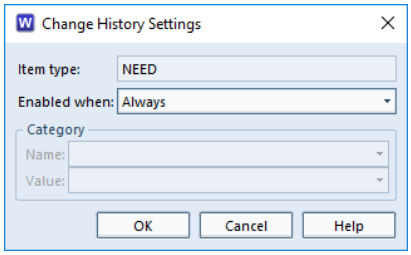
Name	Categories	Frames			Hier-archical	Adap-tations	Change History
		Name	Type	Auto-Numbering			
NEED	**Discipline** **Maturity** **Priority**	**CALCS**	**EXCEL (XLSX)**	Yes **N-*n*** *n*= 1,2,3...	Yes, in the **Key**	No	Yes
		COMMENT	Plain text				
		DIAGRAM	**VISIO**				
		DOCUMENT	**WORD (DOCX)**				
		FIGURE	**JPEG**				
		NOTES	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 needs have a unique identity that will not change even if the needs are reorganised in their hierarchies. We usually recommend that items are auto-numbered.
2. ***Hierarchical*** is enabled, with the hierarchical value stored in the **Key** attribute and with a dot (.) as the level separator in these hierarchical values. Enabling this feature means Cradle will provide the **New Child**, **New Sibling**, **New Hierarchy** and **Reorder** operations for needs, amongst other capabilities.
3. ***Adaptations*** is disabled. This feature is not







necessary for needs.

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

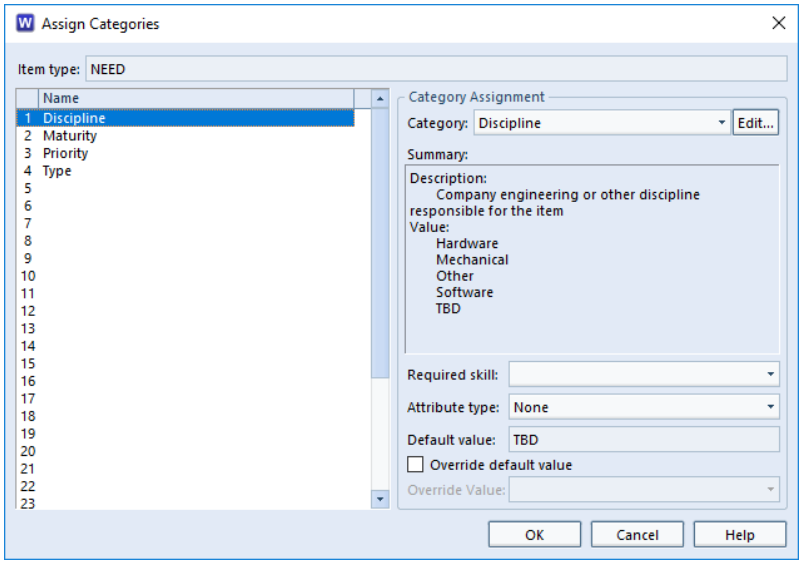
There are other options for each item type, such as:

- The colours used to show items in trees and ***Hierarchy Diagrams*** (HIDs) for graphical views of traceability
- The ***form*** to be used to show items individually
- The ***views*** used to display needs by default (unless a specific view is chosen), and to show the items 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 this facility.

Category Codes 10

Category codes are small data values often 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.



Needs can be characterised in many ways. We suggest you use categories for each of the structuring methods (see "Structure and Organisation" on page 3) that were not used to structure the needs, in this case engineering discipline, maturity and importance.

The categories defined in the schema for needs are:

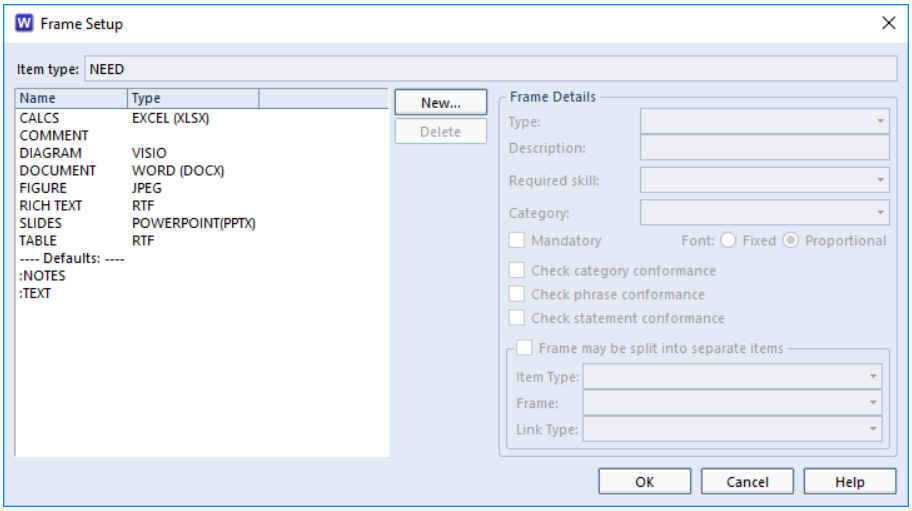
Table 4: Category Code Definitions

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)
Maturity	Single-value pick-list	**Agreed** **Checked** **Deleted** **New*** **Ready** **Rejected** **Review**	The values of the lifecycle for a need. This is also used to avoid deleting needs when their authors have removed them. Such needs may be re-introduced, so keeping them available saves time. Such needs may be linked to other items, so keeping them makes it easy to find items that link to redundant needs, so these items can be considered for removal. It also allows the justification for everything in the database to be traced back to needs that are accepted.
Priority	Single-value pick-list	**High** **Medium** **Low** **TBD***	The importance of the need to the user. In agile projects, this is the first determinant of the ordering of the feature backlog, so that the earliest sprints or iterations will implement those user stories that satisfy the highest priority needs.
Type	Single-value pick-list	**Commercial** **Environmental** **Functional** **Non-Functional** **Health & Safety** **TBD***	The type of the need. This is the type of externally-observable aspect of the product that the need describes. It helps to group needs that refer to the same type of product characteristic, and helps to determine how the needs will be satisfied, and by whom.

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 at a URL), and how to operate on the data (get, view, edit, check, print, set and so on).



We suggest a set of frames that should be able to manage any type of information in your needs, and to store comments from any external groups. You can ignore any frames that are not relevant to you.

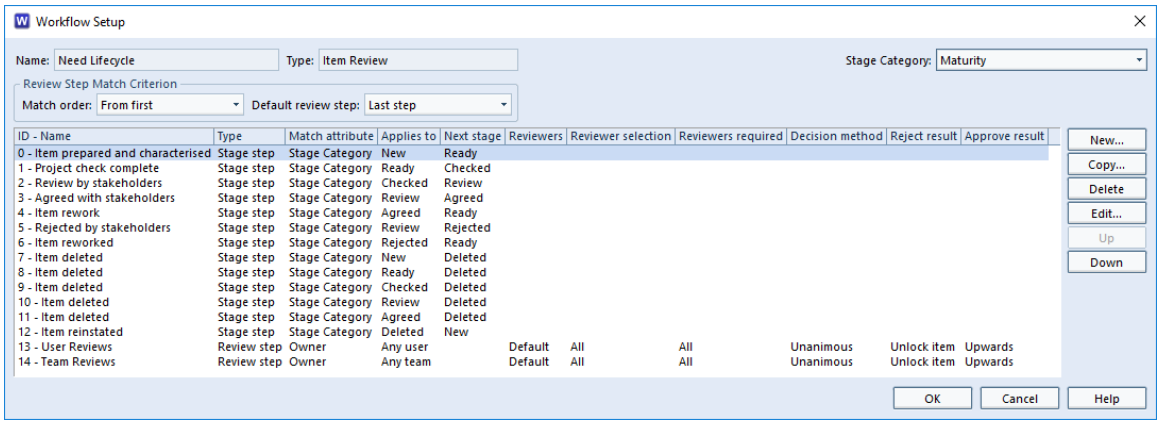
The definitions in

the schema of our suggested frames 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
RICH TEXT	**RTF**	The need 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 need 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 needs to implement the lifecycle defined in “Lifecycle” on page 4. This workflow



also includes the steps that are necessary to review needs 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.

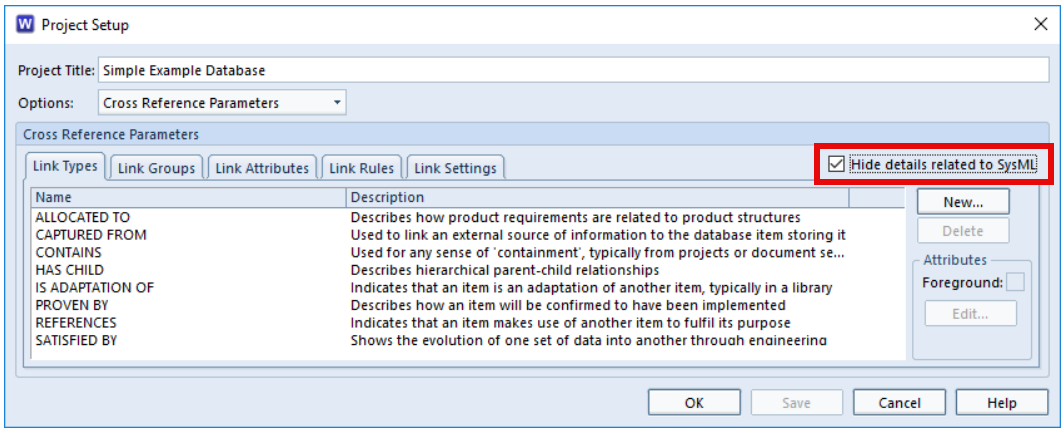
Cross references can store data in attributes inside the links themselves. These attributes are sometimes used to characterise, parametise, justify or simply explain the existence of a cross reference. These attributes can be free-format text or pick-lists of possible values. They can be used to select 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 7 are:

Table 6: Link Type Definitions

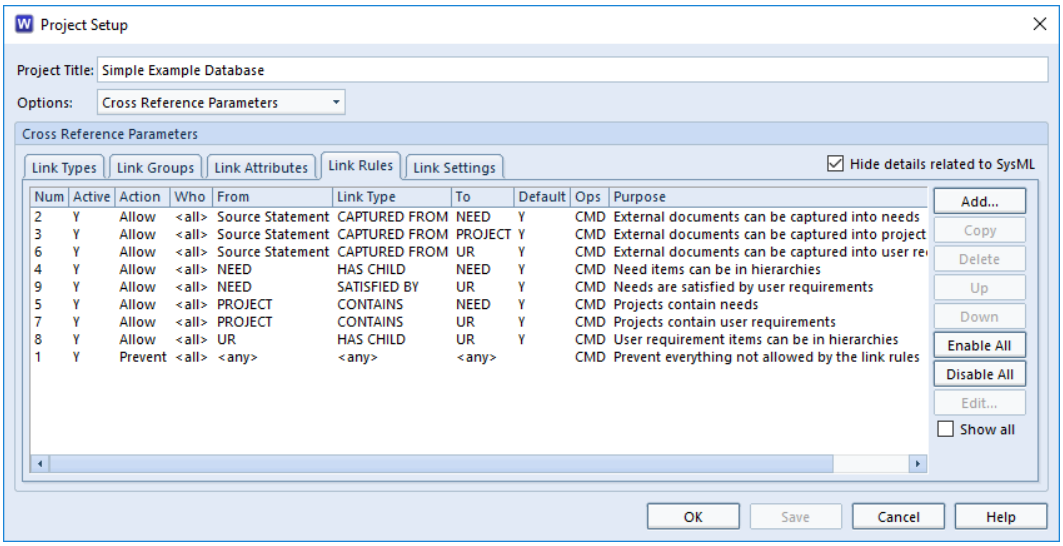
Name	Details
ALLOCATED TO	Describes how system requirements are related to product structures
HAS CHILD	Describes hierarchical parent-child relationships
PROVEN BY	Describes how an item will be confirmed to have been implemented
SATISFIED BY	Shows the evolution of one set of data into another through 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 7

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 needs will be auto-numbered, the most useful of these as-supplied navigations are those that sort linked items by their **Key** attribute. 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**