CQC SOLUTIONS

# Requirements Tracking

## Clarity Add-On TRC Module

Author – Paul J Schofield

# Contents

# Overview

In all but the most trivial of projects it is difficult to plan timescales and effort accurately until a reasonably fleshed-out design exists, as there is no firm base from which to estimate the work involved. Over the past decade an entire discipline of Requirements Engineering has evolved which can assist in this.

Clarity's Requirements Tracking add-on module implements many of the ideas behind Requirements Engineering, by allowing requirements to be typed, grouped, categorized, prioritized and linked into hierarchies. It provides facilities to demonstrate requirement coverage at all levels and generate coverage metrics and produce reports in the form of traceability and verification matrices. These highlight any gaps by showing where customer requirements have not been covered by designs, or designs by test plans and results. They also allow an audit of requirements and their implementation during system testing and acceptance.

Broadly speaking, Requirements Engineering splits into Requirements Analysis, Requirements Definition, Requirements Traceability, and Requirements Verification. Its chief aim in all of these is to introduce rigour by replacing qualitative judgments with quantitative metrics. From these it is possible to state how well a solution has met its intended aims, and mitigate any potential risks.

The Requirements Tracking module can also link to projects and/or tasks within the rest of Clarity allowing individual requirements, or groups of them, to be scheduled and costed if needed, although it can equally well be used stand-alone.

# Standards Overview

When developing solutions and systems, it is paramount that every specified requirement is met. In some industries this is also a critical safety issue, and Requirements Tracking becomes part of a larger certification process whereby a full audit trail of how contractual requirements have been implemented and tested is mandatory.

This is often covered by the generic standard IEC61508 for the design, construction and operation of electrical/electronic programmable systems. However many industries have their own derivations, for example:-

- DO-178B & DO-254 Avionics software and hardware standards.
- ISO 26262 Automotive standards, (currently draft).
- IEC 61513 Nuclear Industry for instrumentation and control for systems important to safety.
- CMMI Level 2 Capability maturity model standard.
- EN5012x/lEC 62425 for railway signalling systems.
- FDA 21 CFR for laboratory practices for conducting non-clinical laboratory studies.

All of the above standards are tailor-made for their industries but the common strand in all of them is to be able to track how the original requirements have been met, how they have been implemented and how they have been tested. Without this kind of evidence it is unlikely that any system will be accepted or certified.

Clarity's Requirements Tracking module may be used alongside any of the above industry standards. Initial requirements gathering can form the top level of a hierarchy and these can be decomposed and refined into as many levels as needed, down to test plans and test results. Even usage of a particular feature in the resulting system can be tracked so that ongoing maintenance can be tied back to the original requirements.

The example that follows comes from an avionics project, where the DO-178B standard applied.

# Requirements Tracking Examples

## Example 1

Initially, requirements come from a customer. They start out as a broad wish list which is then massaged into what we will call, for want of a better title, the Contractual Requirements Specification (CRS), as this is what all parties have signed up to. This can contain requirements of several types.

Formal requirements have a language all of their own and to avoid ambiguity the following is usually the first thing in the document:

**Standard Interpretations**

*Use of 'shall', 'should', 'must', 'will' and 'may' shall conform to the rules below.*

*The word **SHALL** in the text expresses a mandatory requirement that is binding on the supplier. Departure from such a requirement is not permissible without formal agreement between the Supplier and the Buyer.*

*The word **SHOULD** in the text expresses a recommendation or advice on implementing such a requirement.*

*The word **MUST** in the text is used for legislative or regulatory requirements with which both the Buyer and Supplier shall comply. It is not used to express a requirement of the specification.*

*The word **WILL** in the text expresses a provision or service by the Buyer or an intention by the Buyer in connection with a requirement of the specification. The Supplier is implicitly authorized to rely on such service or intention.*

*The word **MAY** in the text expresses a permissible practice or action. It does not express a requirement of the specification.*

*Text that is not explicitly tagged as a requirement shall be considered as information only.*

With that digested, here is a short example extracted from an avionics CRS on a DO-178B project at DAL C.

*Mandatory Requirement 1: The Engine Monitoring Unit (EMU) shall be capable of reporting temperatures redundantly, with a frequency not less than once every 5 seconds (0.2Hz), at each monitoring point.*

*Mandatory Requirement 2: The temperatures reported shall be smoothed over the refresh period to prevent jitter.*

*Optional Requirement 3: The Engine Monitoring Unit will be able to report temperatures in both degrees Celsius (ºC) and also in degrees Fahrenheit (ºF).*

*Mandatory Requirement 4: The Engine Monitoring Unit shall be capable of generating an alert for invalid temperature measurements.*

*Mandatory Requirement 5: The Engine Monitoring Unit shall be capable of reporting vibration frequencies redundantly, with a frequency not less than once every 5 seconds (0.2Hz), at each monitoring point.*

*Mandatory Requirement 6: The vibration frequencies reported shall be smoothed over the refresh period to prevent jitter.*

Traditionally, from these initial requirements a set of System Design Requirements will be created that defines how the contractual requirements will be met. If we take the first 4 requirements we might end up with something like this:

*Design Requirement 1: Dual thermocouples shall be sited equidistant from each monitoring point.*

*Design Requirement 2: Each thermocouple shall be connected to an A/D converter which provides a digital value to the EMU.*

*Design Requirement 3: The EMU shall poll each pair of digitized temperature inputs for all*

*monitoring points at a frequency of at least 0.2Hz.*

*Design Requirement 4: The EMU shall scale the pair of values for each monitoring point and verify they are consistent (within ± 1% of each other).*

*Design Requirement 5: If the values are consistent:*

- *The values shall be stored.*
- *A history of ten measurements shall be maintained for each input for each monitoring point.*
- *The average of each sensor's measurements shall be calculated.*
- *The average of the average calculated in the previous step shall be the value reported.*
- *The number of inconsistent values shall be reset to zero.*

*Design Requirement 6: If the values are inconsistent:*

- *The number of inconsistent values shall be incremented with a ceiling of 10.*
- *The average of each sensor's inputs shall be calculated and compared to its current value.*
- *The sensor with the value closest to its previous average shall have its value stored and this sensor's new average shall be the value reported*
- *If the number of inconsistent values = 10 generate an alert*

*Design Requirement 7: The EMU shall report temperatures in degrees Celsius only. It is suggested that cockpit fascia are calibrated in both Fahrenheit and Celsius since the relationship between the two scales is linear.*

The above list contains requirements that stem directly from the CRS, some of them software related, some of them not.

In this small example it is simple to see how the lower level requirements relate to the higher ones. In a real world example it isn't so easy. Things might start out in an organized way, but changes need to be managed, and when there are several layers (not just two, as above) management can become almost impossible.

The next chapter provides details of the facilities provided within the Clarity Requirements Tracking module, and how the above example can be managed. The following example illustrates what can go wrong when rigorous analysis has not been performed.

## Example 2

All too frequently the analysis and definition phases are truncated because of short timescales, lack of budget or simply a desire to provide a quick fix to a problem. This is almost always a mistake. At best it ends in protracted discussions and cost overrun. At worst it ends in litigation, with both parties to the contract claiming that the other is at fault for not providing a complete and unambiguous set of requirements or delivering a solution that meets them.

Real world examples abound, but as a simple (and true) example of what can go wrong, a hospital decided to move from a clerical appointments system to one that was computer based. The merits of doing so were huge. It (should have) allowed patients to ring in and arrange appointments to suit them and also allow the patient's records to be linked to the appointment and be available on the day. Things progressed through acceptance trials and the system was about to go live.

At this stage a major flaw emerged: The appointments diary could be traversed in a forward direction, but not backwards. The high level requirements stated that receptionists should be able to page through the diary and confirm an appointment whilst the patient was on the phone. Frequently though, the receptionists hit the 'Page Down' key once too often, but were then unable to 'Page Up' to go back to a previous day. It seems obvious in retrospect that paging through a diary ought to allow for going backwards as well as forwards, but this wasn't stated explicitly and the developers took the easy way out and implemented a serial (forward) traversal.

# Module Overview

## Functionality provided

The Requirements Tracking module provides the following:

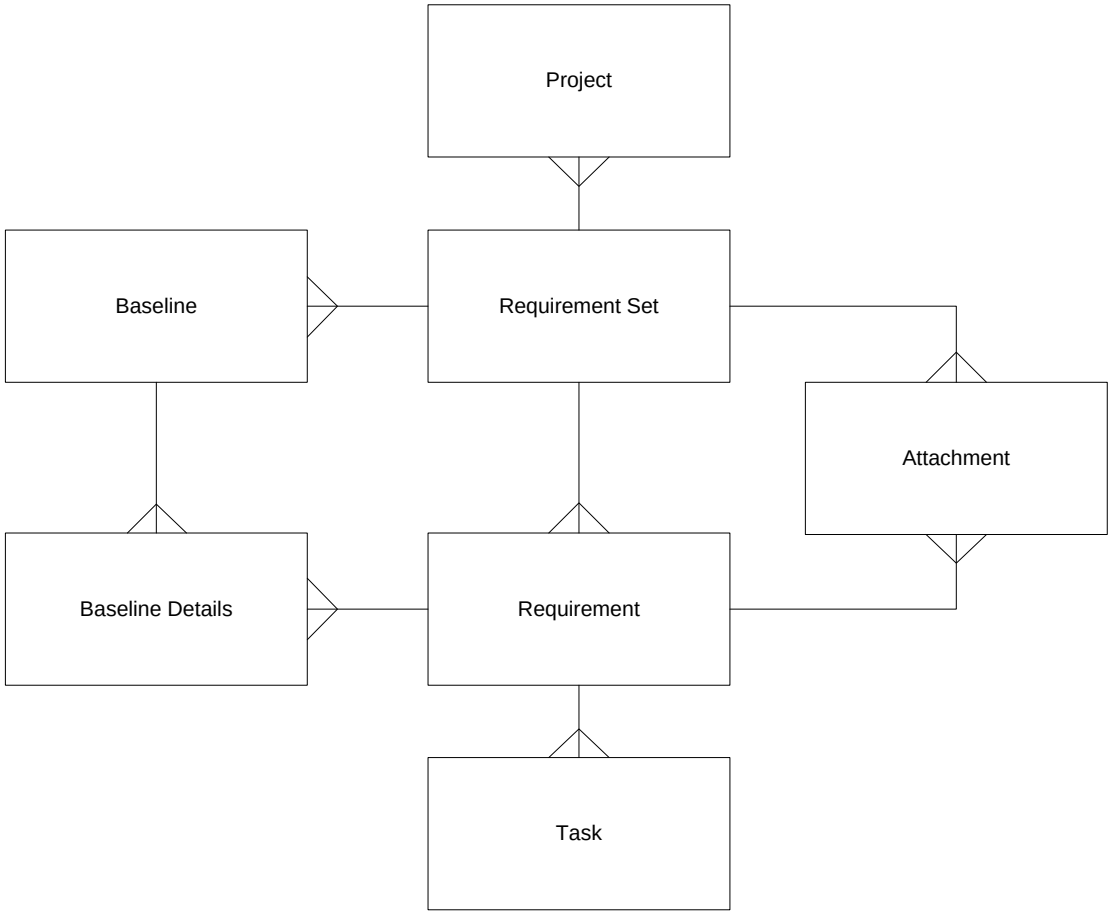
- Entry and versioning of sets of requirements
- Attachment of documents to individual requirements
- Organization of requirement sets into hierarchies
- Creation and versioning of requirement set baselines
- Top down and bottom up requirement traceability from any point in a hierarchy
- Requirement statistics and coverage metrics
- Historical views of requirement versions
- Historical views of requirement baselines
- Options to enforce strict hierarchical requirement coverage (the default)
- Options to disable requirement versioning (enabled by default)
- User configurable lookup lists for requirement sources, types and categories
- Crystal reports showing RTM and RVM
- Leverage of other Clarity functionality

## Leveraging Clarity Functionality

Requirement sets can be linked to projects and individual requirements can be linked to tasks. These links allow the rest of Clarity's standard functionality (resourcing, scheduling, costing etc) to be inherited:

- Clarity's full security model is available out of the box. Requirement Sets or individual requirements can be secured using standard Clarity functionality.
- Requirement sets or individual requirements can be linked to tasks, projects, programmes and portfolios
- Full resource management linked to projects, programmes and portfolios
- Ideas capture, strategic alignment and the ability to run scenarios at the portfolio level
- Accurate real-time metrics for project, programs and portfolios, such as estimated time to completion (ETC), costs and financial plans
- Ability to partition instances so that outsourced work can be monitored and reported against the overall programme or portfolio, without the outsource company having sight of commercially sensitive materials
- Sophisticated workflows that enable business processes such as NPD Stage Gate™ and/or Prince II methodology including notification by Email or SMS when a user is required to take action.

## Entity Relationships




Clarity works with Requirement Sets, each of which consists of individual requirements organized into a hierarchy.

Requirement sets can be linked to one or more projects, and individual requirements can be linked to one or more tasks.

Requirement sets and individual requirements may have attachments (documents, drawings, notes etc) uploaded against them.

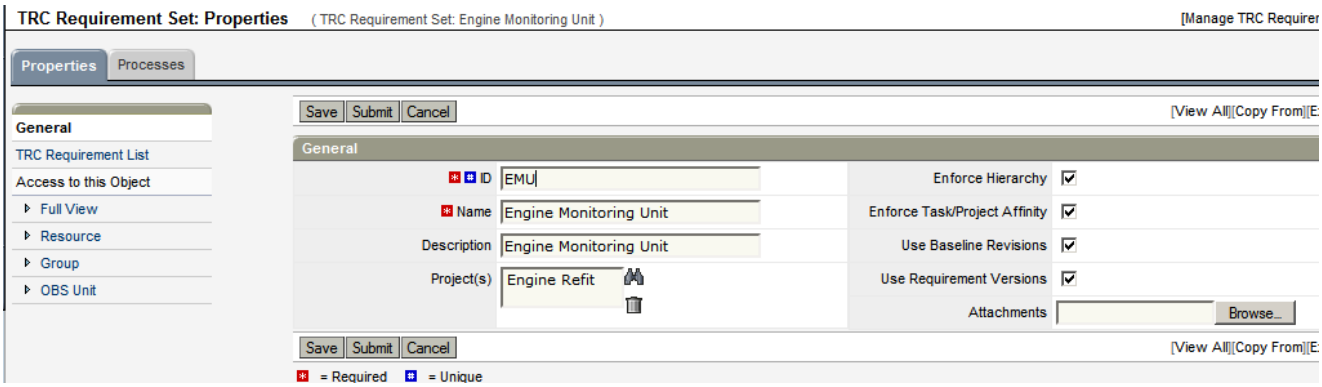
Individual requirements can be versioned, and requirements sets can be base-lined as a whole or partially at any time to provide a timed snapshot. Baselines can also be versioned.

Reports can be run against these baselines starting at any level in the hierarchy to provide coverage matrices.

# Clarity Tracking Example

## Overview

The first thing is to create the requirement set:

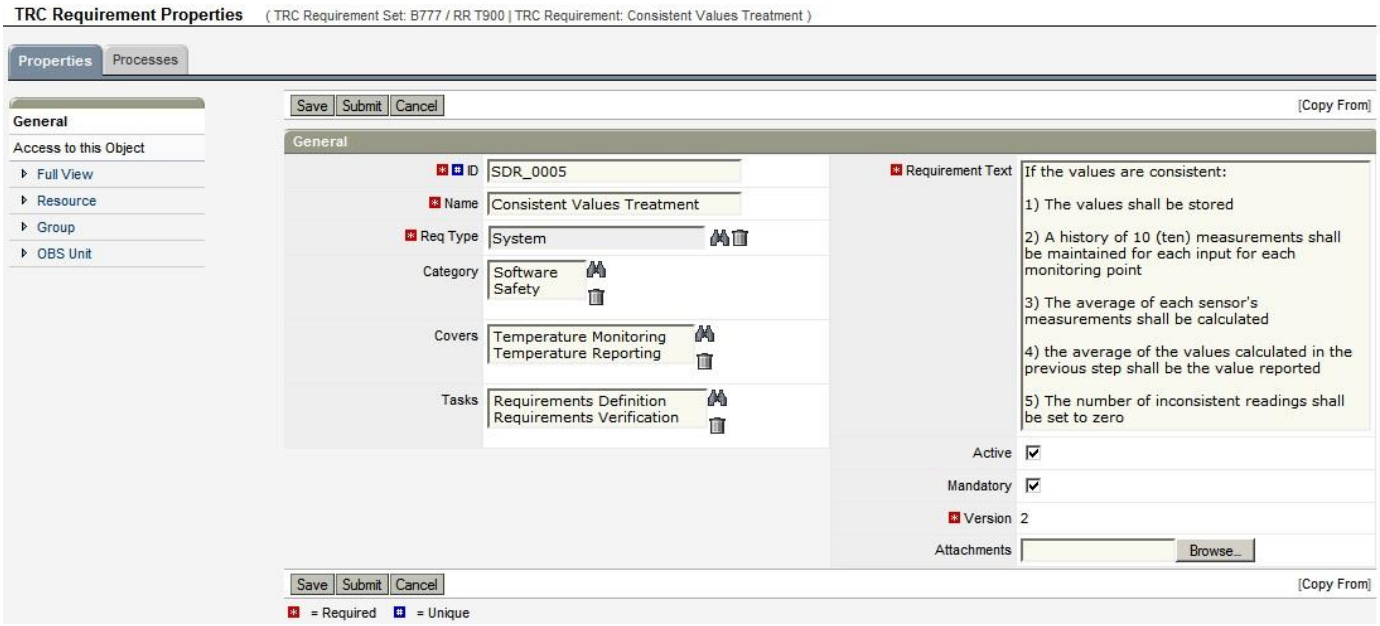



ID, Name and description are standard Clarity fields. The remaining fields are filled in as follows:

- Project(s) specifies the projects (if any) that this requirement set is linked to in the rest of Clarity. Here we have selected just one. If we weren't interested in using other Clarity features and just using the Tracking module stand-alone we would have left tis field blank.
- Enforce Hierarchy (ticked by default) restricts how we cover requirements. In this example we do not want to skip one or more levels in the hierarchy when we are covering higher level requirements by lower level ones. Unticking this box would allow this.
- Enforce Task/Project Affinity (ticked by default) restricts the tasks that may be selected for linkage to individual requirements to the projects specified in the Project(s) field of the requirement set definition .
- Use Baseline Revisions and Use Requirement Versions (ticked by default) allow a history of baselines and requirements to be maintained. If these are un-ticked only, the current entities are stored.
- Attachments allow up to 10 (expandable if needed) documents, drawings etc to be stored against the requirement set.

In this example we have defined 5 levels to the requirement hierarchy: Customer > System > Design > Test > Test Result. The number of levels in the requirements hierarchy is arbitrary. The DO-178B standard mandates going all the way down to tests and results. It is possible to go even further and trace how each requirement is actually used (if at all). It all depends how deep a customer needs to go.

Once these levels are set up we can create the individual requirements:

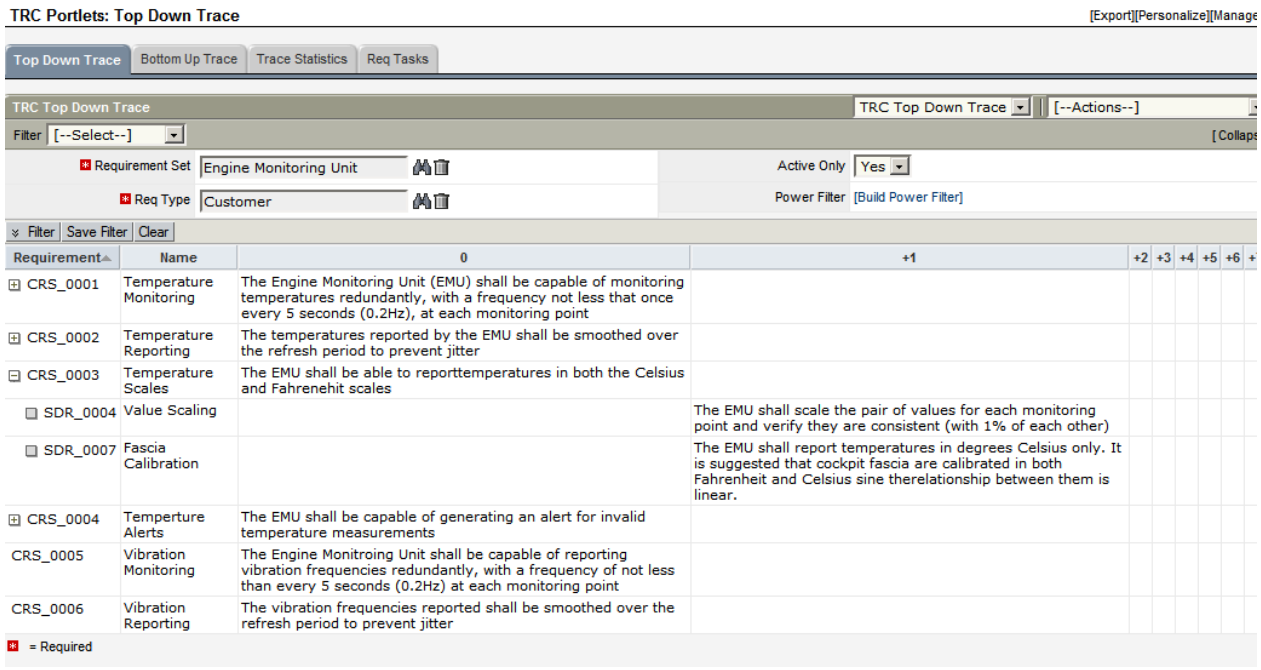



The fields are filled in as follows:

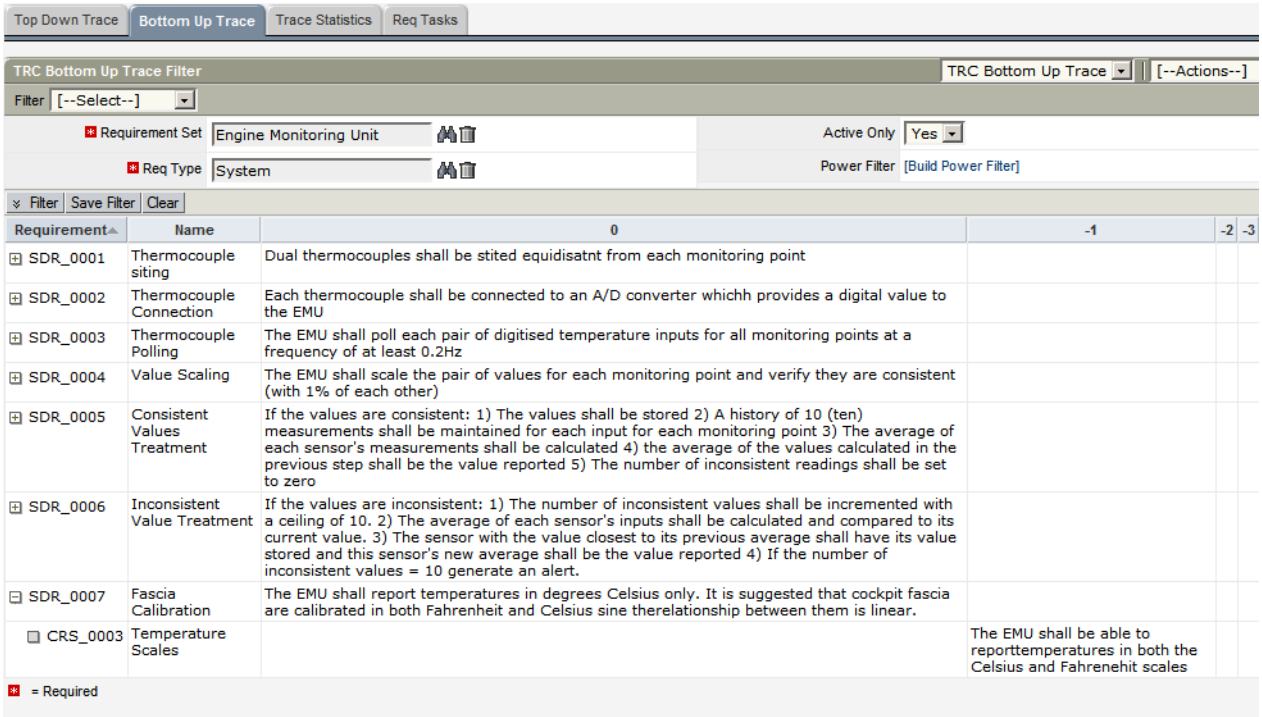
- Requirement type specifies its provenance, which, broadly, corresponds to its level in the requirements hierarchy. Customer requirements will be at the top, test results will be at the bottom. Here we have a system definition requirement that belongs to the 2nd level down in the structure.
- Category specifies how the requirement relates to the solution, such as whether it's a hardware, software or legal requirement.
- If, as here, this requirement covers one or more higher level requirements, these are specified here. This requirement covers the first two customer requirements.
- This requirement is linked to two tasks on the project attached to the requirement set.
- Text is the text of the requirement.
- Active (ticked by default) allows a requirement to me made inactive (either temporarily or permanently) if needed for any reason.
- Mandatory (ticked by default) allows ‘nice to have’ requirements to be included in the requirement set.
- Version is incremented each time the requirement is saved (if versioning was enabled in the set).

Once the hierarchy is set up there are several portlets that can display the coverage, statistics and tasks for the requirements:

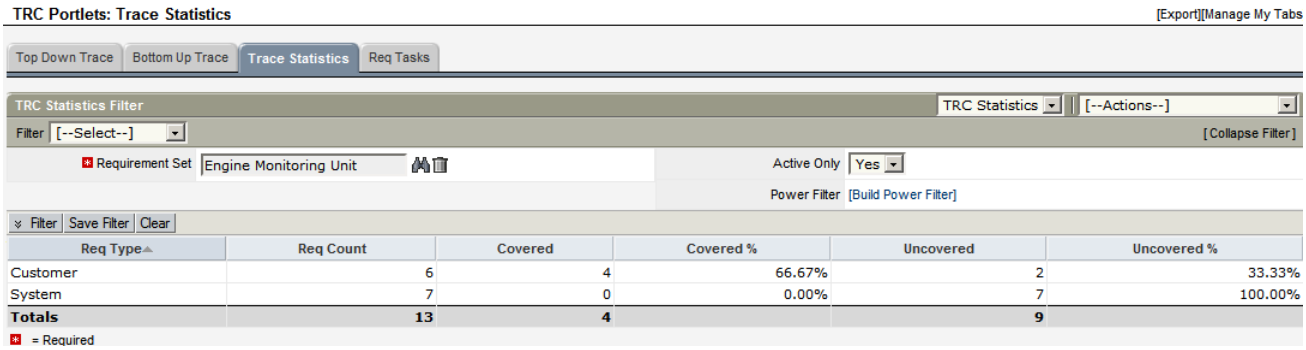
This is the Top-Down RTM (Requirements Traceability Matrix). We’ve selected Customer Requirements as the starting point and expanded the CRS_0003 (Temperature Scales) requirement to show it is covered by two lower level requirements:




We can reverse the focus and display the bottom-up tree. Here we've selected the System Definition Requirements as the starting point and expanded the SDR_0007 (Fascia Calibration) requirement to show that it covers a single higher level requirement.

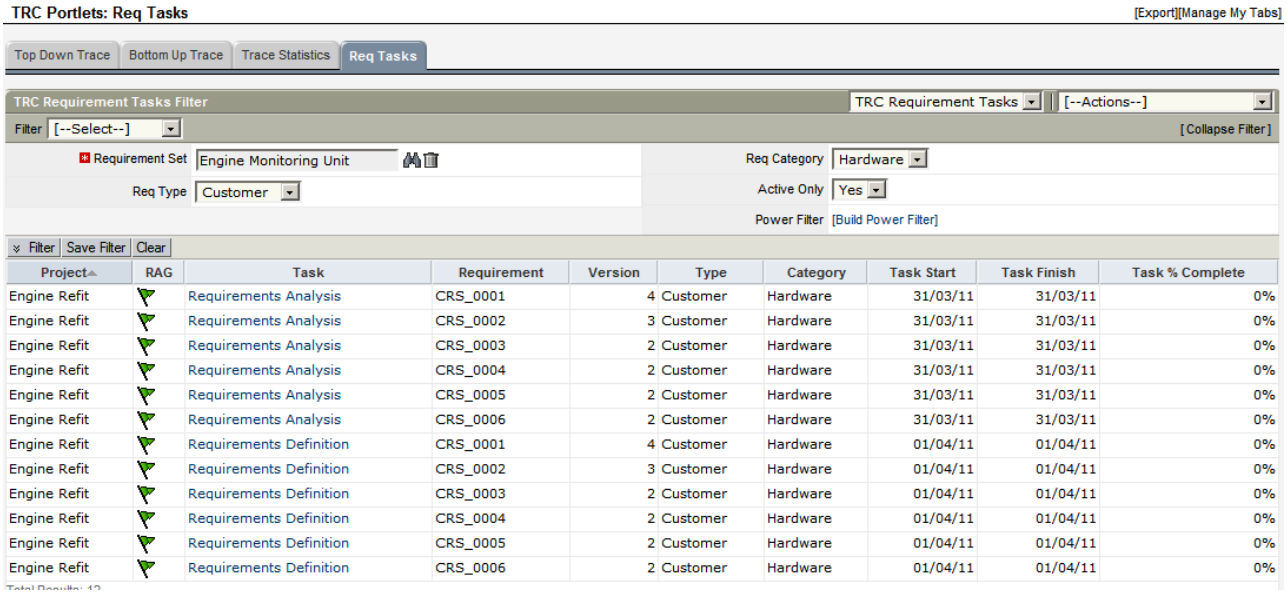



A third portlet shows how well we are doing in our overall requirements gathering:

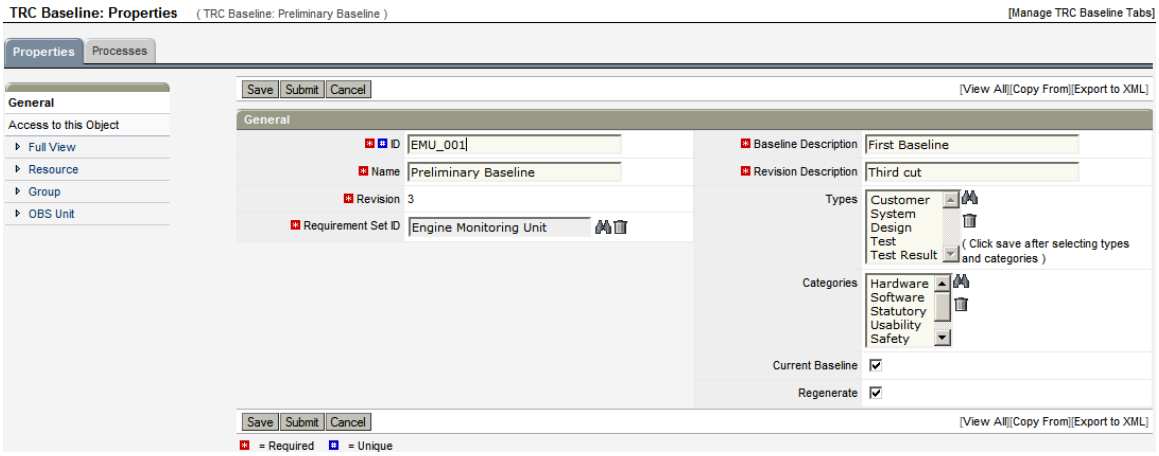



It shows we have a bit of work to yet.

The final portlet lists all the requirements, together with each task they are linked to, and provides drill-downs to them. Here we have limited the display to Customer requirements categorised as ‘Hardware’:




We can generate baselines for the requirement set. This example selects all requirements:




Once we have baselines we can generate reports. Appendix B is the RTM (Requirements Traceability Matrix) starting at the top level.

# Appendix A – Tool Comparisons

Here is a comparison of what Doors and Reqtify (the main packages used for tracking) do, and how they achieve it. The final column shows how the same functionality is provided in Clarity:

| Feature | Reqtify | Doors | Clarity |
|---|---|---|---|
| Requirements Entry | Via complete Word documents | By entering them directly into the Doors database via its UI | By entering them directly into a Clarity via the UI |
| Requirements Linking | By prefixing/suffixing requirements in the documents with tags that Reqtify then matches up | Automatic as part of the UI | Via lookup on higher level requirements as the requirements are created/edited |
| Requirements visibility | Clicking on a requirement opens up the document that contains it in Word, and a context search then locates the tag clicked on. | Already visible in the UI | By selection from a list or by clicking on a link in the portlets. |
| Requirement Hierarchy | Documents are linked in the desired tree in the UI | Automatic in the UI | Created as needed when requirements are linked in the UI. |
| Upstream/Downstream traceability | Detailed visibility via reports. Quicker (more limited) views via the UI. | Automatic in the UI and also via reports | Via portlets and via reports. Export to Excel also possible. |
| Filtering of requirements | By category | By category and source | By category and source and possibly others |
| Uncovered requirements | Visible in the UI and in reports | Visible in the UI and reports | Visible in portlets and in reports |
| Derived Requirements | Entered in the document as any other requirement but without the suffix tag | Entered via the UI with a blank ‘box’ to the left | Entered as any requirement would be, but without a link to a parent |
| Broken references | Possible if tags are entered incorrectly | Not possible given the way the UI works. | Not possible to set up, but care need to ensure a parent isn’t deleted if it still has children assigned. |

# Appendix B – Requirements Traceability Matrix

Requirements Traceability Document

Downstream Traceability Matrix

for

Engine Monitoring Unit

All Baselined Requirements

Generated using baseline Preliminary Baseline revision 3, Dated 05-April-2011

On 05-Apr-2011

Engine Monitoring Unit
Downstream Traceability Matrix

**CRS_0001 (Customer)** **Temperature Monitoring v 4** **13-Mar-2011**

The Engine Monitoring Unit (EMU) shall be capable of monitoring temperatures redundantly, with a frequency not less that once every 5 seconds (0.2Hz), at each monitoring point

Covered by SDR_0006 v 2 SDR_0001 v 2 SDR_0002 v 2 SDR_0003 v 2 SDR_0005 v 2

**CRS_0002 (Customer)** **Temperature Reporting v 3** **13-Mar-2011**

The temperatures reported by the EMU shall be smoothed over the refresh period to prevent jitter

Covered by SDR_0006 v 2 SDR_0002 v 2 SDR_0004 v 2 SDR_0005 v 2

**CRS_0004 (Customer)** **Temperture Alerts v 2** **13-Mar-2011**

The EMU shall be capable of generating an alert for invalid temperature measurements

Covered by SDR_0006 v 2

**CRS_0005 (Customer)** **Vibration Monitoring v 2** **13-Mar-2011**

The Engine Monitroing Unit shall be capable of reporting vibration frequencies redundantly, with a frequency of not less than every 5 seconds (0.2Hz) at each monitoring point

This requirement is not covered in this baseline

**CRS_0006 (Customer)** **Vibration Reporting v 2** **13-Mar-2011**

The vibration frequencies reported shall be smoothed over the refresh period to prevent jitter

This requirement is not covered in this baseline

Engine Monitoring Unit
Downstream Traceability Matrix

**SDR_0001 (System)** **Thermocouple siting v 2** **13-Mar-2011**

Dual thermocouples shall be stited equidisatnt from each monitoring point

This requirement is not covered in this baseline

**SDR_0002 (System)** **Thermocouple Connection v 2** **13-Mar-2011**

Each thermocouple shall be connected to an A/D converter whichh provides a digital value to the EMU

This requirement is not covered in this baseline

**SDR_0003 (System)** **Thermocouple Polling v 2** **13-Mar-2011**

The EMU shall poll each pair of digitised temperature inputs for all monitoring points at a frequency of at least 0.2Hz

This requirement is not covered in this baseline

**SDR_0004 (System)** **Value Scaling v 2** **13-Mar-2011**

The EMU shall scale the pair of values for each monitoring point and verify they are consistent (with 1% of each other)

This requirement is not covered in this baseline

**SDR_0005 (System)** **Consistent Values Treatment v 2** **13-Mar-2011**

If the values are consistent:

1) The values shall be stored

2) A history of 10 (ten) measurements shall be maintained for each input for each monitoring point

This requirement is not covered in this baseline

3) The average of each sensor's measurements shall be calculated

4) the average of the values calculated in the previous step shall be the value reported

5) The number of inconsistent readings shall be set to zero

**SDR_0006 (System)** **Inconsistent Value Treatment v 2** **13-Mar-2011**

If the values are inconsistent:

1) The number of inconsistent values shall be incremented with a ceiling of 10.

2) The average of each sensor's inputs shall be calculated and compared to its current value.

This requirement is not covered in this baseline

3) The sensor with the value closest to its previous average shall have its value stored and this sensor's new average shall be the value reported

4) If the number of inconsistent values = 10 generate an alert.

**SDR_0007 (System)** **Fascia Calibration v 2** **13-Mar-2011**

The EMU shall report temperatures in degrees Celsius only. It is suggested that cockpit fascia are calibrated in both Fahrenheit and Celsius sine therelationship between them is linear.

This requirement is not covered in this baseline