INTERNATIONAL STANDARD

ISO/IEC/IEEE 26531

Second edition
2023-03

Systems and software engineering — Content management for product life cycle, user and service management information for users

Ingénierie des systèmes et du logiciel — Gestion de contenu relatif aux informations concernant le cycle de vie du produit, l'utilisateur et la gestion de service, à destination des utilisateurs

Reference number
ISO/IEC/IEEE 26531:2023(E)

Contents

Page

Foreword vi
Introduction viii
1 Scope 1
2 Normative references 1
3 Terms, definitions and abbreviated terms 1
3.1 Terms and definitions 1
3.2 Abbreviated terms 5
4 Conformance 6
5 Component content management system development and implementation 6
6 Content management project initiation 7
6.1 Developing a business case 7
6.2 Defining requirements for a CCMS 9
6.2.1 Requirements definition 9
6.2.2 Output requirements 9
6.2.3 Storage and retrieval requirements 9
6.2.4 Assembly and linking requirements 10
6.2.5 Writing, reviewing and workflow requirements 11
7 Project planning for CCMS implementation 12
7.1 Implementation plan 12
7.2 Information model 12
7.3 Information model specification 13
7.4 Writing guidelines 14
7.5 Automation-assisted reviews 15
7.6 Reuse strategy 15
7.6.1 Content conditional processing 15
7.6.2 Content inclusion 15
7.6.3 Content variables 16
7.7 Metadata schema 16
7.7.1 General 16
7.7.2 Administrative metadata 17
7.7.3 Descriptive metadata 18
7.7.4 Processing metadata 18
7.8 Workflow 18
7.8.1 Workflow specification 18
7.8.2 Workflow approvals 18
7.8.3 Translation workflow 19
7.8.4 Workflow completion 20
7.9 Schedule of activities, deliverables, and responsibilities 20
7.10 Training plan 20
7.11 Style sheet development 20
7.12 Pilot project specification 20
7.13 Organizational rollout 21
8 Content development 21
8.1 Content creation 21
8.1.1 General 21
8.1.2 Structured writing 21
8.1.3 Unstructured writing 21
8.1.4 Content granularity 22
8.2 Content conversion 22
8.2.1 General 22
8.2.2 Microcontent 23

9 Management and control **23**
9.1 Managing quality 23
9.2 Review of content 24
9.3 Approval of content 25
9.4 Search and retrieval 25
9.5 Localization and translation 25
9.5.1 Translation management 25
9.5.2 Content management for translation 26
9.5.3 Publication of translated content 26
9.5.4 Publication of multilingual content 26
9.5.5 Translation of vector graphics 26
9.6 Content deletion 27
9.7 Content and component archiving 27

10 Publication **28**
10.1 Release management 28
10.2 Version management 28
10.3 Publication of content 29

11 Component content management system requirements **30**
11.1 General 30
11.2 Component content management system framework 30
11.2.1 General storage requirements 30
11.2.2 Content types 31
11.2.3 Metadata structures 31
11.2.4 Administrative metadata 31
11.2.5 Descriptive metadata 31
11.2.6 Classification 32
11.2.7 Additional metadata requirements 32
11.2.8 Organizational structures 32
11.3 Component content management system management 32
11.3.1 Component creation and modification 32
11.3.2 Import/export 33
11.3.3 Bulk export 33
11.3.4 Dependency export 33
11.3.5 Archiving 34
11.4 Content object check-out and check in 34
11.4.1 General 34
11.4.2 Bulk check-out/check-in 34
11.4.3 Check-out and check-in after restart 34
11.5 Link management 34
11.6 Search 35
11.6.1 General 35
11.6.2 Full text search 35
11.6.3 Metadata search 37
11.6.4 Structured search 37
11.6.5 Advanced search capabilities 37
11.7 Versioning 38
11.7.1 General 38
11.7.2 Branch and merge 38
11.7.3 Release management 39
11.8 Graphics and multimedia management 39
11.9 Component content management system administration 40
11.9.1 User administration 40
11.9.2 Security 40
11.10 Content creation and acquisition 41
11.10.1 Writing interface 41
11.10.2 Content and metadata functions 41
11.10.3 Scientific notation and vector graphics 42

11.10.4 Writing integration ... 42
11.10.5 Acquisition ... 42
11.11 Workflow ... 43
11.11.1 Workflow creation ... 43
11.11.2 Workflow specification ... 43
11.11.3 Workflow reporting ... 44
11.12 Content publication ... 45
11.12.1 General ... 45
11.12.2 Export to publishing support ... 45
11.12.3 Centralized publishing support ... 45
11.12.4 Publishing interface ... 45
11.12.5 XML publishing pipeline ... 46
11.12.6 Dynamic publishing ... 46
11.13 Localization and translation management ... 46
11.13.1 General ... 46
11.13.2 XLIFF support ... 47
11.14 Component content management system interoperability ... 47
11.14.1 General ... 47
11.14.2 Libraries and frameworks ... 47
11.14.3 Web services ... 47
11.14.4 Application Programming Interface methods ... 48
11.14.5 Webhooks and triggers ... 49

Annex A (informative) **Business case considerations for content management** ... **50**

Bibliography ... **52**

IEEE notices and abstract ... **54**

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO/IEC documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

ISO/IEC/IEEE 26531 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*, in cooperation with the Systems and Software Engineering Standards Committee of the IEEE Computer Society, under the Partner Standards Development Organization cooperation agreement between ISO and IEEE.

This second edition cancels and replaces the first edition (ISO/IEC/IEEE 26531:2015), which has been technically revised.

The main changes are as follows:

— addition of information on the development of microcontent;

— addition of mathematics and vector graphics;

— addition of classification of objects using metadata and taxonomies;

— addition of webhooks and triggers;

— addition of XML reviews using Schematron or other similar systems;

— addition of reporting capabilities;

— addition of dynamic content generation.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

This document was developed to assist users of ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207 in the acquisition of a content management system and the use of that content management system to manage content used in product life cycle, user, and service management information. The accurate description of the requirements for content management helps organizations create information that meets the needs of its users and is efficiently produced.

This document is independent of the software tools that may be used to manage information for users and applies to both printed information for use and on-screen information for use.

Content management allows an organization to control the storage and retrieval of content objects, track content revisions, maintain a content audit trail, produce different types of reports, and enable a collaborative environment. Component content management supports the reuse of content objects among deliverables and supports multiple deliverable formats.

The use of content management functions can facilitate increased collaboration on content development across the enterprise. Technical writers, instructional designers, support staff, and others can develop a body of content together that is written once and supports many needs.

Information for users is often regarded as something done after the system or software has been implemented. However, for high-quality information for users, its development should be regarded as an integral part of the system or software development life cycle. In fact, quality information for users or information management services are important enough to justify specific planning.

This document is consistent with ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207, as an elaboration of the information management process.

This document is not a management system standard.

This document is intended for use in all types of organizations, whether they have a dedicated information-development organization or not. It may be used as a basis for local standards and procedures. Users are assumed to have experience or knowledge of general processes for information management, project management, and information development.

This document is intended for those engaged in the management of information, such as that included in:

a) information for users such as topic collections, manuals, guides, user assistance displayed with software, style guides, knowledge-based articles, and other content that supports the effective use of a system or software product;

b) product life cycle information such as design documents, use cases, personas, project management plans, feature requests, and testing plans;

c) service management items such as service level agreements, records, policies, procedures, documents in response to tender offers, and other documents.

The order of clauses in this document does not imply that the content management activities should be performed in this order, nor that information for users should be developed in this order or presented to the organization in this order.

In each clause, the requirements are independent of media and document creation and management specifications.

In this document, the following verbal forms are used:

— “shall” indicates a requirement;

— “should” indicates a recommendation;

— “may” indicates a permission;

— “can” indicates a possibility or a capability.

Systems and software engineering — Content management for product life cycle, user and service management information for users

1 Scope

This document specifies requirements for efficient development and management of information produced

— throughout the life cycle of a system and software product;

— for the provision of information for users of systems and software;

— for the management of IT and support services.

This document is independent of the tools, protocols, and systems used for content management. It does not address configuration management of software assets.

The content management process presented in Clauses 6 to 10 is a specialization (lower-level process) of the information management process specified in ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207.

2 Normative references

There are no normative references in this document.

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO, IEC, and IEEE maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

— IEEE Standards Dictionary Online: available at https://dictionary.ieee.org/

NOTE 1 For additional terms and definitions in the field of systems and software engineering, see ISO/IEC/IEEE 24765, which is published periodically as a "snapshot" of the SEVOCAB (Systems and software Engineering Vocabulary) database and is publicly accessible at www.computer.org/sevocab.

NOTE 2 Throughout this document, the term "information for users" refers to information for users of hardware and software.

3.1.1
application programming interface
API
set of functions, protocols, parameters, and *objects* (3.1.23) of different formats, used to create software that interfaces with the features or data of an external system or service

3.1.2
branching
method of development in which a set of *components* (3.1.3) is duplicated so the components can be modified in parallel and optionally synchronized at a later time

3.1.3
component
object (3.1.23) with a discrete *information type* (3.1.20) that is stored in a *CCMS* (3.1.4), such as a *topic* (3.1.31), prerequisite, section, image, or video

3.1.4
component content management system
CCMS
content management system (3.1.5) that supports the entire information-development life cycle from writing through review and publishing, including the reuse of modular content

Note 1 to entry: In case the modular content is XML-based, the individual XML elements available for management are defined by the XML schema or *DTD* (3.1.12). This document is protocol-independent, and it is not necessary to specify numerous markup languages.

3.1.5
content management system
system that makes *components* (3.1.3) available for reuse and linking to build *content objects* (3.1.6) and deconstructs large content objects into components that can be individually managed

Note 1 to entry: See also *document management system* (3.1.11).

3.1.6
content object
self-contained unit of content

3.1.7
content type
specific indicator of content

3.1.8
content unit
identifiable and manageable part of larger information *objects* (3.1.23)

Note 1 to entry: The individual content units available for management are typically defined by an XML schema or *DTD* (3.1.12).

3.1.9
customization
modification of a *document type definition* (3.1.12) to add new structures or change the document type definition in a way that is not compatible with a previous structure

3.1.10
dependency export
operation in which a *component* (3.1.3) and all its dependencies are exported from the *CCMS* (3.1.4) as a single process

3.1.11
document management system
system that supports the storage, retrieval, the production of a *version* (3.1.32), and the manipulation of whole documents, images, and other media

Note 1 to entry: See also *content management system* (3.1.5).

3.1.12
document type definition
DTD
template for the structure, content, and semantics of documents

3.1.13
effectiveness
accuracy and completeness with which users achieve specified goals

[SOURCE: ISO/IEC 25062:2006, 4.2]

3.1.14
eXtensible Markup Language
XML
platform-independent markup language that carries rules for generating text formats that contain structured data

[SOURCE: ISO/IEC 19770-5:2015, 3.15, modified — "license-free and" has been removed from the beginning of the definition.]

3.1.15
faceted search
progressive search that allows users to narrow the results by selecting values for one or more attributes

3.1.16
framework
<CCMS>essential data structures, operations, and rules that form the foundation from which all other features of the *CCMS* (3.1.4) are built

3.1.17
Hypertext Markup Language
HTML
language for creating web pages

3.1.18
Hypertext Transfer Protocol
HTTP
application-level protocol for distributed, collaborative, hypermedia information systems

3.1.19
information item
separately identifiable body of information that is produced, stored, and delivered for human use

Note 1 to entry: An information item can be produced in several *versions* (3.1.32) during a project life cycle.

[SOURCE: ISO/IEC/IEEE 15289:2019, 3.1.12, modified — The preferred term "information product" has been removed; the original note 1 to entry has been removed; note 2 to entry becomes note 1 to entry.]

3.1.20
information type
class of *topics* (3.1.31) that addresses a particular user question

EXAMPLE An information type that answers the question "how do I ..." is called a task information type.

3.1.21
Levenshtein distance
measure of the difference between two-character sequences based on the minimum number of single character edits (insertion, deletion, or substitution) needed to convert one word to the other

3.1.22
link
part of a computer program, often a single instruction or address, that passes control and parameters between separate *modules* (3.1.24) of the program

3.1.23
object
encapsulation of *content units* (3.1.8) in a *CCMS* (3.1.4)

3.1.24
module
appropriate independent information unit

3.1.25
publishing pipeline
series of defined processing steps that are connected to transform content from its source format into a final deliverable format

3.1.26
regular expression
Regex
string of characters that allows patterns to be used to match search results

Note 1 to entry: Patterns can dictate that matches start or end with specific sequences of characters or allow the use of wildcards to match any characters in a sequence.

EXAMPLE 1

^admin* - Find all matches that start with 'admin' and contain any sequence of characters afterwards

\d{5}$ - Find all matches that end with the number 5

^[0-9()-]+$ - Find matches that contain a 10-digit phone number

EXAMPLE 2 A semantic label such as prerequisite describes the content as a pre-requisite to the following task information. In contrast, a format label simply describes the content as a paragraph or a list.

3.1.27
Schematron
language for making assertions about the presence or absence of patterns in *XML* (3.1.14) documents

3.1.28
specialization
specification of targeted *document type definitions* (3.1.12) that share the common output transformations and design rules developed for more general types and domains

3.1.29
structured writing
development of content elements according to a pre-defined and enforced organization of content elements including metadata in specified templates

Note 1 to entry: In structured writing, content elements are labelled according to the nature of the content they contain. Structured writing also permits quasi-semantic labelling, such as Heading1 or NestedList, to indicate the hierarchical position and function of a content element.

3.1.30
taxonomy
scheme that partitions a body of knowledge and defines the relationships among the parts

3.1.31
topic
unique label or identifier, with which one or more items of information may be associated

3.1.32
version
form of a text or illustration differing in certain respects from an earlier form

3.1.33
XML schema definition
XML (3.1.14) based language that specifies a set of rules and structure for the creation of XML documents

3.2 Abbreviated terms

BMP	Bitmap Image File
CSS	Cascading Style Sheets
DITA	Darwin Information Typing Architecture
FAQ	frequently asked questions
FOAF	Friend of a Friend
GIF	Graphics Interchange Format
JPEG	Joint Photographic Experts Group
JWT	JSON Web Token
QTFF	QuickTime File Format (abbreviated as mov)
MP3/MP4	See MPEG
MPEG	Moving Picture Experts Group
PDF	Portable Document Format
PGP	Pretty Good Privacy
PNG	Portable Network Graphics
RNG	REgular LAnguage for XML Next Generation
RDF	Resource Description Framework
SKOS	Simple Knowledge Organization System
SVG	Scalable Vector Graphics
WAV	waveform audio file format
WMV	Windows Media Video
XLIFF	XML Localisation Interchange File Format
XSL-FO	XML Stylesheet Language-Formatting Objects
XSLT	XML Stylesheet Language Transformations
XHTML	eXtensible HyperText Markup Language

4 Conformance

The purpose of this document is to define a process for the development and acquisition of a component content management system and the requirements of a component content management system through which content is created, gathered, managed, and published, including the requirements of a system that is supported by an electronic database. Such a database should support documents or topics and content units that can be assembled to produce complete documents for print, electronic output, or content published through electronic media. This database is defined as a component content management system (CCMS), which differs from a document management system. The objective of component content management is to create content objects once and use them through linking mechanisms in multiple output formats, including but not limited to documents.

The intended users of this document are managers and developers of technical information and acquirers and suppliers of content management systems. Any organization that develops content, regardless of size, can benefit from maintaining an effective content management solution and following best practices for the development and management of technical information.

Systems conforming to this document can fulfil business needs for content development and management, especially the need for a single source of information. Content objects that are unique and are maintained as independent database objects are efficient to review, approve, and update; can be combined to produce multiple deliverables; and are cost-effective to translate.

This document may be used as a conformance or a guidance document for projects and organizations claiming conformance to ISO/IEC/IEEE 15288 or ISO/IEC/IEEE 12207.

Use of the nomenclature of this document for the parts of information for users, such as topics, content units, modules, is not required to claim conformance.

This document may be included or referenced in contracts or similar agreements when the parties (called the acquirer and the supplier) agree that the supplier will deliver services and systems in accordance with this document. This document may also be adopted as an in-house standard by a project or organization that decides to acquire information for users from another part of the organization in accordance with this document.

5 Component content management system development and implementation

Component content management allows an organization to control the storage and retrieval of content objects, track content revisions, maintain a content audit trail, and enable a collaborative environment. Component content management supports the reuse of content objects among deliverables and supports multiple deliverable formats.

The implementation of a component content management may result in increased collaboration on content development across the enterprise. Technical writers, instructional designers, support staff, and others may develop a body of content together that is created once and supports multiple needs.

Component content management system development and implementation processes described in this document are illustrated in Figure 1. The details of each activity in the content management development and implementation process are discussed in separate clauses.

NOTE ISO 21500 and ISO/IEC/IEEE 26511 can be consulted for managing a content management project.

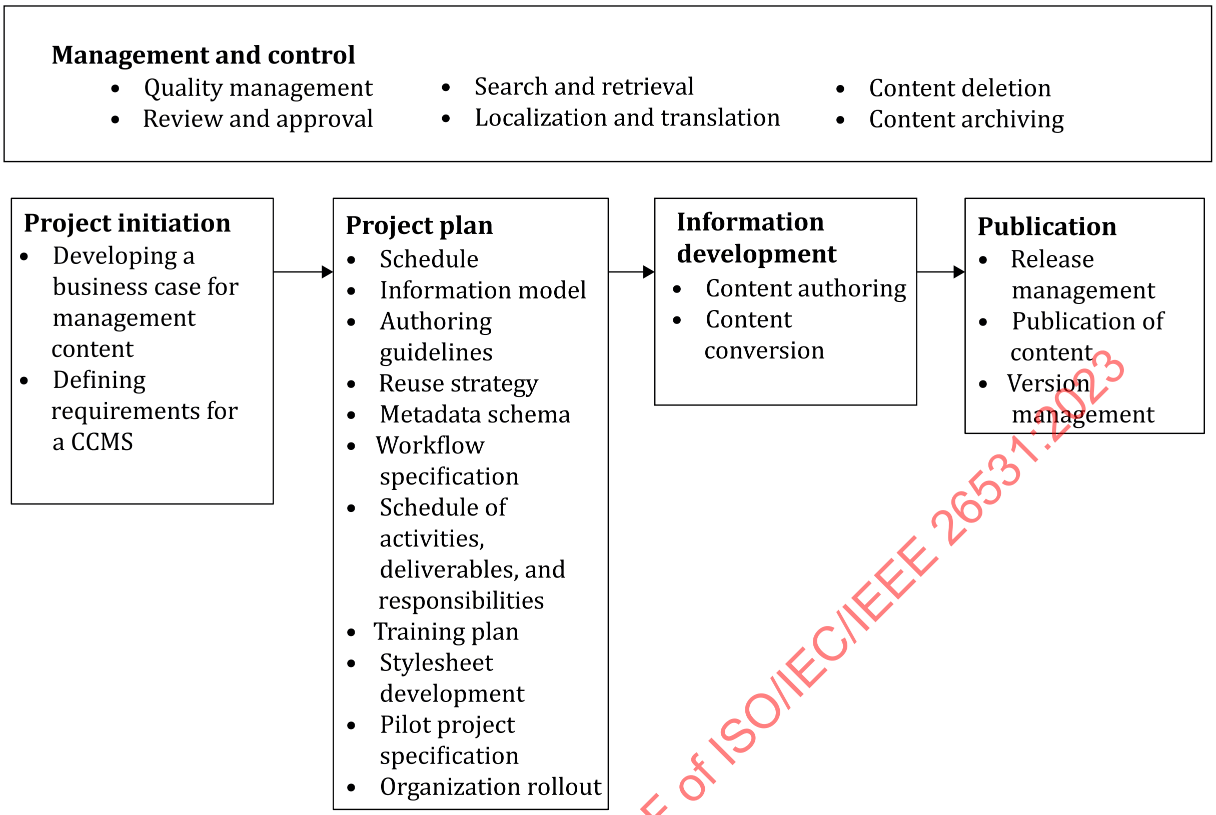

Figure 1 — Component content management development and management processes and activities

6 Content management project initiation

6.1 Developing a business case

The organization shall develop a business case to support the acquisition, development, and implementation of a component content management solution.

In developing a business case for component content management, an organization shall evaluate the needs of the organization, as follows:

a) Analyse the current state.

b) Identify potential customer benefits.

c) Identify potential author benefits.

d) Identify potential organizational benefits.

e) Identify cost reduction opportunities.

f) Calculate acquisition and implementation costs, including both technology and personnel costs, including the cost of training for the working group.

g) Calculate the return on investment.

h) State the risks of the project.

The development of a business case can be necessary to obtain support and funding for a component content management solution. Establishing the business benefits of component content management in an organization includes carefully evaluating both immediate and long-terms gains and evaluating enterprise and departmental solutions. Understanding the present situation assists in developing a vision of why and how that situation should be changed.

Component content management allows an organization to control the storage and retrieval of content objects, track content revisions, maintain a content audit trail, and enable a collaborative environment. Component content management supports the reuse of content objects among deliverables and supports multiple deliverable formats.

A consequence of component content management is increased collaboration on content development across the enterprise. Technical authors, instructional designers, support staff, and others should develop a body of content together that is created once and supports multiple needs.

By implementing a CCMS and activities to support that system, organizations can experience the following benefits:

— Standardized methods for creating content

— Standardized levels of content quality

— Avoidance of duplicated or near-duplicated content

— Reduced maintenance of content objects

— Improved access and successful search to content objects with appropriate permissions

— Reduced storage and maintenance requirements through consolidation of content in a unified system

— Traceability of review and approval cycles and change management

— Reduced development costs in source and target languages

— Improved relationship linking between source and target languages

— Ability to mitigate legal and compliance risk through a single source of authoritative content

— Ability to deliver content in multiple formats according to current and future standards

For more detailed information about developing a business case, see Annex A.

Component content management, if well controlled and managed, increases staff productivity and reduces development and publishing costs. Productivity increases are achieved by implementing processes that support structured writing and content reuse. Structured writing implies that content is written according to specified templates, depending upon the type of content to be developed. Content reuse implies that content objects are stored and maintained uniquely, decreasing the cost of updating and translating the same content multiple times.

Content reuse also implies that a content object is a single source of authoritative content, ensuring that the same information and the exact message appears in every instance of output, decreasing the liability associated with incorrect information. Development costs are reduced because existing content is available to authors through search and retrieval, implying that specific information is created and updated only once. Publishing costs are reduced because content objects may be published in more than one format using automated publishing routines.

If content is developed using an XML schema definition (XSD) or comparable standard grammar, the cost of formatting that content in multiple languages is eliminated once appropriate style sheets are in place. For those organizations that are required by customer requirements to publish content in multiple languages, CCMSs and processes enable significantly decreased time and cost. CCMSs that enable rigorous linking between content objects also support links between source and target language content objects. Once a body of content has been translated, CCMSs establish links between source and target language content objects so that only new or revised content need be translated. Previously translated content doesn't need to be sent again for translation, reducing not only translation costs but also administrative costs in handling already translated content.

6.2 Defining requirements for a CCMS

6.2.1 Requirements definition

The business case identifies concerns to be addressed in the explicit definition of requirements for a CCMS. The development of a detailed requirements definition enables an organization not only to specify its requirements internally but also to carefully evaluate competing products. A well-structured requirements definition focuses on five key areas:

a) Output (6.2.2)

b) Storage and retrieval (6.2.3)

c) Assembly and linking (6.2.4)

d) Writing, reviewing, and workflow (6.2.5)

e) Information technology and architecture requirements

6.2.2 Output requirements

Output requirements shall specify the content delivery mechanisms that make information available to the user community, both internally and externally.

Translation requirements, including the languages and character sets (fonts) required, shall be specified.

Requirements for automatically creating translation packages for transmittal to localization service providers should be specified, typically including the use of the XLIFF standard for the transmission of unified data files for translation.

NOTE The OASIS XLIFF provides such a specification at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xliff. Also see ISO 21720.

By profiling output, an organization can target different versions of information to different groups, based on internal needs, market segments, specific customers, and others.

Information can potentially encompass multiple versions of a collection of products, where much of the information is shared and some of the information is varied depending on product type, mode, product release, or version.

Information can be output in multiple media, including print, PDF, embedded user assistance, electronic devices, electronic applications (apps), e-learning, web pages, information portals, and others. Information can vary depending on the location or language of the users.

Output requirements should include the potential use of topic- and publication-specific metadata to enable faceted search mechanisms. Metadata should also include output specification such as the content of text referring to legal requirements, headers and footers, title pages, and covers.

6.2.3 Storage and retrieval requirements

In preparing storage and retrieval requirements, organizations shall identify how each division or component of the organization using the CCMS will access its content. Storage requirements shall include the ability to maintain links between content objects.

Requirements shall be stated for the storage and retrieval of translated content, especially the requirement to maintain synchronicity between source and target languages to reduce translation costs.

Retrieval requirements shall specify how the CCMS enables internal users to search for and find information.

Various CCMS support either folder-like structures supported by an underlying database management system for navigation to content or search mechanisms that enable users to identify specific content using metadata-enabled search.

Organizations may need to store multiple content types, including images, video, and voice, and entire document constructs as well as storage of individual components that will be assembled into larger constructs on output. Additional content types, including content created in word processing, slide production, spreadsheet, or other content-development systems, may need to be stored.

Search mechanisms may be focused on full-text search, metadata-based faceted search, XML-element enabled search, or a combination of these mechanisms.

6.2.4 Assembly and linking requirements

Organizations shall specify how they expect to collect components of content for assembling into final deliverables, including producing full, preview, and partial draft content for reviews.

In an XML environment, linking requirements are especially important because final deliverables are produced through elaborate linking mechanisms. Topics may be assembled into chapters and sections of large documents. Reusable content, such as warranty statements, hazard statements, and others may be stored once and included in multiple instances in final document assemblies. Individual words and phrases may be stored separately so that they may be updated as needed before final assemblies. Each of these situations requires a robust linking mechanism.

In addition, when a complex hierarchy of linked topics, as in a table of contents (TOC), is required, link filtering should be used to include or exclude parts of the table of contents for profiled publications.

For example, different final tables of contents can be constructed starting from two distinct tables of contents (see Figure 2).

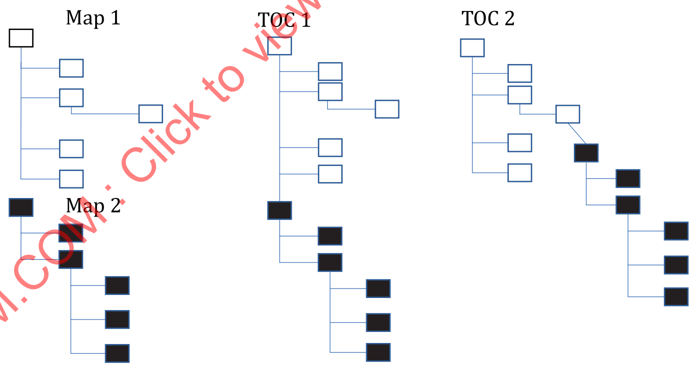

Figure 2 — Using two distinct maps to produce two different tables of contents for publications

In DITA XML, two different TOCs can be built starting from two different DITA maps (.ditamap files). In Figure 2, a.ditamap 1 and a.ditamap 2 can be composed in two different ways, for two different deliverables. This mechanism can be also provided by a CCMS.

Some parts of a table of contents can be filtered for publication (see Figure 3).

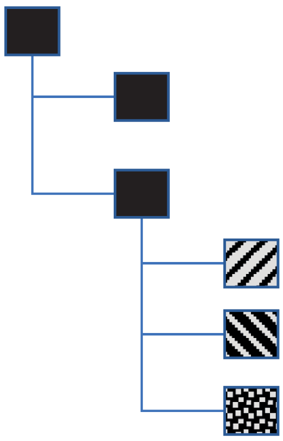

Figure 3 — Table of contents filtered for publication

In Figure 3, a table of contents is developed for a machine with three different customizations.

A deliverable can be constructed that contains subparts of the full publication. Other customizations can be obtained by using filters applied to the links.

6.2.5 Writing, reviewing and workflow requirements

Organizations shall specify how an audit trail of changes and comments shall be maintained throughout the information life cycle.

Organizations shall specify the elements of their information life cycle management process, including

a) importing legacy content,

b) enabling writing by both full-time professional writers and casual contributors,

c) supporting simultaneous review by groups of reviewers and electronic signature of approvers,

d) assembling content into larger deliverables,

e) releasing approved content for publication, and

f) archiving deliverables at end-of-life.

In preparing writing and workflow requirements, organizations should identify how writers, reviewers, and those who are designated to approve content before its release expect to work with the CCMS. The organizations may capture expectations as use cases or user stories.

Organizations can develop requirements to manage workflow, including assignments and content completion requirements, as well as moderating content completion actions.

Organizations can specify the elements of their information life cycle management process, including importing legacy content, enabling writing by both full-time professional writers and casual contributors, supporting simultaneous review by groups of reviewers and electronic signature of approvers, assembling content into larger deliverables, releasing approved content for publication, and archiving deliverables at end-of-life.

7 Project planning for CCMS implementation

7.1 Implementation plan

The elements of the content management project shall be planned by the organization. The organization shall establish an implementation plan including the following activities:

a) Schedule activities, deliverables, and responsibilities

b) Define the information model

c) Develop authoring guidelines

d) Define a reuse strategy

e) Establish a metadata schema

f) Specify the workflow

g) Schedule activities, deliverables, and responsibilities

h) Conduct training

i) Develop style sheets and templates

j) Conduct a pilot project

k) Define the organizational rollout

See Figure 1.

7.2 Information model

An organization shall create an information model to specify the document types, information types, and content units that will be managed. An information model defines the structure of the content to be created and managed in the content management environment.

The information model shall define the use of each XML element in the writing environment.

Creating and approving an information model allows an organization to specify the structure of the content to be stored in the CCMS. It may include references to specific published standards, such as OASIS DocBook, OASIS DITA, or S1000D, but may also be specifically designed for an organization's publications as a proprietary model.

The information model promotes consistency in the development of well-structured content objects that best meet the needs of the information users. By following a well-constructed information model, content developers throughout the organization are able to create uniform content that is capable of use in a wide variety of output types and among a wide variety of information products.

NOTE For further information, see IEC/IEEE 82079-1:2019, 5.3.6 and IEC/IEEE 82079-1:2019, 8.3.

For organizations using a public standard, the information model describes the specific implementation of the public standard within the organization. In most cases, an organization may use a subset of the public standard or may add specializations or customizations of that standard to meet specific internal requirements.

7.3 Information model specification

An information model that supports structured writing shall specify the following details about the organization's required content structures:

a) definitions of each generic document type such as description, plan, policy, procedure, report, request, and specification and descriptions of the typical content and sequence of content;

NOTE It can be useful to provide outlines and templates to assist in the development of each document type. ISO/IEC/IEEE 15289 includes recommended content for information items.

b) definitions of the structures for collections of content such as chapters, parts, sections, books, embedded user assistance, lessons, learning modules;

c) definitions of each information type, such as overview, task, reference, concept, and troubleshooting, explaining the purpose of each information type with reference to the appropriate audience;

d) structure of each information type with required and optional content units, required and optional sequence of content units in the information type, and number of permitted content units;

e) definitions of each content unit, such as sections, lists, paragraphs, tables, illustrations, examples, hazard statements, specifying how each is to be used to build each information type;

f) definitions of elements, such as keyword, index term, citation, user input, and variable used within larger content units to specify words or phrases;

g) definitions of terms to be used as metadata to describe each content unit, such as writer, date created, subject matter, and version;

h) direct or indirect mechanisms to be employed when reusing variables or microcontent, such as words, phrases, or full content units across information or document types;

i) file-naming conventions for both text and illustrations to support consistency and retrieval.

These structures are illustrated in Tables 1 and 2.

Table 1 — Content data types in a CCMS information model

Data type	Purpose	Examples
Generic content (archetype)	Provide outlines or lists of typical/required content	Description, plan, policy, procedure, report, request, specification
Collection of content	Organize content for use in an output document or document set	Chapter, part, section, book, help topic
Information type	Convey information for a specific purpose	Overview, task, reference, concept
Content unit	Build an information type with smaller components	Paragraph of text, table, list, illustration, example, hazard statement
Element	Specify words or phrases	Keyword, index term, citation, user input, variable
Direct or indirect mechanisms	Content reuse	Microcontent, such as words, numbers, and phrases

Table 2 — Templates for CCMS metadata

Template	Specifies	Examples
Publication	Information associated with the entire publication	Copyright or publication date, distribution limitations/restrictions, doc control numbers, product, version, language

Table 2 *(continued)*

Template	Specifies	Examples
Information type	Required and optional sequence of required and optional content units	Task, concept, reference, troubleshooting, process, procedure, fact
Content unit	Metadata to retrieve the content unit	Writer, date created/revised, subject matter, version, format (e.g. size and borders)
Variable insertion	Method for storing, retrieving, and using variables	Changing words, phrases, or content units in a publication, information item or content unit
File name	Convention for storage and retrieval of files containing text, illustrations, and multimedia	

The information model helps to apply the XML markup consistently to the organization's content so that content output can be generated using automated style sheets and that content objects can be searched and manipulated by writers using a component content management system.

XML markup is supported using an XML schema, an XML document type definition (DTD), or an RNG. A schema or DTD validates the structure of the document or topic. The DTD or schema can be used to identify invalid content. In some cases, the invalid content may still be allowed to be entered into the CCMS but should be marked as invalid. An RNG is a schema language for specifying the structure and content of an XML document.

NOTE For further information, see IEC/IEEE 82079-1:2019, 8.2.

7.4 Writing guidelines

Organizations shall develop writing guidelines to provide instructional material to support the development of content that conforms to the information model. Guidance shall be provided on the correct selection of information types, the correct implementation of inline elements, the correct application of XML elements within each information type, and other requirements that help to specify the organization's XML standards implementation.

Writing guidelines, as well as the information model, assist writers in planning and developing each information type, and content unit required in the information-development life cycle. Writing guidelines instantiate the information model in a form that provides assistance and training for writers. The goal of writing guidelines is to help ensure that writers understand and conform to the structures defined in the information model. Writing guidelines may also provide guidance for writers who are developing unstructured content. They may include guidance on the proper use of terminology, writing style, grammar, and spelling-typical content of style guides.

Many organizations adopt writing guidelines based on prominent published guidelines that emphasize producing quality technical information. In addition, such organizations supplement published guidelines with specific information dealing with the content the organization produces. However, traditional style guides are not sufficient to provide writing guidelines for developing structured content.

Writing guidelines and the information model shall be used to train new writers. Writing guidelines should also include examples of developed content objects, including documents, topics, content units, and inline elements.

Writing guidelines should include process guidelines for content management processes, such as creating variants and revisions, managing translation processes, and incorporating reuse mechanisms.

7.5 Automation-assisted reviews

If organizations have implemented standards for XML-based writing, they shall conduct automation-assisted reviews of XML elements to support conformance with the information model. XML validation engines test conformance with the XML schema or document type definition (DTD), but do not test the coding practices specified by the organization in its information model.

Automation-assisted reviews should be conducted by individuals with expertise in the structured information model and the writing guidelines instantiated in the organization. Code reviews can also be conducted using the Schematron standard, in which coding requirements are explicitly defined in the Schematron system. Using Schematron, an organization can develop an additional rule set that tests the presence or absence of elements or sets of elements in the XML source. The rule state can also check for non-structural writing guidelines such as paragraph length or the use of appropriate terminology. Schematron testing supplements, but does not replace, expert code reviews.

NOTE Schematron is part of ISO/IEC 19757-3.

The component content management system should provide the capability to enforce business rules or style guide rules that use technology such as ISO Schematron. Schematron schemas should be configurable so that they report warnings and errors to writers as they are creating and editing content. The system should be able to suggest edits to content that the writer can apply automatically.

Schematron can also test the presence or absence of attributes on elements and help ensure that those attribute values conform to a specific list of values. Schematron can provide logical relationship testing to help ensure that combinations of attributes also make sense. For example, there are @product values of X, Y, and Z and @platform values of OS-A and OS-B. But if product X doesn't have an OS-A version, Schematron will flag that state; even though the two attributes are valid by themselves, they are not valid in combination.

7.6 Reuse strategy

7.6.1 Content conditional processing

CCMSs may facilitate a variety of reuse mechanisms to assist writers in maintaining single sources of content and assist organizations in establishing reuse policies and practices. Organizations should maintain a balance between the usefulness and cost effectiveness of reuse mechanisms and risk of developing overly complex mechanisms that are difficult to maintain.

So that content is written only once, organizations shall require that writers maintain records of content objects created for specific purposes or query the CCMS for content that matches a proposed new content object before that new object is created. Content objects that are intended for reuse may be stored in special "reuse" folders if these are available in the CCMS and should be labelled with reuse metadata.

The objective of content management is to create content once and use it, as needed, in multiple deliverables. Content can be reused either as complete modules or topics or as microcontent such as hazard statements, lists, paragraphs, words, and phrases.

Content reuse enables writers to update content in only one source, thereby maintaining the consistency and accuracy of the source content. Content reuse enables the organization to translate content only once from the source into the target languages, ensuring that the same content is translated consistently and accurately in each target language.

7.6.2 Content inclusion

Organizations shall identify and designate content objects that are frequently used in multiple deliverables and describe these content objects with metadata so that they are easily located for reuse.

Content inclusion mechanisms, which are often specific to the writing platform or to the XML standard being used to write, permit content to be inserted in a target file from a separately maintained source file at time of publication. Consequently, the included content is maintained in one source rather than repeatedly maintained as multiple targets. Maintenance of content objects in a single source reduces maintenance and translation costs and improves consistency, accuracy, and quality.

7.6.3 Content variables

Organizations shall develop policies to direct how writers use variables to allow substitution of words, phrases, or other content elements into content objects.

Variable substitution mechanisms allow lists of terms or phrases to be maintained in separate source files and inserted into content objects at publication. Variable substitution permits the same content to be used in multiple deliverables that need small changes to content, such as product or system names or user interface terminology that is changed frequently during the information-development life cycle. Variable substitution also can enable the same source content to be used for multiple deliverables that require only certain variables be inserted to correspond to a specific set of products, systems, or versions.

Policies should account for the possibility of mistranslation of sentences containing variables or require that translators attend to moving variables within a sentence.

Organizations shall establish policies on the use of conditions in source content objects that can be filtered at publication. Conditional processing uses metadata to identify alternative words, phrases, paragraphs, lists, illustrations, or other components that are included in source content objects. At publication, the metadata are used to filter the content so that only one version of the source content is published, corresponding to the selected filter.

The presence of multiple conditions in the source content objects enables writers to create one content object that is used for multiple outputs, functioning as a single source. The content containing multiple conditional content objects can be updated once, reducing maintenance and translation costs and improving consistency, accuracy, and quality.

7.7 Metadata schema

7.7.1 General

Metadata are sets of data that describe and give information about other data. Metadata can be added as attributes to content so that the content can be searched and retrieved quickly and effectively. Metadata describe the content of the objects in terms that are meaningful to the users.

An organization shall establish a model that defines the metadata it will use to describe objects in the CCMS as well as the metadata to describe content delivered to customers.

A metadata schema shall be developed to label content at all levels so that it can be effectively searched and retrieved and used to support automated processes for selecting or filtering content objects in final publications.

To create a metadata schema, an organization shall examine its content and identify the terms that best describe that content for one or more user communities. User communities may include:

— writers who need to search the CCMS for content related to their writing assignment;

— internal users who need to search the CCMS for content related to their work;

— consumers of content outside the organization who use metadata to retrieve content best associated with their information needs.

When associating metadata with content in the CCMS itself, rather than storing it in the objects themselves, organizations should consider whether the metadata should be translated. Metadata that describe a category of content or type of task may be translated to facilitate searches by readers of that language. However, if a CCMS does not enable translation of user-specific metadata stored in the CCMS without the creation of additional, language-specific metadata fields, a process should be designed to export those fields for translation.

Metadata don't need to be applied uniformly to content within a CCMS. Rather, different metadata elements or values may be associated with different object types within the CCMS, such as library, publication, map, topic, illustration, template, or content unit. At the publication level, metadata may include information such as copyright dates, publisher information, volume, and part numbers. Topic metadata may include product name and version, brand, category, and keywords. Metadata associated with higher-level object types, such as a publication, should apply to the content within that object type. It is important that an organization understand how metadata values are transferred among nested objects in the publishing system they have adopted. Organizations should know if metadata values specified at a higher level add to or overwrite values specified within nested objects.

A metadata schema may be flat or hierarchical. In a flat structure, metadata are simply associated directly with the object type. In a hierarchical structure, different metadata may be applied to different levels of the object. Certain metadata may apply to the base content, but other metadata may be associated only with a specific version, language, or workflow stage of the object.

Metadata schema can be hierarchical, not only based on the hierarchy of the object, but on the hierarchy of the taxonomy. For example, Product A can have components X, Y, and Z, while product B has components L, M, and N. Based on the metadata chosen for product, the choices of component can be limited to those that apply.

EXAMPLE Version 1 of a topic includes information about a specific feature; if that feature is removed for version 2, metadata about that feature is also removed. If metadata are retained in version 1 of the topic, the version control system stores metadata associated with a specific version of a topic.

A metadata schema should define the approved values for each attribute. A list of possible values for each attribute reduces the risk of writers specifying similar, but different, values for the attribute making it difficult to employ a metadata-specific search. Each item in a list should be distinct and clearly defined so that writers know which values apply. If metadata values cannot be predefined, the information model should include examples of the form the values should take to be valid. A quality management system or a taxonomy management system may be used to validate metadata values.

NOTE See ISO/IEC 11179-1.

Metadata are designated as required or optional. A minimal set of required metadata should include a basic set of descriptive metadata as determined by the information model.

Organizations shall specify the metadata that is collected and maintained by the CCMS.

7.7.2 Administrative metadata

Administrative metadata shall include the following:

a) name of each individual who has contributed content or interacted with content in the CCMS, including writers, editors, reviewers, and approvers;

b) dates and times associated with every action taken on the content in the CCMS, including the date and time of every change committed to the CCMS for a component;

c) workflow status, such as draft, review, approved, released, etc.;

d) a field for optional comments from individuals contributing or interacting with content to explain the reasons for the changes made to a component.

Organizations should be aware that some metadata are automatically collected and maintained by the CCMS. Administrative metadata are typically maintained through the life of the components in the CCMS. These metadata should be available if the content is moved to a new CCMS without loss of information.

7.7.3 Descriptive metadata

Writers and other people responsible for identifying the attributes and values associated with the content shall be required to apply metadata to content objects. Such application of metadata can be supported by automated metadata systems. Metadata attributes may include values to label the subject areas of the content. Examples of metadata values added by writers to the content include:

— product attribute with values consisting of product names and versions;

— audience attribute with values consisting of target audiences;

— component attribute with values consisting of hardware components;

— platform attribute with values consisting of the platforms supported by the products;

— other attribute and value combinations needed to describe the content in both documents and topics.

In developing a metadata model, organizations should aim to optimize subsequent searching in published content by including metadata that facilitate searches. Metadata associated with documents and topics may be exported with that content for delivery to users. The effectiveness of search systems that use metadata to focus end-user search and retrieval is dependent on the quality and consistency of the metadata and its regular addition to content objects.

Descriptive metadata are likely to change as products are updated, user communities realign, and standard terminology changes. The organization should review its metadata schema periodically and update.

NOTE ISO 25964-1 provides guidance for the development of thesauri-controlled vocabularies.

7.7.4 Processing metadata

Writers and other people responsible for identifying the attributes and values associated with publishing output may be required to apply metadata to objects, which may include attributes with values for:

— layout specifications or styles;

— conditions for variables in content;

— location and inclusion of external data.

7.8 Workflow

7.8.1 Workflow specification

The organization shall specify a workflow. The workflow specification defines each action that occurs during the life cycle from the initiation of a project through the release of content to the consumer, sustaining the content in a reuse environment, and archiving the content for permanent storage. Once specified, the workflow may be partially automated using the workflow functionality of the CCMS.

The goal of the information-development life cycle is to help ensure that the quality of the content developed is maintained and that work occurs in a timely manner, often expressed as conformance to required deadlines. A workflow specification includes the process of setting up and using a workflow, including the automation of that workflow in a CCMS.

7.8.2 Workflow approvals

The workflow system shall allow for repeating the writing/editing cycle. Multiple workflows may be specified, allowing for the use of the most appropriate workflow for the type of publication. The organization shall establish a policy that content is not released to the users until required approvals are completed.

Workflows may establish routing for the following activities:

— Project initiation

— Content development through various draft states

— Content editing and proofreading

— Content code review

— Content technical review

— Content validation and verification

— Content approval

— Content translation

— Content publication

— Content sustainment

— Content archiving after initial and subsequent releases

If the workflow specification is automated, the workflow system enables each action to be initiated, with assigned deadlines if applicable, and automatic notification of those responsible for each stage of the process that input is required.

The workflow system enables project managers and other people participating in the workflow to track the status of each action, monitoring its progress and completion, including sending reminders when the actions have not been completed in a timely manner.

Projects may be initiated in the CCMS by assigning the development of content, including topics or complete documents, to the appropriate writers. Once assigned, the workflow system notifies the writers of their assignments and provides access to the CCMS in which those assignments will be completed. The assignments may specify the deadlines for one or more drafts of the assigned content.

When writers have completed their assignments, they use the workflow system to automatically forward the content to those responsible for editing or proofreading. The workflow system notifies the assigned editors or proofreaders of their assignments and tracks their completion. After editorial tasks are complete, the content may be returned to the writers for resolution of changes.

The same process occurs for subsequent steps in the information-development life cycle, including reviews by technical experts, work by test professionals to verify and validate the accuracy of content, and approvals by those responsible.

If the organization requires that a pool of editors, proofreaders, technical reviewers, testers, or approvers be available, the workflow system can be designed to accommodate the specification of a pool so that one member of the pool may accept the assignment, thereby removing the action from the project work of the additional members of the pool.

7.8.3 Translation workflow

If translation of the content is required by the process, the project manager or other responsible party shall initiate the development of a translation package by the CCMS. A translation package can be created following the appropriate method. The CCMS creates a translation package that is made available to the organization responsible for translation and accepts the returned package when the translation is complete.

7.8.4 Workflow completion

The workflow system supports the final steps in the workflow specification, including publication of the content in one or multiple formats. A snapshot may also be taken of the content so that writers can return to that version for future reference.

7.9 Schedule of activities, deliverables, and responsibilities

The organization shall develop a schedule of activities to be conducted, the deliverables to be produced, and the responsibilities of team members in the implementation plan.

7.10 Training plan

The organization shall develop a plan for the training required for the participants in the pilot project and through an organizational rollout of the content management environment. Training shall include the information model and writing guidelines as well as the systems used for writing, publishing, and managing content objects.

The organization should schedule the required training immediately preceding the time when it is used in the implementation schedule. Training provided too early often needs to be repeated to ensure the project implementation plan's success.

Many methods of training can be used concurrently, including a buddy system, formal training, and informal FAQ sessions. Each method meets a different need and together allows working effectively.

7.11 Style sheet development

The organization shall develop style sheets to support the formatting and publication of approved content. In a structured writing environment, each deliverable produced has a style standard that specifies the page layout, font selection, and other aspects of well-defined final deliverables. In an XML structured writing environment, the final design and layout of deliverables are established using coded style sheets, often using the XSL-FO standard for PDF publications and CSS for PDF and HTML publications. Individual writers are unable to affect the final styles of the documents; those styles are determined by the organization’s standard style sheets.

7.12 Pilot project specification

Organizations shall specify one or more pilot projects to validate and verify the development activities before they are introduced to the larger organization. The selection of the pilot project should be based on the following criteria:

— exercises the information model and the workflow;

— is important to the organization, but not critical;

— ensures the usability of the CCMS;

— can be completed in a reasonable period of time;

— demonstrates the capabilities of the CCMS to improve productivity, decrease costs, and improve quality;

— demonstrates that the business case has been met;

— provides an opportunity to train staff members who will lead subsequent projects in an organization-wide rollout.

In addition, criteria shall be established to evaluate the pilot project. Criteria may include the ability to implement the information model successfully and the ability to apply the selected tools successfully.

7.13 Organizational rollout

After the pilot project is complete, the organization may extend the implementation plan to include the next set of projects for a rollout through the organization. Staff members trained in the pilot project may be selected as project leads for the additional projects to be included in the next stages of project implementation, while staff continue to be trained and new tools introduced.

8 Content development

8.1 Content creation

8.1.1 General

Mechanisms for writing content in a component content management environment may include both mechanisms to support word processing and desktop publishing, as well as XML-based structured writing.

Policies for content development in a content management environment may permit structured or unstructured content to be included. Structured writing prescribes the sequence and content required for specific document and topic information types. Structured writing can be enforced by validated XML schema or document type definitions (DTDs). Unstructured writing does not prescribe structure, but unstructured content may be included in the CCMS. Many component content management environments include both structured and unstructured content.

8.1.2 Structured writing

If structured writing is used, organizations shall require that components conform to structures defined in the organization's information model. The organization shall define the purpose of each information type and content unit and its required and optional subject areas.

An organization implementing structured writing shall require that writers label content elements using semantic labelling and following the specification provided in an XML schema or a DTD and manifested in the organization's information model.

In structured content, components are labelled so that they can be programmatically manipulated to create a variety of outputs. The output is not restricted to the sequence in which the content was created by the writer. Semantically labelled content elements may be addressed and restructured to correspond to the needs of the organization or the users.

Structured writing is supported by XML editing tools. Writers conform to the requirements of the XML schema or DTD through the validation routines of the XML editor. Templates designed to guide writers to produce well-formed and valid XML that also conforms to the requirements of the organization’s information model may also be used to support XML-based writing.

8.1.3 Unstructured writing

If unstructured writing is used, organizations shall define the content types of unstructured content that may be written and managed by the CCMS. The organization can also define informal structures for certain content types. However, unstructured writing decreases the advantages provided by the CCMS in a well-formed writing process.

By definition, unstructured writing is not controlled by an underlying well-defined formal architecture. Rather, the content is organized according to the decisions of individual writers. Writers of unstructured content can indeed follow guidelines, but the resulting topics are considered unstructured because the individual content elements cannot be individually addressed because they are not labelled.

Word processing and desktop publishing systems typically produce unstructured content that has low semantic value, low consistency, and low automation potential. Content structured in such systems is imitated through formatting controls, enabling writers to produce consistent informal structures if they follow the rules for formatting. Content defined by heading levels, lists, and paragraphs can be programmatically addressed and manipulated in limited fashion to produce a variety of outputs.

8.1.4 Content granularity

The organization should define the optimal length of topics, depending upon the needs of its audience and the requirements of its content reuse strategy. Optimal lengths may be associated with specific content types.

Writers often ask what constitutes an appropriate length for an individual topic. The general rule is that a topic answers a single question. A concept defines what something is. A task explains how to complete a procedure and reach a goal. A reference topic provides facts that the user needs to successfully complete a task or make a decision. A troubleshooting topic assists the user in identifying problems and finding appropriate solutions. Additional information types may be used to answer a specific user question or assist a user in reaching a goal.

A user can require a complete set of information to answer a single question, which can be found in a primary topic and its subsections. On the other hand, a user can require a topic to be brief, able to be comprehended in a few minutes. A more complex piece of information may be contained in multiple topics. The length of a topic requires a balance between these two constraints: quick reference and thoroughness.

The length of a topic can also be influenced by the experience and sophistication of the user. A more experienced and sophisticated user can be more interested in and tolerant of the details of a more complex topic than a less experienced beginning user.

Providing users with links among topics is a mechanism to increase the coverage of a topic while at the same time keeping individual topics at a minimal length. However, linking mechanisms, such as cross references, should be minimized to avoid sending the user in multiple directions to answer a single question or solve a single problem.

Topic size is also a factor of the reuse environment implemented by the organization. A brief topic is more amenable to reuse than a lengthy and complex topic. Small topics may be joined together to provide a longer discussion of a subject for a particular user community. Small topics may remain separate to accommodate those needing only a subset of the information.

Topic size may be governed by the need to assign unique metadata classes for precise retrieval of topics. However, topic size is governed by the concept that a topic answers one reader question, such as “what is ...?”, “how do I ...?”, etc.

8.2 Content conversion

8.2.1 General

Information development addresses the conversion of legacy content according to the rules established in the information model.

Organizations shall provide policies and procedures for converting unstructured content to structured components if the content management system is to be restricted to structured content. Pre-defined templates should be created and made accessible to writers when they convert unstructured content or when they develop new content.

Most organizations that inaugurate a content management project have unstructured content that does not conform to a standard XML schema or document type definition (DTD). In addition, much content is poorly organized, in that it is written inconsistently and does not conform to standards, even those that have been defined by the organization.

Organizations that prepare to implement a structured content development process shall perform a content inventory to identify the following:

a) unstructured content to be converted in conformance with a structured information model;

b) content to be maintained without conversion, such as content nearing end-of-life;

c) content to be rewritten to conform to a structured information model.

Well-organized existing content can be automatically converted to structured components that conform to an XML-based information model. So that the conversion process is successful, an organization may decide to better organize existing content in the unstructured writing environment before conversion or may decide to repair structural problems following conversion. Automated conversion works best when existing content is well organized and follows writing guidelines.

If necessary, existing content should be rewritten to conform to the information model before conversion. After content is converted to a structured writing environment, writers find it more difficult to restructure the new components than if they revised content in their familiar writing environment. However, unstructured writing environments provide no mechanisms to enforce and validate structure. Therefore, rewriting content that should be strictly structured in an unstructured writing environment can be less successful than desired.

Some existing content in its legacy unstructured form, especially content that is near its end-of-life, should not be converted to a structured form at all. Content that is unlikely to be updated or have only minor changes may best remain in its existing form.

8.2.2 Microcontent

Microcontent is typically a single content unit, e.g. a paragraph, a hazard statement. The granularity of reusable content is critical, or it becomes unmanageable and sometime untranslatable because of lost context. Reusable content should not be less than a single sentence. Writers need to keep reuse in mind, using limited cross references, but also leaving out unnecessary details, using more general words ("the product" instead of the actual product's name), and using examples that apply regardless of the context in which the content is used.

9 Management and control

9.1 Managing quality

Quality shall be monitored through the information development life cycle as defined by the workflow specification.

NOTE ISO/IEC/IEEE 26513 provides a detailed specification of review and assessment activities.

The organization shall establish guidelines and minimal requirements for quality review during the following process activities:

— Project initiation

— Content development

— Content editing and proofreading

— Content code review

— Technical review

— Content validation and verification

— Translation review

— Publication review

In each activity of the process, the organization should define standards of quality, such as the following:

a) Process requirements

   1) Quality requirements are defined for each activity in the content management workflow.

   2) Steps in the process are done according to the content management workflow.

   3) Steps in the process are completed according to the quality requirements defined for that process.

b) Information model, writing guidelines, style guide, and terminology consistency

   1) Components conform to the structural and XML coding requirements of the information model.

   2) Components conform to the writing guidelines for the correct application of organizational standards.

   3) Components conform to the appropriate style guides required by the organization.

   4) Components conform to the requirements for the consistent use of terminology as established by the organization.

c) Technical review, validation, and verification

   1) Components are reviewed by designated subject-matter experts (SME) to confirm the accuracy of the content and its conformance to system or product specifications.

   2) Components are validated and verified as technically accurate by a process in which procedural information is tested with the system or product that is delivered to the consumer.

   3) Components that contain sample code or other verifiable content are validated and verified as technically accurate.

d) Translation review, validation, and verification

   1) Components that are translated into target languages from the original source language are reviewed and verified by qualified in-country SME.

   2) Terminology used in the source language is translated into the target languages based on the content of a verified organizational terminology database.

   3) A terminology database consisting of pairs of source and target language terminology is verified by in-country SME.

e) Publication review

   1) All components are validated for adherence to the organizational requirements for final publication review before being released to the consumers.

   2) The correct conditions and variables are assigned in the final publication.

9.2 Review of content

A CCMS can be used to automate aspects of the quality management process.

Adherence to the consistency requirements of an information model, authoring guidelines, a style guide, and terminology standards can be automated using a combination of Schematron rules, quality management systems, and assisted writing tools linked to the authoring environment, including XML editing. Such systems should be configured to assist writers in creating content that conforms to their organizational standards and maintains content quality. The CCMS shall provide for integrations with such tools through its API or through direct integration with specific vendors.

As with any file changes, all suggestions made in a review tool shall be stored and maintained in the CCMS, including the changes themselves, the person who made them, the date made, and the resolution. Reports shall be available to show a document history and audit report with this information to meet the applicable requirements (these may include requirements from regulatory bodies).

The review process shall be managed through the CCMS workflow tool, which can be automated to notify individuals that content is ready for review or when a review is completed.

Other automation in the CCMS may be limited to the notification of parties associated with actions in the workflow specification. However, review activities may be supported by review systems that facilitate reviews by SME within the CCMS. Such reviews may facilitate simultaneous reviews by several experts, thereby reducing the review time and complexity. Simultaneous reviews permit experts to view each other's comments as they are made.

Review automation should facilitate storing review comments and their resolution within the CCMS and automatically create reports of the review actions and resolutions.

9.3 Approval of content

A CCMS can facilitate formal approvals of documents or topics through the use of electronic signatures. A CCMS can store electronic signatures of each designated approver within the workflow specification. The electronic signatures are permanently associated with the final approved version of the document or topic within the CCMS.

The workflow specification may include the notification of an organization's official approver or a pool of approvers for a document or topic. If a pool of approvers is available, the first available individual may accept the approval assignment, thereby blocking others in the pool from duplicating the approval process. If multiple approvals are required, each individual in the pool of approvers shall be notified that the document or topic is ready for approval. In this case, the approval process is not complete until required approvers have completed their reviews and designated their approval.

9.4 Search and retrieval

Organizations shall define how content writers search for components in the CCMS, including full-text search, metadata search, XML element search, and combinations of these methods. Organizations shall provide guidance for writers and other people searching for content that explains what search mechanisms are available and how to use them.

Organizations shall also develop policies and practices to assist writers to search the CCMS for existing topics that they may use in new publications as they plan new information-development projects. Maintaining an inventory of topics that address particular subject-matter areas as part of the planning process can reduce the practice of searching for topics at random. Writers are more likely to plan effectively from well-structured topic inventories rather than through random searches.

9.5 Localization and translation

9.5.1 Translation management

Organizations shall indicate which components are to be reused so that the components are updated once and translated once.

CCMSs support the efficient synchronization and management of source and target language XML components. Changes in source XML language components result in status changes that reflect the need for translation updates of target language components. The synchronization of source and target components occurs through the link-management function of the CCMS. In component content management, every component is also linked to maintain synchronization between source and target languages.

Managing localization practices in a CCMS enables organizations to:

— write and translate content once using content reuse mechanisms;

— use graphic formats that allow text to be stored independently of illustration content; in the case of a vector graphic, an example format is SVG;

NOTE The SVG specification is an open standard developed by the World Wide Web Consortium (W3C): https://www.w3.org/TR/SVG2/.

— using CCMS functions, isolate components that have changed from those with no changes and include only the changed components in translation jobs.

XML-based reuse mechanisms allow organizations to establish policies that require writers to isolate and maintain collections of reusable components that are referenced by XML topics in both source and target languages.

9.5.2 Content management for translation

Organizations that deliver translated content shall establish workflow policies and practices to prepare content for translation that can include:

a) allowing localization specialists or those responsible for the translation process to be assigned to the content-development workflow so that they are notified when content is ready for translation;

b) allowing localization specialists to prepare translation packages using the functions of the CCMS;

c) sending the translation packages to the appropriate localization system providers;

d) receiving the translation packages back from the localization system providers and import the translated components to the CCMS;

e) including review and validation of translated content in the content-development workflow, if required.

9.5.3 Publication of translated content

Organizations that deliver translated content shall establish policies and practices to publish content in target languages. An organization may:

— invoke the publishing pipeline in the CCMS to produce appropriate output in selected target languages;

— export the translated content from the CCMS to be published locally at the desktop or through a separate publishing system.

9.5.4 Publication of multilingual content

Organizations may choose to deliver a single publication in multiple languages. However, organizations shall confirm that the structure of multilingual content not become burdensome on the user. Separate sections for each language and accompanying illustrations avoids having users navigate among multiple areas of the text to acquire the information needed.

9.5.5 Translation of vector graphics

If vector graphics are stored in the CCMS, they may be converted to SVG, an XML format, from which text may be automatically extracted for translation. Translated text is then added back into the illustration files, producing final illustrations that may be incorporated into translated documents. Text is extracted only if the source illustration is in a format that supports layers and the text is in a separate layer.

9.6 Content deletion

The organization shall establish policies and practices for deleting components from the CCMS. Component deletion shall be reserved for authorized administrative individuals to avoid deleting components inappropriately.

CCMSs incorporate a function that automatically checks where a component is used in the CCMS. Given the nature of XML-based component writing for reuse, in which components are used in more than one publication or delivery environment, an administrator shall review the "Where used?" results so that deleting a specific component does not indiscriminately delete a component that is being used in multiple publications. If a component is used in multiple publications, the owners of those publications shall be consulted before the component is deleted. A deletion activity shall request confirmation that the translated versions of the component no longer need to be stored in the CCMS.

In general, given the low cost of storage, components need never be deleted from a CCMS. However, at times careful cleanup of the CCMS can be needed. That cleanup function should be conducted with a complete understanding of how components are stored and how links are maintained among components. CCMSs should not allow a component to be deleted if it is referenced by other components in the CCMS or if it is used in more than one publication or one version of a publication. Such systems may be configured so that writers are not able to delete components but may flag components for deletion after careful research by the CCMS administrator.

Because CCMSs maintain version control, they store multiple historical versions of every component. Deleting the most recent component can delete the version record and the previous versions of that component or delete a branch without impacting the base content. It is likely that earlier versions of the component are used in earlier versions of published output from the CCMS. Therefore, it is always unwise to delete topics and the previous versions until the components have been archived.

CCMSs link source and target versions of a component. If the source component is deleted, the CCMS also deletes the translations of that component.

Despite the concerns discussed with component deletion, a CCMS may become overwhelmed with components that are no longer used. An option is to place these components in an archive area of the CCMS or to formally archive them to another system.

9.7 Content and component archiving

The organization shall define policies and practices for archiving digital content.

CCMSs manage archiving of electronic components depending on the requirements of the organization. Some organizations require that components be archived outside the CCMS, typically in another CCMS designed specifically for archiving purposes. Some organizations require that an archived copy of a released document or set of components be managed within the CCMS so that a copy of record is maintained.

Due to the low cost of storage, component archiving does not ordinarily occur simply to save space in the CCMS.

If component archiving is necessary, organizations shall maintain policies that explain why and when components should be archived. In a CCMS, the components associated with a release, including illustrations, multiple media, and referenced components in the source and the target languages shall be maintained through archiving, so that the release can be recreated in the future. CCMSs should not allow a component to be archived if it is referenced by other components in the CCMS or if it is used in more than one publication or one version of a publication.

If components are to be archived to a CMS developed solely for archiving, organizations should consider its accessibility, format, regulatory policies, and other issues in determining their policies. See digital archiving policy in ISO 14721.

Archive policies may include:

— requirements that specify when components are to be archived, such as component age, component size, components about products or systems no longer supported, components about product or system versions no longer supported, components about products or systems no longer owned by the organization;

— format in which the components are to be stored;

— how the archived components are to be accessed;

— requirements affecting the time that components shall be maintained in an archive CMS.

Components may also be archived internally in the CCMS so that a copy of record of a publication release is preserved.

10 Publication

10.1 Release management

The organization shall define the process for publishing source content using the publishing capabilities of the CCMS.

Organizations that support multiple and simultaneous product versions shall establish policies and practices to manage multiple versions of components in the CCMS.

Many organizations support multiple product release versions simultaneously. Writers simultaneously update and edit content that is used across multiple versions of the product or system information for users and may merge those versions at a future time.

Branching and merging are widely used outside the software industry. Branching and merging can also be used to manage variants as well.

To support multiple versions of a product in the information for users, components are branched to account for ongoing changes and then optionally merged back into a main topic when the next product release occurs. When this process is handled manually, often by maintaining multiple versions of a component, merging is error-prone, tedious, and time-consuming, as the writers examine multiple copies of the component and discover that some variants are incorporated and others are not. This manual process may be automated to the extent that changes among branched components do not conflict. If changes conflict, manual intervention is necessary.

10.2 Version management

Organizations shall maintain policies and practices to avoid conflicts among writers of concurrent versions of components. These policies and practices shall address the following aspects:

a) under what circumstances to create and manage branches (multiple versions of the same base components);

b) who is responsible for creating branches of components;

c) at what level branches are to be allowed;

d) how to create and label the component branches in the CCMS.

The organization shall establish a policy and process for merging branches in accordance with the functionality of the CCMS. Once the branches are created, organizations shall establish practices for writers to follow in updating and editing branched components. Each branch is a unique instance of the components that may be independently edited. Writers working on the set of branched components work collaboratively, ensuring that decisions on potentially shared components are made consistently.

At some point, the organization may decide to merge branches into a new single version of the component. The organization may also establish a policy that some or every branch of components be maintained or discontinued.

Branched components may be merged automatically, which means that the CCMS functions to compare the branched components and institute automatically the non-conflicting changes. A non-conflicting change is typically unique, without multiple changes to the specific text.

When conflicts in branched components or other areas are discovered, the organization should have procedures and a designated authority, such as the content owner, a change control board, or by informal collaboration to decide which changes should be implemented and which should be resolved through collaboration.

10.3 Publication of content

The organization shall define the output formats (i.e. PDF, HTML, user assistance) required by their users and establish a set of publishing pipelines to enable writers to select and implement a suite of publication types. These may include:

a) functionality of the CCMS;

b) word processing and desktop publishing applications;

c) external processing systems;

d) mixture of these methods.

For documents, the publishing of output is handled within the application that was used to create the source documents. Word processing and desktop publishing files may be printed directly or transposed to PDF using the application functions themselves. For XML files, the publishing of output may also be handled within the application or may be handled through an external processing system or through the CCMS.

Documents that originate in standalone applications for word processing or desktop publishing are processed for publication at the desktop. Some organizations choose to publish XML content at the desktop as well, which can introduce inefficiencies and the inability to scale to large volumes of content. Desktop publishing can also introduce performance problems because of file size. File management may also become complex because of requirements for local versions of publishing software and templates and style sheets.

To avoid duplication of effort and technology, the organization may establish a centralized publishing responsibility. The centralized publishing authority may elect to publish through an external processing system or through the publishing functionality of the CCMS.

Organizations that develop XML topic-based content may publish individually on the desktop using the XML editor applications, may publish through external dedicated processing systems, or may publish through functionality integrated into the CCMS. External dedicated processing systems provide performance advantages because they are typically housed in independent server environments that exceed desktop publishing capacities. CCMSs provide the same capacity without removing the source material from the CCMS. They also manage maintenance of the output files in the CCMS as an archive of the published version. The source files that remain in the CCMS allow the published version to be generated on demand. Because the publishing software, templates, and style sheets are stored in the CCMS, the organization can maintain the publishing standards for every publication.

CCMSs support the publication of content for distribution outside the working content, typically to an external website, to print, to embedded user assistance, or to multiple mobile devices. Dynamic links may be established between the CCMS and the websites or other delivery mechanisms or systems so that content may be automatically updated or may allow the users to select content for a custom deliverable that is then generated automatically from the most recent approved versions stored in the CCMS.

11 Component content management system requirements

11.1 General

To manage content efficiently, provide distributed and controlled access, and maintain a single source of authoritative content, content components developed by an organization shall be stored in a CCMS. XML- structured content shall be managed in a database that stores and is capable of addressing individual components for identification and retrieval. The input, storage, removal, and output of the components in a CCMS shall be controlled through automated functions.

Using a CCMS allows each item of stored content to be independently maintained so that it may be accessed and used by individuals with the proper permissions. Each component may be added to the CCMS after writing or may originate in the CCMS.

Database support for a CCMS may be of several types:

— Relational database

— XML database

— Mixed database types that include both relational and XML structures

— Object-oriented databases.

The type of database selected depends on the needs of the organization.

Given the requirement that components be available for reuse and linking, a CCMS should manage components created using an XML structure, supported by a document type definition (DTD) or a schema. These systems are generally referred to as CCMSs because they can address and manage the individual components that are used to build larger content objects. A content object may consist of paragraphs, lists, and tables as well as more detailed semantically named structures such as <steps> or <notes>, in conformance with the associated DTD or schema. A CCMS that can deconstruct large content objects into their components allows these components to be addressed and managed individually.

11.2 Component content management system framework

11.2.1 General storage requirements

Regardless of the underlying database used, the CCMS shall store content as components in the CCMS. Components shall

a) have an associated name (human-readable label) by which a component can be referenced;

b) have a unique ID that shall be used to reference the component irrespective of its name or location in the CCMS;

c) have an assigned content type;

d) support setting and retrieving associated metadata.

When storing components, the CCMS shall provide the ability to retrieve components such that their content or linking relationships are not unexpectedly modified. These unexpected modifications include adding vendor-specific information that can affect the ability of other systems to validate, transform, or integrate the content. For XML systems, adding processing instructions is not considered an unexpected modification.

11.2.2 Content types

The CCMS shall provide for the concept of a content type. A content type is a re-usable definition of settings for storage, metadata, workflow, and behaviour. Content types enable settings to be administered to components in a centralized and re-usable manner. A component shall only be associated with a single content type.

Content types may support inheriting settings from other content types. The CCMS shall associate components with their given content type. Common reliable associations are provisioned from a basic component property (file extension, mime-type, or DTD).

11.2.3 Metadata structures

The CCMS shall support storing and retrieving metadata associated with components as metadata fields. The CCMS shall support arbitrary textual content for metadata values. However, these values may be further controlled by applying constraints or typing to fields. A metadata structure, typing, or control mechanism that is not defined by the CCMS shall be administered through its associated content type.

Metadata fields should be associated via a key-value pair.

The CCMS shall accept at least two basic forms of metadata: administrative and descriptive.

11.2.4 Administrative metadata

Administrative metadata is meta information which is provided to aid in managing components. The CCMS should provide the following administrative metadata values:

— Created date and time

— Last modified date and time

— Content type

CCMSs may provide additional administrative metadata, including:

— Owner user and group

— Permissions

— Component life cycle state

— Validation status

— Rights management metadata

This metadata should be available if the content is moved to a new CCMS without loss of information.

11.2.5 Descriptive metadata

Descriptive metadata is meta information to aid in search, discovery, classification, and identification. Common descriptive metadata includes the following:

— Keywords

— Tags or labels

— Classification

— Other taxonomic values

The CCMS should support typed metadata values. Typing metadata values allows each metadata field to be assigned a specific information type. Using information typing of metadata values means that input is validated or provided from a controlled classification mechanism.

Taxonomic metadata definitions should support storing and retrieving hierarchical values.

11.2.6 Classification

The system should provide a mechanism for classifying objects using taxonomies and schemas such as RDF/SKOS/FOAF. This system should include operations like adding, removing, and editing taxonomy terms (such as label and id).

The system should also provide a way to integrate external taxonomies and classification mechanisms. External taxonomies should be able to be surfaced in the user interface to allow writers to apply metadata to components using an external taxonomy or classification scheme. The system should also provide the ability for an external system to apply automatic classification to objects, or writer-assisted classification where the classification system can suggest high value metadata that writers can then apply to components as needed.

11.2.7 Additional metadata requirements

The CCMS shall allow the metadata schema to be updated programmatically.

If some values cannot be directly modified by the user, the CCMS should treat them as "read only."

The CCMS may also support the ability to assign metadata to components based on the associated content type definition.

11.2.8 Organizational structures

The CCMS shall provide at least one primary mechanism to organize content into logical structures. The logical structure may be accomplished through folders, metadata, publication structure, or other means.

11.3 Component content management system management

11.3.1 Component creation and modification

The CCMS shall provide basic functions for managing content in the CCMS:

— Create

— Read

— Update

— Delete

— Rename

— Move

— Copy

"Create" allows a new component to be created in the CCMS based on a defined content type. The component shall be created based on a human-readable name and the CCMS shall generate necessary administrative metadata. The CCMS shall also check that the component is valid with respect to the content-type definition. This may be accomplished via querying the user for needed information or automatically populating content and metadata based on predefined templates.

"Read" allows the content and metadata for a component to be retrieved for viewing by the user.

"Update" allows an existing components content or metadata to be updated. The CCMS shall validate the updated content and notify the user if problems are present. If validation fails, the CCMS can choose to deny the update. In this situation, the CCMS shall roll back changes that may have occurred during the update process.

"Delete" allows a component to be removed from the CCMS. Links to that component should result in a component-not-found exception. However, the CCMS shall retain that object's metadata and version information so that it can be restored to its previous state at a future time.

"Rename" allows a component's name to be changed. When a rename occurs, the CCMS shall update links to that component from other components as part of the process.

"Move" allows a component's apparent location in the primary organizational structure to be changed. When a move occurs, the CCMS shall update links to that component from other components as well as links from that component to other components, to reflect its new location in the CCMS.

"Copy" allows an object to be duplicated in the same location or to a new location and given a new object name.

When applying these functions, the CCMS shall update links to that component from other components as well as links from that component to other components during the process.

11.3.2 Import/export

Acquisition is the process of importing existing content into the CCMS. The CCMS shall provide a way to import content into the CCMS. The CCMS shall provide the ability to import components from and export components or content to a local file system or a cloud environment. During the import process, components that do not exist in the CCMS shall be handled by the "Create" function. Components that already exist shall be handled by the "Update" function. If an issue occurs during the import process, changes made during the process should be rolled back.

The CCMS should also preserve the organizational structure of imported components.

11.3.3 Bulk export

To support bulk export, the CCMS shall provide a mechanism to download a set of files as a single package. When the user selects a folder or a set of files to export, the CCMS shall package and deliver those files to the user in a format that can be unpacked and used on a local file system. The CCMS shall support bulk export.

11.3.4 Dependency export

The CCMS shall allow dependency export.

The CCMS shall provide a mechanism for a user to download a packaged set of files and its dependencies based on internal linking structures.

The CCMS shall not export non-local dependencies or dependencies that are marked as "external" in the linking structures. Links to websites, reference materials, or other content that does not exist locally shall not be exported.

The dependencies of an object are the other objects on which the object depends through linking. Dependencies include objects linked to by the object as well as the dependencies of those objects.

EXAMPLE Object A can link to object B and object B can link to C, resulting in A->B->C. The dependencies of object A are B and C.

11.3.5 Archiving

The CCMS shall create an archive or snapshot of a set of content and its subsequent dependencies at a specific time. An archive shall include source content, translated content, and published versions. Content or metadata in an archive shall not be modified. If users need to modify content, they shall do so by exporting or branching content from the archive.

Components in an archive, including content and metadata, shall be retrievable in the exact state in which they existed at the point of archiving.

11.4 Content object check-out and check in

11.4.1 General

The CCMS shall check components in and out of the CCMS. When a component is checked out, it shall be locked. For locked components, the CCMS shall indicate that the component is currently checked out by a specific user. The CCMS shall prevent other users from checking out a component that has been checked out by another user. If another user attempts to check out a locked component, the CCMS shall deny this operation and indicate the user who currently holds the lock.

Read access to locked content is allowed. Readers can review a complete list of checked out components and identify which user has locked a specific item.

When components are checked back in, the CCMS shall remove locks put on these components.

The CCMS shall allow single component check-out/check-in, as well as bulk check-out/check-in.

The CCMS shall allow administrators to unlock components regardless of which user originally locked them.

11.4.2 Bulk check-out/check-in

The CCMS shall select and check out or check in an arbitrary set of components from the CCMS. The CCMS shall also support dependency-based check-out/check-in. When dependency-based check-out/check-in is invoked, the CCMS shall check-out/check-in the selected components, as well as components that are direct or indirect dependencies of those components.

The CCMS shall not allow a user to perform a dependency-based check-out/check-in if a component in the set of dependencies is checked out by a different user.

11.4.3 Check-out and check-in after restart

In the event of a CCMS restart, the CCMS shall provide the following capabilities:

a) Retain locking information for components.

b) On check-in, trigger an incremental version of the components either automatically or manually.

c) Connect writing tools directly to the CCMS, without the user needing to checkout and download components manually and set components' check-out/check-in state.

d) Prevent users from moving, renaming, or performing other operations that can affect links contained in components that are currently checked out by other users.

EXAMPLE If document A links to document B and document A is checked out, document B cannot be renamed or moved until document A is checked back in.

11.5 Link management

The CCMS shall manage links between components.

The link management system shall provide the following reporting capabilities:

For each link, the system shall report to the user the type of link: relative, absolute, or external. For relative and absolute links, the system shall report if the link is broken. A link is considered broken if the CCMS cannot resolve the component being linked to.

For each component, the system shall list links contained in that component and the destination component of each link.

For each component the system shall list every link that references that component. This function is often called "where-used".

For each component, the system shall list the components that are dependencies of that component.

The link management system shall automatically correct links referencing components when a component or set of components is renamed.

11.6 Search

11.6.1 General

The CCMS shall allow components to be found by querying the CCMS. When the CCMS is queried, the CCMS shall take the access control settings of the CCMS into consideration, filtering components from the result list to which the querying user is not permitted to access.

In addition, the CCMS shall allow queries of the entire CCMS, as well as confining the query to a specific set of components. The CCMS shall allow confining a search query to the

a) set of selected components and confined only to those components;

   EXAMPLE 1 A search can be confined to a specific folder or a selection of files. A user can select files and folders manually, or it can be selected via other methods like taxonomy.

b) single component and only to that component;

   EXAMPLE 2 A user wants to search inside a single component if that component is very large.

c) single component as well as that component's dependencies and only the component and its dependencies.

EXAMPLE 3 Someone wants to search a map and all the files in a map, or all the files that a file links to.

During a search, matched results shall be scored based on their relevance to the search query. Higher scored results shall appear higher in the list of returned search matches.

The CCMS can also combine two or more querying methods to further refine the matching results.

The CCMS should save searches so that the user can invoke the same query without having to enter it again manually.

11.6.2 Full text search

11.6.2.1 General

In a full text search query, the CCMS shall return matches based on text content in components.

The CCMS shall allow for entering more than one search term. The CCMS may consider phrases to be indivisible, as a single unit, if the user designates that the whole phrase should be used as an exact match.

The CCMS can also support advanced matching capabilities. Advanced full text matching can include:

— Near search: Matches are scored higher the closer together the terms appear in the component.

— Wildcard search: The user can specify a wildcard value in a search term that will match a set of characters that results in a match for that term.

EXAMPLE 1 “digg*” matches “digged”, “digging”, “diggings”, “digger”, and “diggers”.

— 'Regex' regular expression (Regex) search: The search query is a regular expression.

— Phrase search: Treats the group of full text search terms as an ordered set that is to be matched to content in the source components in the same order as supplied in the query.

— Boolean search: For each term, the user can specify whether terms are a connected set with Boolean logic. This logic states if terms shall appear, can appear, or are not permitted to appear.

EXAMPLE 2 This query states: Return documents that match terms “dog” and “cat” or match “horse” but do not contain “donkey”.

(“dog” AND “cat”) OR (“horse” AND NOT “donkey”).

The CCMS should report on which terms matched a given component and show a highlighted excerpt of the content where the match was found.

Full-text search shall include auto completion of search queries and spelling corrections or suggestions.

The CCMS should save searches so that the user can invoke the same query without having to enter it again manually.

11.6.2.2 Full text search in XML repositories

For CCMSs supporting XML content, the CCMS shall:

a) not consider XML tag names, namespace declarations, comments, processing instructions, or attribute names as part of the full text content being searched;

b) tokenize component content on whitespace based on XML whitespace processing rules;

c) tokenize component content based on the XML structure; thus, the CCMS shall recognize that the tags between texts imply a break in the text token.

EXAMPLE In the following markup, the CCMS recognizes that “Item one” and “Item two” are independent clauses, even though they are not separated by whitespace or punctuation.

<ul><li>Item one</li><li>Item two</li></ul>

11.6.2.3 Full text search configuration

The CCMS should support search configuration for full text search on structured content. In a CCMS with XML content, the structure should be used to improve searching capabilities, such as:

— Boosting matches: The CCMS should boost the score of matches found in particular parts of a component.

EXAMPLE 1 The CCMS is configured so that matches in the "title" of the component are scored higher (given a higher ranking in search results) than those found in other parts of the document.

— Ignoring matches: The CCMS should ignore matches found in designated parts of a component.

EXAMPLE 2 The CCMS can be configured to ignore matches found in alt text for images or information in files used for processing.

— Rules for mixed content: Mixed content occurs in structured documents when markup splits words or phrases that would otherwise be considered contiguous. The CCMS should configure which elements split content that should be considered contiguous.

EXAMPLE 3 In the following structured content, the word "unclear" is split but is intended to be considered a single entity: <p>This is <b>un</b>clear</p>.

11.6.3 Metadata search

In support of metadata search, the CCMS shall match queries against component metadata. For XML-based CCMSs, this matching can include metadata inside components.

Faceted search, often called faceted search navigation or faceted browsing, allows the user to search components through classification collections. Each collection represents a predefined facet on which the user can search.

When searching, the CCMS shall present the user with a list of facets. Each facet shall be presented with an indicator informing the user how many search hits belong to that particular facet with respect to the original search query. The user can select one or more facets, and the CCMS shall return results based on the intersection of the results from each facet.

11.6.4 Structured search

Structured search allows query targets to specific parts of the XML component structure using the XPath specification. Structured search is often combined with other forms of search to return highly specific results over large content sets. The CCMS shall provide a user interface to allow the submission of a structured-search query.

However, the CCMS should prevent the submission of malformed queries for malicious purposes.

EXAMPLE A user wants to search only components that contain a source code example inside procedures:

//code[ancestor::task]

Or, a user wants to find each frequently asked question that also contains an image and an example:

//faq[image and example]

11.6.5 Advanced search capabilities

11.6.5.1 Stemming

Stemming is the process of reducing search terms to their stem form. For instance, cats can be reduced to "cat". An aggressive stemming system can reduce "argued, argues, and arguing" to "argu".

11.6.5.2 Taxonomy support in search

Taxonomy support in search allows a taxonomic set of terms to influence how search is performed. The most common improvements to search are query expansion, search refinement, related search suggestions, and highlighting of related terms.

With query expansion, the CCMS shall analyse the search query and augment it based on the taxonomy to search on broader, narrower, or equivalent terms based on the situation.

EXAMPLE 1 Given the following taxonomy:

"Medical professionals"

"Nurse"

"Nurse practitioner"

"Doctor"

"Oncologist"

"Pediatrician"

"Cardiologist"

The user can search for “doctor” and the CCMS expands the search query to also include “Oncologist”, “Pediatrician”, and “Cardiologist”.

With search refinement, the CCMS shall analyse the taxonomy and indicate to the user that there are narrower terms that produce a more refined search.

EXAMPLE 2 If a search for “doctor” returns a large number of results, the CCMS suggests to the writer to search on a narrower term such as “Oncologist”, “Pediatrician”, or “Cardiologist” from the taxonomy.

Like search refinement, the CCMS shall analyse the query and suggest other related searches based on the taxonomy.

11.6.5.3 Fuzzy matching

With fuzzy matching, the CCMS shall recognize terms that are not exact matches but similar enough to be considered relevant. Fuzzy matching is especially important when users spell terms incorrectly or use shortened versions of words. The degree of similarity of terms is often measured by the Levenshtein distance.

11.7 Versioning

11.7.1 General

The CCMS shall track version information for components in the CCMS. The CCMS shall maintain a version history for each component that tracks changes made to the content or metadata of that component.

The CCMS shall also maintain a local version number for components that can be used to return information about the committed change. The CCMS shall retain the following information for each committed version:

— User who committed the change to the component

— Time at which the change was committed

The CCMS shall have the ability to:

a) return the content of a component at a version in the component's history;

b) return the content for a set of components at a specific time in version history;

c) show the differences as insertions and deletions between the content of a component at two points in the component's version history;

d) restore a component's content and metadata to a specific point in the version history.

11.7.2 Branch and merge

The CCMS shall allow the branching of a set of components. When branched, the CCMS shall create a clone of the original components that shall exist separately from the original components. This clone is often referred to as a 'parent branch' and 'child branch', where the parent is the original content and the child is the clone. The clone reproduces the content, structure, and metadata of a component. Changes made to branched components do not affect components in the parent branch or other branches. The CCMS shall maintain relationships between the parent and child components so that a parent component can be retrieved from a given child component and vice versa. The CCMS shall offer the same functions on branches that are available on the original content."

The CCMS shall allow a branch or a subset of a branch to be merged with its ancestor or a derivative of its ancestor. The CCMS shall also allow changes to be merged from an ancestor into a branch or a derivative of a branch. When merging, the changes in one component set shall be transferred to the other in both content and structure. Merging can result in conflicts that require manual intervention. The CCMS shall provide a user interface to resolve conflicts in merging.

Ideally, the CCMS shall provide a visual indication of conflicts/changes between files, both at the component and publication level and allow them to be accepted or rejected to complete the merge process.

The CCMS shall report on the last time a branch or subset of a branch was merged and what content was affected.

11.7.3 Release management

The CCMS shall allow concurrent working, future (or parallel), and past releases. Releases shall have the following associated metadata:

a) the history of the given release, which shall contain a log of past or parallel releases which are ancestors of the current release;

b) a name and target version number for the release.

A working release encompasses the unbranched content that is targeted for the next release of the publication. A working release can have a past release, but it cannot be a branch of a release. A past release is an archive of the publication and its components at the point in time when that publication was released.

As an archive, a past release is frozen, and as such the CCMS shall prevent changes to components in a past release. The CCMS shall instead allow a past release to be branched and modified and then merged or again released. A future or parallel release is a branch of a past or working release. The CCMS shall allow work on parallel releases concurrently with the working release or other releases.

11.8 Graphics and multimedia management

The organization shall specify how graphics and other media types shall be managed in the CCMS, in accordance with the capabilities of the CCMS. Graphics and other media types shall be maintained with the version control functions of the CCMS.

The CCMS shall store digital images, including graphics and other media types. Appropriate graphics formats that are widely used and are recognized industry standards, such as JPEG, GIF, BMP, PNG, SVG, and others, should be supported.

The CCMS shall allow metadata to be applied to graphics and other media types to aid in search and retrieval. See 11.2.3 for more details on metadata. The CCMS may use metadata embedded in digital assets to populate metadata fields.

After the text has been translated into target languages, the CCMS shall allow parallel, linked storage of the translated versions of the graphics files so that changes to the source language versions will initiate the re-translation of the target language version of the graphics.

The CCMS should store video and sound files in commonly used formats, such as WMV, SWF, MPEG, QTFF (.mov), WAV, MP3 and MP4.

Image formats with extractable text can be managed as part of translation packages provided by the CCMS.

EXAMPLE Text within SVG files can be extracted and packaged for translation.

11.9 Component content management system administration

11.9.1 User administration

The CCMS shall allow for the creation, change, and deletion of a user and the users associated properties. A CCMS user is a person who participates in the creation, editing, and distribution of content.

The CCMS should include the following capabilities:

— have a programming model and interface that is flexible enough that features can be turned on and off (or displayed and hidden) to provide a user experience tailored to the access level of the user;

— have a CCMS structure that is flexible enough to allow or deny access to components individually or by type;

— allow for the definition of access levels such that access to CCMS functions and components can be granted or denied to each access level;

— allow users to be assigned to or removed from access levels.

11.9.2 Security

Because some vulnerabilities allow unauthorized persons to run malicious code and thus take over the system or obtain access to private or proprietary information, the CCMS owner shall take precautions to protect the content and the users and to help prevent the system from becoming a host of malicious code or other attacks.

The CCMS shall provide levels of user authentication using methods such as password or biometric and can use two or even more methods. If password or other information is stored for use in authentication, it shall be encrypted or cryptographically hashed or otherwise obfuscated.

The CCMS should log the dates and users involved in the following activities:

— User login and logout

— User create, change, and delete events

— File create, change, and delete events

Additionally, the CCMS should sort or filter on date, user, and event type and report the filtered and sorted results.

As with IT systems, CCMS security encompasses human access as well as machine-to-machine interfaces, including allowing remote control and monitoring. CCMS hardware and software are subject to security vulnerabilities.

For an audit of the CCMS security, guidelines are provided in the ISO/IEC 27000 family of standards.

The CCMS shall apply the following security controls:

a) provide a method to authorize users and securely manage associated sessions; if the system stores sensitive user information, such as passwords, they should be encrypted; encryption should be used to secure transmitted user data;

b) prevent users from elevating their privileges without re-authenticating;

c) not transmit sensitive data in order to assist a user retrieving a lost or forgotten password or other personal information that can be used to compromise the security of the user's account;

d) provide encryption for sensitive data that can be exchanged between remote clients;

e) provide sufficient and safe logging for unusual conditions, monitoring and alerting facilities to allow audit;

f) enforce the roles, permissions and responsibilities of users as they apply to the CCMS;

g) prevent third party applications, such as web browsers and writing tools used to interface with the CCMS, which can have their own security vulnerabilities, from granting access or privileges that can compromise the security of the CCMS.

NOTE Controls as defined in ISO/IEC 27001 can be used as guidance or requirements. The ISO/IEC 27000 family of standards detail techniques for information security management.

Security designations or characteristics should be included when content is displayed or printed. The exact wording can vary in different organizations and can have legal implications (which can vary by country). Typical security banners include the following:

— XYZ Corp. Confidential

— Internal Use Only

— Public Information

Pages without appropriate security designations can be implicitly public information (even though protected by copyright) or lacking in essential legal protections, depending on the legal jurisdictions from which they can be accessible. The security designation does not ensure automated enforcement of the security designation. Declaration of security designation should not be considered sufficient to provide security control. CCMS design should include evaluation of passwords, encryption, and other techniques to provide additional security controls.

A qualified person associated with the CCMS owner should assess the adequacy of the security indicators and security protection for the system. Reviews should be at regularly scheduled intervals, as a result of a review-triggering event (e.g. system modification, change in ownership).

11.10 Content creation and acquisition

11.10.1 Writing interface

Writing is the process of creating new content. CCMSs can provide native writing functionality or import content using integration with an external writing interface. If the CCMS provides a native writing function (a writing interface), that interface shall:

a) allow for the definition of content types;

b) provide an interface for writing components of the types that have been defined;

c) guide or constrain the writer, based on the definition of the information within and metadata around the content type being written.

11.10.2 Content and metadata functions

The CCMS shall describe the structure of the content type being written and allow:

a) creation of metadata elements to be applied to content types;

b) addition of metadata elements individually to content types;

c) display of metadata elements during the writing process;

d) storage and retrieval of content objects between the writing interface and the CCMS;

e) deletion of components in the CCMS from the writing interface;

f) creation of a content hierarchy into which components can be added and support component-to-component links;

g) creation of multi-paragraph text elements.

The CCMS should validate and enforce the structure to:

— allow the creation of independent metadata elements with controlled vocabularies that can be applied to one or more content types;

— apply metadata through drop-down lists and other methods to consistently apply controlled vocabularies;

— support a variety of component organizational structures to be applied to components from the writing interface;

— allow the creation of rich text elements that include block and inline semantic tagging (preferred) or format tagging (less preferred).

11.10.3 Scientific notation and vector graphics

The system shall provide the ability to work with equations using a standard such as Mathematical Markup Language (MathML https://www.w3.org/TR/MathML3/). The system shall be able to allow for editing and publishing of scientific notation, such as mathematical expressions, chemical formulas, and so on to all formats supported by the system.

The system shall provide the ability to work with vector graphics such as SVG (https://www.w3.org/TR/SVG2/). The system shall allow users to manage, insert and format vector graphics the same way they would manage other media in the system. The system shall publish vector graphics to all output formats supported by the system.

11.10.4 Writing integration

Integration with an external writing environment can be direct, where the CCMS projects its user interface into the external environment, or indirect, where the CCMS consumes the independently created output of an external environment. Indirect integration is considered acquisition as discussed in 11.10.5. If a direct writing integration CCMS is provided, the integration shall allow the writer to:

a) specify the target content type into which an externally written component is to be stored;

b) add additional metadata to the component created that is not supported by the external interface;

c) guide or constrain the writer based on the definition of the information within the content type being created;

d) store, retrieve, and delete components between the external environment and the CCMS.

The system should provide the ability to enforce style guide rules via technology such as Schematron or a similar technology. Schematrons should be able to be configured to report warnings and errors to writers as they are editing content. The system ideally should also provide the ability to suggest fixes to content that the writers can apply automatically to address issues in content.

11.10.5 Acquisition

Acquisition is defined as adding existing content from an external source into the CCMS. The CCMS can include acquisition functionality. If acquisition is included, the CCMS shall:

a) parse input into a single component, mapping the parts of the input to elements of the content type;

b) log input parts that cannot be segmented or parsed and distinguish between mandatory and optional elements in the target content type;

c) validate the components that are created;

d) track the file name or query that constitutes the input content, the date and time of the acquisition event, and the identifiers of the components that were created and rejected during the event;

e) fill additional metadata (not present in the input) with default values.

If acquisition is included, the CCMS should allow:

— input content to be targeted to content types in the CCMS (or put in the CCMS as non-categorized components);

— a single input to be segmented into multiple components;

— the specification, detection, and logging of useful exceptions during the acquisition event;

— monitoring, review, and configuration of acquisition events;

— an administrator to specify rules for how to fill additional metadata based on values that are present in the input.

11.11 Workflow

11.11.1 Workflow creation

A workflow is a named sequence of states that components of a particular kind are expected to go through from their original creation to their eventual retirement (by archiving or deletion).

The CCMS shall allow:

a) creation of a single workflow;

b) creation of a status (or state) metadata field with a controlled list of values that can be applied to each content object;

c) administrators to apply the status field to content objects so that it is part of the metadata of each content object;

d) writers to see and change the value of the status field in a native writing CCMS if implemented or in a separate interface if native writing is not supported;

e) tracking the date and time for each status change;

f) administrators to override writer set statuses;

g) writers and administrators to view content objects sorted by status;

h) suppression of given statuses in publications that the CCMS creates.

11.11.2 Workflow specification

The CCMS should allow for the creation of a range of named workflows. An enhanced CCMS should allow administrators to:

— create (name), edit, and delete a number of workflows;

— add, edit, and delete the steps that are within a workflow;

— add, edit, and delete a set of tasks (e.g. deliverables, instructions, examples) that can be applied to the steps of various workflows;

— define mandatory and optional steps as well as conditional branches in the steps;

— create roles (e.g. writer, editor, reviewer) and assign CCMS users to those roles;

— assign either individual writers or roles to workflow steps;

— apply workflows to individual components or to entire content objects such that components will go through the same steps;

— define automatic and manual workflow triggers;

  — a manual trigger occurs when a writer or administrator performs an action in the user interface to cause a step change;

  — an automatic trigger occurs when a CCMS event (e.g. when a date and time is reached or when a new file is added to the CCMS) causes a step change;

— integrate workflow with the localization CCMS such that localization can be made part of the overall workflow of a content object.

An enhanced CCMS should allow writers to:

— see the roles they have been assigned;

— perform workflow tasks from their writing environment sorted by content object, role, or workflow step they are responsible for;

— see a list of the content objects for which they are responsible in the workflow and indicate the writer's current workflow step;

— see a list of the content objects that are waiting for the writer to progress along with the time that the content objects have been waiting;

— display their content objects and, when appropriate, advance their content objects to the next workflow step and log comments they can have about the current step;

— pause a workflow and log a comment about their reasons for doing so.

11.11.3 Workflow reporting

The CCMS shall allow an administrator to view the workflow status of each content objects in the CCMS sorted by status, content type, or date of the last change.

The CCMS should:

— support a workflow audit trail that logs the steps transitions for each content objects;

— allow for sorting and filtering workflow reports by workflow, content object, step, role, user, date and task;

— allow for export of the reports into common table formats, such as CSV or tab delimited.

The system should provide the following list of reporting capabilities. For each, the system should be able to run a report in the "context" of a set of components that can be grouped into folders by metadata or a set determined by the linking dependencies of another component:

— Validation reporting: The system should provide the ability to report on components that have validation errors.

— Comment reporting: The system should provide the ability to report on components that have open comments that need resolution.

— Workflow status reporting: The system should provide the ability to report on the status of components and a breakdown of the number of components in each workflow status.