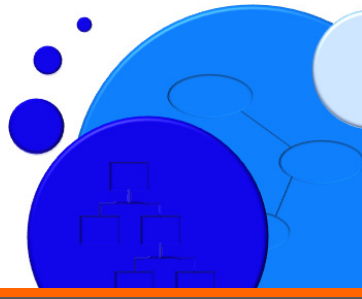
Cradle-7
From concept to creation...

3SL

Traceable and Tracked Document Management from Cradle

RA001/07 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

Most projects use documents, as sources of information or as outputs to customers or to other groups. This includes projects using agile processes. It is also common for some documents to have an important purpose or role, for example to:

- Define the customer's technical and commercial requirements
- Define the acceptance criteria
- Record the system design
- Record the verification or validation of some or all of the system

While these documents may be produced many times, some specific instances of the documents often have a special significance, shown by one or more of:

- A particular issue number
- Stamping with a marking
- Bearing a seal
- Signature(s) on printed copies

The legal and/or commercial significance of documents forces most projects to have a document-centric process. The effect is that regardless of the extent that the data-centric methods of requirements management and systems engineering are being used in a project, the project is typically managed by the issue states of its key input and output documents.

Subject

This paper considers how to reconcile the document-based demands of the customer and the project's own organisation with the data-centric perspective that is natural for requirements management and systems engineering techniques.

In particular, how can the problems traditionally associated with documents be reconciled with the needs for formal traceability and tracking inherent in formal systems engineering processes.

The Needs 1

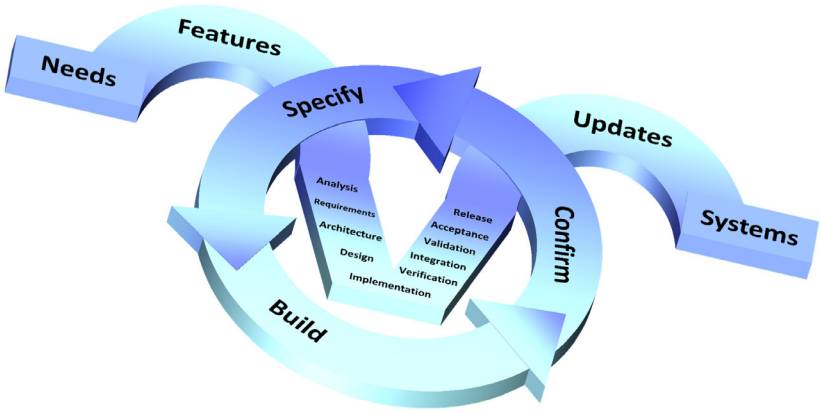
Most projects use processes in which information, such as user stories, requirements, use cases or verifications, are managed as individual items that are linked to each other. However, the relationship with the customer will be based on documents, each containing thousands of items. This creates some fundamental needs, such as to:

- Report the items that have been created or updated from a specific instance of an input document
- List differences between the items loaded from two input documents, or two issues of a single document
- List the documents that have updated specific items
- Report which items have been published into a given instance of an output document
- List differences between the items published into two output documents or two issues of a single document
- List all documents in which items have been published

For example, consider the following customer questions:

- ***Which versions of the requirements are in this SRD?***
- ***Which requirements have changed in the SRD issues?***
- ***What are the differences between the designs in the original and current issues of the SDD?***
- ***Requirement 1.2.3.4 needs to change, so which documents need to be re-issued?***

Some may say that agile projects do not need to manage documents as do projects using phase-based processes. Agile projects ***do***, or ***should***, have the same need. Most agile processes are based on a ***feature backlog*** expressed as ***user stories*** (the terms vary), sometimes grouped into ***epics*** (again, the terms vary).



But user stories are not requirements. They are convenient summaries of one or more user needs. They contain too little detail to be validated by anything other than a general user experience functional test. So each user story will, or should, link to a collection of user requirements that are traced back to the users' original needs in the input documents.

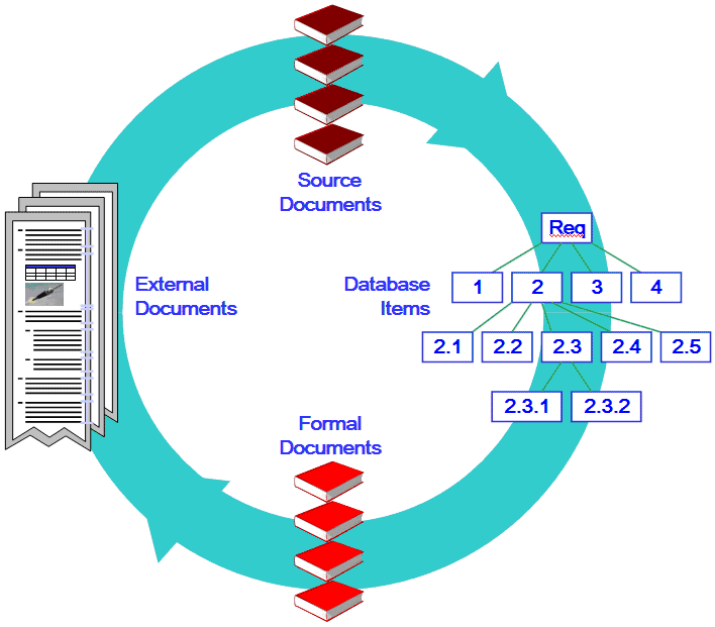
The essence of agile is short-cycle times, variously called ***sprints*** or ***iterations***, and the idea that a ***viable system*** must exist at the end of ***every*** iteration. Nonetheless, each iteration is, or should be, a full V-lifecycle, with analysis, design, implementation and test activities.

Each iteration needs traceability, coverage and full documentation. Their short timescales create an urgent need to automate the production of this documentation.

The Solution 2

Cradle meets these needs with ***source documents*** and ***formal documents***:

- Source documents are external documents that are split into their component paragraphs, tables, table rows and so on, that are loaded into the database as items, such as requirements, references, acronyms, standards or acceptance criteria
- Formal documents are documents published from the database to fulfil a specific need of an external group, typically a customer



Cradle has facilities to process and correlate the documents and database, cyclically, if needed.

This allows a project to:

1. Work in a data-centric way:
   - When data is stored once and used many times
   - Where the database is a ***single point of truth***
2. Interface fully and seamlessly with its document-centric environment
3. Track every part of every input document as it evolves through the database
4. Track the source of every part of every output document
5. Guarantee the consistency of all published documents

The Benefits 3

Cradle allows a project to avoid the problems associated with producing documentation manually:

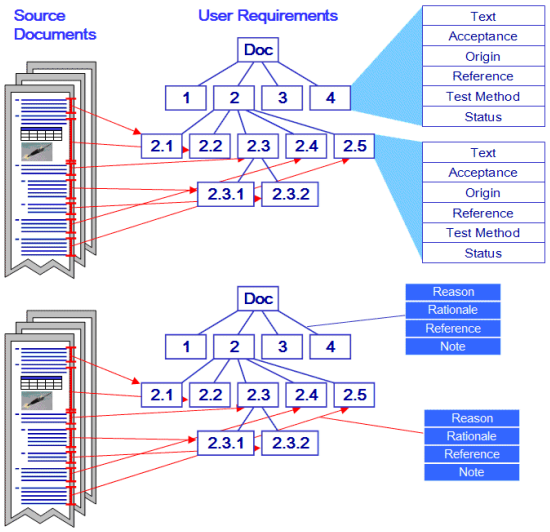
- ***Incompleteness***: documents cannot be guaranteed to be complete, and attempts to ensure completeness are time consuming, costly and invariably not done
- ***Inconsistency***: the same information appears in many documents and attempts to ensure consistency are time consuming, costly and invariably not done
- ***Obsolescence***: the underlying information changes rapidly and the time taken to manually prepare, quality assure and cross check documents means they become obsolete soon after release
- ***Irrelevance***: the obsolescence of manually produced

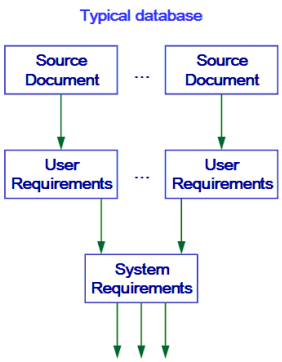
documents makes them irrelevant, they are ignored and so lose the authority intended by their authors

Source Documents 4

Any number of documents containing requirements, standards, regulations or other inputs can be loaded into a Cradle database using the **Document Loader** tool, that:

- Creates a source document in the database
- Copies the external document into the source document, thereby protecting it from change
- Divides the source document into ***source statements***, one for each section, paragraph, table (row, cell or table) being loaded
- Creates a database item for each source statement
- Links the items into a hierarchy corresponding to the hierarchy of document sections and subsections
- Links each item to its originating source statement, that is, to its original text in the source document





Cradle keeps a detailed record of the database items that are associated with each version of each source document.

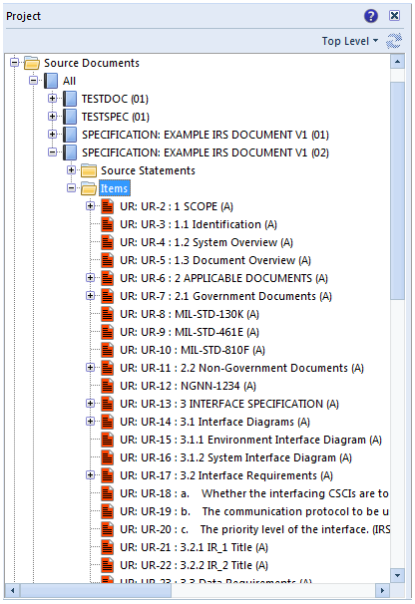
When a new version of a source document is loaded, **Document Loader** highlights the differences between it and the latest version in the database. This allows you to decide if the new document should be loaded.

If the new document version is loaded, Cradle repeats the above process, updating database items, creating a new set of links, and recording the instances of each database item that correspond to the new version of the source document.

Impact analyses from these changes allow an assessment to be made of the acceptability of new source document versions. Unacceptable new versions can be deleted pending discussion with their authors, causing Cradle to revert to the previous version of that document.

Source document management therefore means:

1. Any number of source documents can be captured
2. Any number of versions of each source document
3. Each source document is cross referenced to database



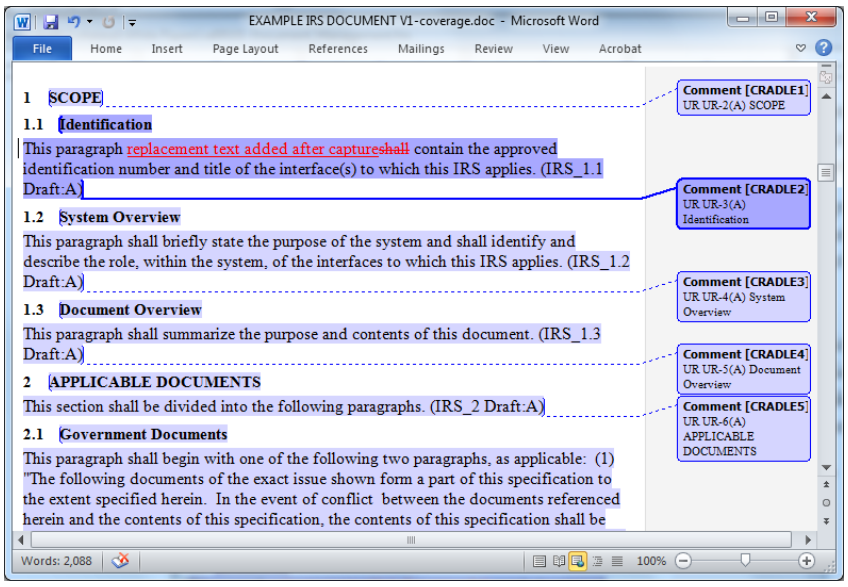
items created, or updated, by it

4. Records are kept of which items are associated with each version of each source document:
   - That can be viewed from items to documents
   - That can be viewed from documents to items
5. Source document versions can be compared
6. The sets of database items associated with source documents' versions can be compared
7. Source document versions can be deleted, which:
   - Reverses all changes to database items
   - Deletes that source document version and the cached document
   - Reverts to the previous source document version
   - Reverts to the set of cross references associated with the previous version of that source document

Collectively, these features create a comprehensive facility allowing all aspects of all source documents, and their updates, to be tracked, managed and related to the rest of the project's information.

Customer Compliance 5

Many projects operate within strict regulatory frameworks in which it is vital to demonstrate full compliance with customer documents. The source document mechanism also meets this need by providing three types of ***coverage analysis*** of any source document:



1. Which parts of the source document have, and have not, been captured into the database
2. Where is each part of the document in the database, shown inside comments
3. Show all differences between the text in the document and the text in items in the database, in red with underline and strikethrough markup

Use this to prove:

- Which part(s) of the documents have been captured
- Where each document part is stored in the database
- That the customer's document's content has not been changed once loaded into the database, or has only been changed in ways agreed by the customer

Since all further compliance reports are based on the database items that contain the customer's information, these document compliance views are the vital first step, by confirming the validity of these database items.

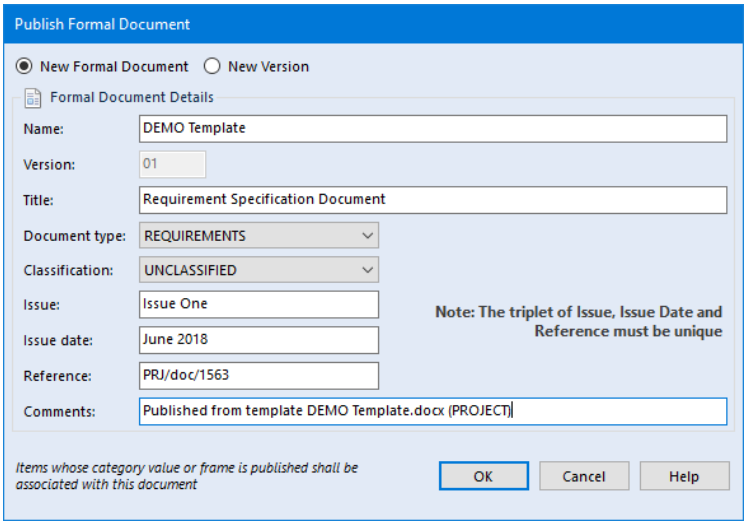
Formal Documents 6

Most projects produce their significant, formal, output in documents. Typical examples are:

- **CONOPS** : Concept of Operations
- **SRD** : System Requirements Document
- **SDD** : System Design Document
- **SSDD** : Subsystem Design Document
- **IFS** : Interface Specification

Cradle's **Document Publisher** tool is used to generate documents from a database. The form, structure and content of a document is described in a ***template*** which is itself a document that contains references into the database to retrieve the information to be published. One template is created for each style of document that your project will produce.

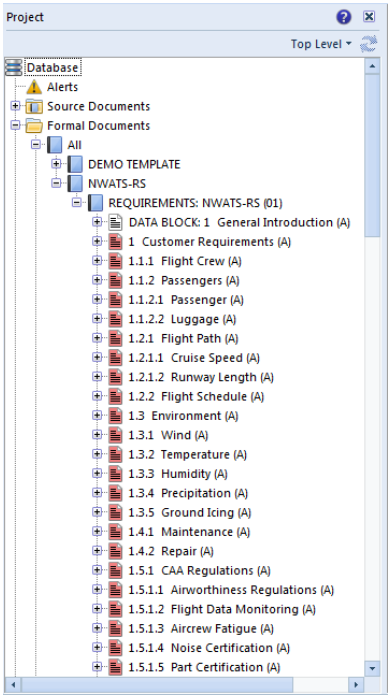
Each template can be used many times to publish items into many instances of a document, but when a significant project deliverable is to be produced, this same template is published in a way that marks the resulting document as a ***formal document***. That is:



- Publish formal documents from the same templates as other documents
- Users decide if a document is to be published as a draft, or as a formal document
- The content of a formal document is intrinsically no different to any other document produced from the same template
- You specify a ***title***, ***issue***, ***issue date*** and ***reference*** to publish a formal document, these values can be printed in the document, typically on its cover page.
- Cradle does further processing when publishing a formal document

Formal documents are often published from baselined since this information has been reviewed and cannot change. However, Cradle does not force this to be so.

Cradle keeps a record of the database items that have been published in each version of each formal document.



Users can browse which items have been published in each formal document, as reports and in the Cradle UIs.

As the information in the database evolves, new versions of existing formal documents can be published as new issues of these documents.

Formal document management therefore means:

1. Any number of formal documents can be published
2. Any number of versions of each formal document
3. Records are kept of which items are associated with each version of each formal document:
   - That can be viewed from items to documents
   - That can be viewed from documents to items
4. Formal document versions can be compared
5. The sets of database items associated with formal document versions can be compared
6. Formal document versions can be deleted, which:
   - Deletes the formal document and the cached document
   - Deletes the association between that version of the formal document and the items published in it

Summary 7

When tracking input source documents, Cradle:

- Keeps a record of the external documents that have been loaded
- Keeps a record of which items were loaded from each version of each source document

When publishing to formal documents, Cradle:

- Keeps a record of the formal documents that have been published
- Keeps a record of which items were published in each version of each formal document

Collectively, these provide full traceability from every issue of all input documents, into the database, to every issue of all output document and full tracking of the contents of all input and output documents.