**Alice: Concepts & Facilities**

v1.1, 28 November 2003

**Alice** is a unique metadata-driven collaborative content management product: a flexible, scalable software suite, designed to facilitate the collaborative, iterative development and management of high-value, knowledge-intensive content through the use of metadata extracted from XML objects or associated with non-XML objects both to represent structured collections of documents by means of multi-contextual views and to track reusable content components and content change units, and implemented as a set of modular services that plug into a multi-tier architecture to connect through standard protocols to an organization’s existing workflow, data and content management applications.

1. Purpose & Scope

The purpose of this document is to explain the capsule description of Alice given above, to show at a conceptual level both what Alice does and how it does it, and to clarify the nature of metadata and its role in sustaining the functionality.

2. Alice?

We are often asked “Why Alice?”, “What does the name mean?”. It is not an acronym¹. It was applied originally as an internal code name for an R&D project in metadata management, the first in a sequence of projects designated by arbitrarily selected personal names. The others have all fallen out of use – Bob became our Tagless Editor, now superceded by Clara, otherwise known as x4o – but Alice has persisted, largely because of its association with Lewis Carroll’s Alice.

Charles Dodgson, the person behind the pseudonym “Lewis Carroll”, taught mathematics and logic, invented a number of games for teaching and exploring logic, and filled his Alice books with word play and puzzles about logical categories, properties and relationships. Consider the dialogue between Alice and the Knight from *Alice Through The Looking Glass*:

‘The name of the song is called “Haddocks' Eyes”.

‘Oh, that's the name of the song, is it?’ Alice said, trying to feel interested.

‘No, you don't understand,’ the Knight said, looking a little vexed. ‘That's what the name is called. The name really is “The Aged Aged Man”.

‘Then I ought to have said “That's what the song is called”?’ Alice corrected herself.

‘No, you oughtn't: that's quite another thing! The song is called “Ways and Means”: but that's only what it's called, you know!’

‘Well, what is the song, then?’ said Alice, who was by this time completely bewildered.

‘I was coming to that,’ the Knight said. ‘The song really is “A-sitting On a Gate”: and the tune’s my own invention.’

¹ At the time this document was created “Alice” was not an acronym. As both the terms “Project Alice” and simply “Alice” evolved, our customers and partners became quite fond of name. Rather than depart from the Alice theme, i4i has chosen to provide it with the requisite acronym. ALiCE is now referred to as i4i’s Authoring Lifecycle Collaboration Environment (the “i” in the middle represents i4i). Clever, we think, and in the spirit of Mr. Dodgson.
It’s a spoof about metadata, about the names of properties, their value domains, and the sets of objects designated by name-value combinations. Like any spoof it carries a grain of truth, in this case the significance of the levels of logic involved in categorization, but for humor it exaggerates the confusions that arise between natural language and formal classification schemes. In its way it also illustrates a very simple multi-contextual view of a single object, the song in question. So it seems entirely fitting that the fictional Alice, a clear-thinking, independent-minded person often confronted by categorization puzzles, should also name our metadata management product.

Throughout this document we have tried to use terms consistently, but like one of the characters in the Alice stories, when we use a term it means precisely what we chose it to mean, neither more nor less. The terminology of the content management industry is not as established or based on a formal system to the extend that it is in, say, the relational database industry. The reader might wish to consult the Glossary in Appendix I, and should note particularly that we use the term “metadata” to refer to the descriptive form as opposed to the prescriptive form.

3. What’s Alice For? The Application Problem

There are a variety of document and content management (DM/CM) systems on the market, many of which are metadata driven and most of which support collaborative work. Is Alice merely another competitor in an already crowded market? No, Alice is different in two important ways: (a) it addresses the general problem in a different application context; and, (b) it uses metadata to organization and manage the content collection within its purview in a novel and very flexible way.

If you look at the market positioning and historical development of most of these DM/CM systems, one or both of two application contexts are prominently featured: publishing and/or business process automation (BPA). Most of these systems were developed in the context of print or web publishing, where the content typically comprises marketing information, technical specifications, user manuals or the like, and where the production process is largely linear, moving through a sequence of steps from authors through editors to production/distribution. A few were developed in the context of document processing, as in the insurance industry for example, where the content typically comprises policies, claims, adjustments or the like, and where the production process often branches through a network of specialized steps guided by strict business rules.

There is another application context that is not squarely addressed by these systems with roots in publishing and business process automation: the management of large, complex document sets, often referred to as dossiers, which embody both significant intellectual property and legal obligation, and which are developed and maintained over very long life cycles by highly specialized subject matter experts. A prime example of this context is regulatory compliance in the pharmaceutical and biotech industry, where the content comprises large sets of technical documents that carry proprietary information about bioactive materials and information about their quality, effectiveness and safety that binds the owners under the law and exposes them to significant liability, and where the process involves contributions from a broad range of subject matter experts over many years.

This application context stands out as different from the more traditional application contexts in several important ways: (a) the extent to which the content carries intellectual and legal value; (b) the number of subject matter experts who contribute to the development of the content; (c) the complexity and variability of the classification of dossiers, (d) the high occurrence of re-usable material and cross-references, and (e) the long life spans of documents and dossiers.

Given the complexity and variability of dossiers, applications in this context demand a greater degree of flexibility for the content administrator and the end-user, and, given the IP value and liability, they demand a higher standard of consistency and control than the traditional
applications. The larger pharmaceutical companies have installed DM systems based on the
publishing model, under the rationale that they are “publishing” submissions. This has the
positive effect of providing the necessary controls, but creates a layer of operations separated
from the on-going knowledge development behind it. This separation adds labor, time and cost,
and does not usually include adequate control over the supporting documents not part of the
submission proper nor integrate the feedback from the regulatory agencies into the process.

With the largest volume and highest financial stakes in this kind of content, the pharmaceutical
industry has moved to improve their situation through the Common Technical Document (CTD), a
modular and highly structured document set aimed at incremental exchange of documents
between the company and the regulatory agency, as opposed to publishing whole sets. The
transition to the CTD offers the opportunity to extend their controls to encompass both more of
their knowledge development process and the interaction with the outside world.

DM/CM systems of both lineages are more about control than flexibility, and in consequence
have rigid, multi-step user interfaces requiring special training, and limited or no capability for the
creation of new forms of content organization by document administrators: most require
programmer intervention. For example, most DM systems use a document profile as the user-
visible metadata entry form, predefined for each “application” (i.e. a file format) so that all Word
documents have the same set of metadata elements. Content administrators are forced to
choose between short profiles, which provide less descriptive detail for many documents classes
but which users are more likely to complete accurately, and long profiles, which provide sufficient
detail for those classes that need it but which users tend to treat as a nuisance. The profile form
also serves as the framework for a query to a set of documents, but provides no mechanism for
hierarchical or other forms dossier organization without custom programming.

While it is entirely possible to build a layer of application code on top of a traditional system to
achieve this result, Alice takes a different approach from the ground up. It focuses not on the
content objects themselves, but on the metadata as the key to meeting application requirements:

(a) By separating the metadata from the content objects and managing it independently,
Alice provides the flexibility to create classified organizational views as needed, while
ensuring the integrity of the metadata.

(b) By basing workflow, versioning, and change unit management on metadata, Alice
provides a mechanism for tracking the history of every document, and with its parallel
eedit/review capability enables a high degree of collaboration among the many participants
in its creation.

(c) By embedding XML content as a integral part of the system, Alice enables effective re-
use and cross referencing with minimal effort – the metadata tracks all this – and
supports an XML editor that enables subject matter experts to create and edit valid XML
content in an intuitive way in the context of familiar document structures with no
knowledge of the underlying XML.

(d) By using industry standard protocols, and by separating the metadata from the content
objects, Alice provides a framework that is easily configured to integrate with other
services and applications in a multi-tiered architecture and connect with existing DM/CM
repositories to protect sunk IT investments.

The Alice approach, based entirely on metadata, supports a user interface based on the widely-
understood, intuitive “folders and files” metaphor embodied in Windows Explorer or IE. It needs
few operations besides the obvious check-out/check-in, and by embedding the controls in
metadata through required properties and controlled vocabularies, it enables the simplest form of
user compliance: only a few values to be added or updated on check-in through a simple form.

Folder structures can be created or re-organized by document administrators without programmer
intervention as needed by particular work groups without in any way compromising the underlying
controls required by the whole process. Even individual users can construct folders to their own
preferences without compromising system integrity. Metadata definitions are controlled by the system administrator, but new ones are easily added and existing ones easily extended.

XML authoring is important if the advantages of change unit management and parallel review of documents is to be exploited fully. Here too Alice provides that same combination of flexibility and control. The x4o editor can be used by any subject matter expert who can work in MS Word, without any special knowledge of XML. The editor produces valid XML in a constrained authoring environment that is easy to use because it presents familiar document structures and context-dependent options and does all the XML work behind the scenes. The x4o Designer enables document administrators to create new classes of XML documents or update existing ones for use in the system based on their knowledge of the logic of the document (i.e. the visible topical structure, and the user-controlled embedded metadata properties). They do not need a programmer’s knowledge of XML; all they need to understand is the nesting structure of topics and the names of the properties and where they may be used in the document.

Alice provides a framework for a tightly controlled dossier, such as the CTD, with the flexibility needed to integrate the dossier’s production and use into other functional areas and seamlessly incorporate other documents as needed. Alice does not reduce control by enabling flexibility; it merely reduces the visibility of the controls so that the system can be used directly by the knowledge workers who create and use the content in a manner that serves their needs as well as the overall needs of the organization.

4. Alice Architecture

4.1 Conceptual Architecture

The term “conceptual architecture” refers to the abstract models used to frame the specification of the structures and functions embodied in Alice. There are eight principal models involved:

(a) The metadata model, framing the way in which the software components of Alice are organized and communicate with each other;
(b) The collection model, framing the way in which content objects are represented and managed, and how higher-order collections of content objects are represented, organized and managed;
(c) The workflow and versioning model, framing the way in which stages in a document’s life cycle and individual versions of content objects are represented and managed;
(d) The content reuse model, framing the way in which re-use of content objects is enabled, represented and managed;
(e) The cross reference model, framing the way in which cross references (links) between content objects are represented and managed;
(f) The change unit model, framing the way in which individual changes to XML content objects are identified, represented and managed;
(g) The security model, framing the way in which user access rights are represented and managed; and,
(h) The inter-working model, framing the way in which other applications and servers can be interconnected with Alice.

4.1.1 Metadata Model

All objects known to Alice carry metadata, consisting of named metadata elements with a corresponding value. These name-value pairs are often referred to as properties.

There is a core set of metadata elements carried by every object in the system, though not all of these elements must necessarily have a value on every object. The default core set is the Dublin Core standard (plus some intrinsic system metadata). Other default sets can be implemented on a given Alice server. Additional sets of metadata, distinguished by namespace names used to qualify the property names, can be defined by the system administrator and used
on objects as needed. Metadata sets for ICH CTD content, MARC and other standards are available.

For each given class of content object, properties may be specified as optional or required (must have a value, and single or multi-valued (meaning that the property name may occur more than once with different values in association with the object).

Metadata elements can be further constrained as having a prescribed domain of values (i.e. a controlled vocabulary taken from a list of permissible values) or arbitrary values of a given type (e.g. a text string or a date).

The intrinsic system metadata comprises elements that designate object ownership within the framework of the security model, and housekeeping elements, such as Last Modified (date/time), State, Version, Type and Deleted. Most security and housekeeping properties cannot be modified directly by the user.

The metadata model is then simply a flat space of properties, name-value pairs, subdivided into sets of properties designated by namespaces which qualify the names.

Alice also maintains other information that represents relationships between objects rather than properties of objects. Though it is considered metadata in a general sense, such information is logically represented, not as a name-value pair, but as a relationship type plus a pair of object ID's. Inclusion, or re-use relationships, and cross-reference relationships are two of the important relationships tracked by Alice.

On a technical note, the reader with a knowledge of relational DBMS practice will recognize that the formal logical difference between properties and relationships is reflected in the rules for their update and use but that both can be stored as triples in a relational table, in the form Object ID, Property Name or Relationship Type, Property Value or Object ID respectively.

4.1.2 Collection Model

Alice exploits descriptive metadata to provide multi-contextual views of the collection. Alice recognizes two classes of objects:
(a) content objects, which are content bearing files in any format, represented as typed files in Alice views; and,
(b) dynamic troves, which are queries on metadata, represented as folders in Alice views, and hereafter referred to as folders.

When any object is checked-in through Alice, the metadata manager extracts the metadata elements and their values from the object using XSLT and updates its metadata repository. On folders and non-XML content objects, metadata is applied by the user when the object is created or checked-in. On XML content objects, some of the metadata may be embedded in the content itself, such as title, or elements designated as "in-line semantics".

The metadata repository is essentially an index of properties (names and values) for each object; the objects themselves, including folders, are stored in a content repository.

Folders are essentially a query against the property index. The query retrieves the metadata representing a set of objects, whose members are defined by a logical combination of names and values in the query. The details of the queries are set out in “Creating Folders” in Appendix II. Folders, like content objects, carry properties. Therefore a folder may retrieve folders as well as content objects; i.e. folders can appear within folders. Thus a hierarchy of sets of objects can be constructed, as shown in Figure I.
Since content objects, and folders too, are retrieved by property, they can appear in any folder whose query selects their properties. There is no particular “place” that the object is anchored to; such views are called placeless. Since folders can be created to select on any combination of properties, different views of a document collection can be created to meet the needs of individual users or work tasks. This combination of placeless views and multiple, property based views is said to be multi-contextual.

When an end-user opens a folder in an Alice view; the query is executed and the metadata retrieved is used to display the set of objects as the contents of the folder. The user can display all of the properties and other information associated with each object, open folders, or check-out objects.

Folders can be created and checked-in as new objects, or checked-out, modified and checked-in by users with the appropriate privileges, but system-wide root level folders can only be created by the system administrator.

Note that content objects may be virtual as well as real. That is, a content object may be a query against a relational or XML database that returns an XML content object. These appear to the end-user as XML content objects, and are handled by Alice exactly as other content objects except for special processing in the connector layer (see section 4.2.2.4) when they are checked-in or out.
4.1.3 Workflow And Versioning Model

Alice is intended for content with long and complex life cycles. It provides two levels of control and tracking, one under the control of the user and one under the control of the system.

At the user level are two housekeeping properties: edition and state. Edition is used to designate, at the user’s discretion, the target publication version of a content object. State is used to identify, at the user’s discretion, a stage in a sequence of workflow processes. The edition value can be set by the user when a content object is checked-in as new, or it can be set by the user as a parameter of the Copy For New Edition operation. The Copy For New Edition retains the inclusions and relationships of the source object in the copied objects, and maintains a “lineage” recording the source from which a content object was copied, which can be extended back through many generations.

The state value can be set by the user when a content object is checked in, or automated processing routines can be implemented to do this under organizational business rules. The default list of state values can be modified or replaced by the system administrator.

At the system level is an object versioning mechanism conforming to WebDAV Delta V. In the simplest terms, each time an object is checked in a new version is created. The previous version is retained, is listed in the object’s Version History, and can be retrieved by the user if needed, but under normal operating circumstances only the most recent sequential version of an object is visible to the user through the collection views.

Figure II: Versioning In Alice

WebDAV Delta V also requires support of “parallel” versions, through the use of its Fork operation, which are ultimately resolved into a single version through the use of its Merge operation. This enables draft documents to be reviewed or edited in parallel. The Delta V fork is implicit in the Alice check out for edit and check out for review operations, which enable the user
to make changes and comments or simply to add comments respectively, as opposed to the check out exclusive. When an object checked out for edit or review is checked in, the fork from the version checked out is created if it does not already exist, and any other documents checked out in the same way will be retained as separate parallel versions when checked in. The Delta V merge is explicit in the Alice merge operation, which merges the changes and comments from a set of parallel versions back into the original from which they were checked out to create a new version. The merged changes and comments are retained as individually identifiable units, along with the original content, not applied irrevocably. The process is shown in Figure II.

Only XML content objects may fork and merge; in Alice, the process is controlled by the change unit management model (see section 4.1.6). An object checked out for edit or review may be converted to RTF or PDF format, which can be used in MS Word or Acrobat so that comments and change units are identifiable. Such converted files may be checked-in, serve as parallel versions, and be merged back into the original XML content.

Arbitrarily many parallel versions, both edit (with changes and comments) and review (with comments only) can exist between a fork and a merge.

Only the most recent sequential version of an object is “visible” in Alice collection views, so when a fork is active the unmerged parallel versions are not visible, though their presence is indicated on the displayed version in collection views and they are listed in the version history of the object. It is therefore the most recent sequential version that is selected by the user to be merged, and changes and comments from the pending parallel versions forked from it are merged by a system process to create a new sequential version.

WebDAV Delta V allows forks to be nested, but Alice by default does not enable this capability (i.e. does not allow the user to check-out a parallel version for edit/review) because it can be confusing to the users, and without careful controls can compromise the effectiveness of the change tracking process.

User permissions can be used to control when a fork may be created, by tying the availability of check-out for edit and check-out for review to particular states for defined document sets.

4.1.4 Content Reuse Model

XML content objects can include other content objects, both XML and non-XML. The non-XML objects are generally embedded graphics of various formats, and can be included wherever the content model of the including document allows a graphic. XML content objects can be included wherever their “root” element is the same as a “container” element allowed by the content model of the including object.

Inclusion is performed during editing using the i4i x4o editor, or any XML editor that can retrieve content objects from Alice, by selecting from a collection browse view the object to be included. The content of the included object is copied into the including object but is “write protected”. This means that objects intended to be reused must be authored and updated as stand-alone instances. When the including object is checked-in it is not decomposed into its component parts, but the inclusion relationship is recorded in the metadata repository. When an object with inclusions is checked-out, if there is in the repository a more recent sequential version of any included object than the one already present in the including object, then that more recent version replaces the one already present.

While Alice is aware of container elements that represent re-usable content objects it does not require that every instance of such an element be saved as a separate reusable object. For example, if a user creates a document from scratch and populates a reusable container element by hand, or by copying and pasting from another object (including an already existing reusable instance), it is not separated on check-in and is not recorded as an independent reusable object.
4.1.5 Cross Reference Model

Alice records the existence of cross reference relationships from an XML content object to another content object, by identifying cross reference links as part of the metadata extraction process and recording them as relationships in the metadata repository. The user can display a list of the objects referenced by, and of the objects referencing, a given content object.

It is important to recognize that Alice does not perform complete link management, in that it does not record the full path to the internal location of a reference (i.e. it does not identify the reference anchor within an object). If a referenced content object is modified so that the target of the anchor is not there when it is checked in, the system will not notice. Ensuring that all cross reference links are resolved is an important aspect of a final production process, particularly when packaging a set of documents in electronic form. However, it is our experience that attempts to enforce link resolution during the content development process is a frequent cause of user frustration. Most print or electronic production packagers have their own link resolution checkers built in, and additional link registration and capabilities can be easily added at the Alice metadata layer if considered necessary to be applied at any point in the business process.

4.1.6 Change Unit Management Model

Change tracking is automatically enforced by i4i’s x4o editor when it is used on an XML instance checked-out for edit, and enables only the addition of comments when it is used on one checked-out for review. Appropriately privileged users can also turn on change tracking on an instance for use when checked-out exclusively.

All changes to a document in change tracking mode are stored as individual insertions or deletions, with the date/time and the identity of the user who made them. All comments are similarly identified.

4.1.6.1 Sequential Changes

If an XML instance with change unit tracking turned on is successively checked-out exclusively to a series of users, changes and comments made by each preceding user are visible to the current user but cannot be deleted, invisibly altered or “accepted” (applied to the original content) in the x4o editor. The current user can make changes against original content or against existing changes. Changes to changes appear as inserts/deletes on the content of the change, essentially fragmenting the previous change into several change units, still bearing the information about their origin, plus the new change units.

An appropriately privileged user (normally the “owner”) can checkout exclusively an instance with changes and comments, and accept or reject them, and delete or retain comments. When the document is checked-in the new version with the consolidated changes becomes the visible representative.

4.1.6.2 Parallel Changes

Parallel changes are enabled through forks in the versioning, as described in section 4.1.3. When operating in parallel, individual editors and reviewers see the original content only, because they check out the same visible version in parallel. Their comments and changes are tracked individually. When they check-in a document with changes and/or comments it becomes one of the set of parallel versions waiting merging.

An appropriately privileged user (normally the “owner”) can perform a merge on the visible, original instance of a content object with pending parallel versions. This invokes a system process that reads the original and each pending parallel version and copies the comments and
changes into the original to create a new version with all of the comments and changes present as individually identified change units. When change units from different parallel versions overlap against the original they are fragmented into separate change units, and saved in the original exactly as changes to changes are saved in a sequential change process.

The new version with the aggregated comments and changes can then be checked-out exclusively by an appropriately privileged user who can accept or reject the changes, and delete or retain comments. When that instance is checked-in the new version with the consolidated accepted changes and comments becomes the visible representative.

4.1.7 Security Model

The security model used for Alice is itself based first on the security model of the underlying CMS(s), and second on metadata. The general model is illustrated in Figure III.

![Figure III: Security Model](image)

Alice is configured by default to expect a single log-in to both the metadata server and the underlying CMS (or CMS’s if multiple backend repositories are used).

The default security model assumes the conventional CMS form of Individual Users assigned to one or more User Groups, with a list of permissions, one for each CMS operation, attached to each group. Each Alice-managed object is assigned to precisely one group.

When Alice is accessing the CMS in any way, the CMS security governs. For example, when executing the query associated with a folder Alice must retrieve the metadata “packages” representing the objects in the set from the CMS; these packages have the same access rights as the documents they represent, so the user making the query is presented with only those objects on which he/she has at least Read permission. Similar restrictions are applied to check-ins.
Atop this mechanism Alice adds a further control layer based on metadata. Document sets for security purposes are defined by independent folders. Associated with each of these sets is a set of metadata-based control properties, each of which is in turn related to a single user group in the CMS security structure. For example, in a pharmaceutical application, the document set might specify all the documents for a particular drug, and the associated metadata-based control set might define an entry for each document state, in turn related to a user group with specific privileges. Thus the user might be allowed to check-in (i.e. update) a document in the “Review” state, but not in the “Approved” state. This provides a straightforward method for relating metadata properties to access rights.

4.1.8 Inter-Working Model

Alice presents a program level interface based on industry standard protocols to facilitate inter-working with other applications and servers. On both the client side and the repository side of the metadata server, Alice presents and uses WebDAV and related SOAP/HTTP protocols, all XML-based, as its interface with the outside world. This has two primary consequences:

(a) It enables other applications to use the Alice metadata in a straightforward way, with a minimum of development effort; and,

(b) It enables Alice to inter-work with a number of back-end content repositories, databases and other back-office applications.

An example of a client-side application is an integration of Alice with MS Project provided by i4i. In this implementation, a VBA application embedded in a project template, supplemented by i4i’s WebDAV client module, enables the user to view resources in Alice and relate them to tasks defined in a project through the Task Management interface shown in Figure IV.

Figure IV: Alice MS Project Integration Task Management Interface
Those tasks in a project that refer to work to be performed on individual documents or sets of documents represented by folders can be linked to the resources in Alice. Periodically, the project manager can run the Update Alice Tasks function through the project, and this will update % Complete, Actual Start and Actual Finish dates in the project based on state information extracted from Alice.

This application uses only the WebDAV PropFind method to extract the information. Alice remains “uninvolved” in the process. This illustrates how simple it is to use Alice information from another client-side application, in a “pull” mode where Alice is a passive participant.

If it is desirable to have Alice inter-work with another application as an active participant, in a “push” mode, this can be achieved through the “back-end”, by adding a module in the CMS Connector sub-layer of the Alice Services Layer (see section 4.2 below) to be triggered when the appropriate change of state or other event happens. The back-end is already set-up to communicate with WebDAV-compliant repositories or to create out-bound SOAP/HTTP requests, so it is easy to pass event information or content objects to flow managers such as MS Biztalk.

Applications which set Alice metadata through the WebDAV PropPatch method, are also possible; however, as with any database or DM/CM system, care must be taken to ensure than basic Alice rules are followed in order not to introduce unexpected side-effects.

One of the most important benefits of this standards-based approach is that it enables an organization to leverage its existing investment in DM/CM, database or workflow software. XML authoring and flexible metadata management functionality can be added “on top of” existing software and processes, which lowers both the cost and the operating impact of implementation.

Since not all DBMS’s or DM/CM packages are XML-aware or fully WebDAV compliant, i4i has provided repository connectors for some of the most widely-used, and custom connectors can be easily developed for others.
4.2 Implementation Architecture

The term implementation architecture refers to the structure and organization of the Alice software and data storage as deployed for use.

Alice software is organized in three distinct layers:
(a) Client layer
(b) Alice Services Layer
(c) Content Repository Layer

All Alice software components communicate using WebDAV over HTTP, so they can be deployed on separate machines to enhance overall performance if required.

Figure V: Alice Implementation Architecture

Legend:
Operations on views: search, open folder, display properties
Operations on objects: check-out/check-in/copy/delete
4.2.1 Client Layer

The client layer is the part of Alice visible to the end user. Two primary clients are available:
(a) standard network-based client integrated into MS Windows Explorer, and,
(b) web-based client integrated into MS Internet Explorer.

Both clients provide the same set of services:
(a) browse collection views
(b) ad hoc search
(c) create, modify and organize collection views (folders)
(d) content object check-out
(e) content object check-in and check-in as new, with integrated metadata editing

All actions available to the user are subject to access controls based on individual user permissions and individual collection/content object parameter values.

All clients communicate with the Alice metadata repository using WebDAV Delta-V over HTTP.

Content objects are checked out to the desktop so they are available for use by various content creation tools.

Client capabilities (a), (d) and (e) have also been built into i4i’s x4o XML editor.

The client software uses local storage on the client PC to save checked-out objects, and to maintain a history from which changes to folder contents can be deduced.

4.2.2 Alice Services Layer

The Alice services layer is itself divided into several distinct components, some of which are optional, and which form “sub-layers” that implement independent functions, as follows:

4.2.2.1 Unit Change Management

The optional Change Manager merges unit changes and comments back into the source XML content object from change-bearing XML, MS Word or PDF content objects undergoing parallel review and update.

4.2.2.2 Complex Document Management

The optional Complex Document Manager processes, on check-in and check-out, XML content objects which include (i.e. re-use) other XML content objects.

On check-out the Complex Document Manager ensures that the latest version of each included content object is being used in the object being checked out.

On check-in the Complex Document Manager updates the metadata representing the inclusion relationships with any objects reused by the object being checked-in.

4.2.2.3 Metadata Management

The required Metadata Manager maintains the metadata repository, handles check-in and check-out functions, performs queries and provides the property information to support collection views.
4.2.2.4 CMS Connectors

Content repository connectors (one required), maintain the mapping between Alice content object ID's and their ID in the content repositories in the next lower layer, and interact with those repositories for check-out and check-in operations.

Any one Alice server can support multiple back-end servers provided that any given content object lives in only one of them. Any WebDAV Delta-V compliant server can be used as a repository, and connectors can be created to support any protocol.

Connectors are available to support:
- Oracle
- Software AG's Tamino
- Documentum
- Microsoft SharePoint
- OpenText's LiveLink
- Windows file servers

Note that some of these CMS's are not WebDAV Delta V compliant; the existing connectors emulate Delta V compliance for those with basic WebDAV services, and full WebDAV Delta V for those without any WebDAV functionality.

In the current version of Alice these connectors are “one-way” in the sense that they act to check-out or check-in content objects from a content repository, but are unaware of any other actions performed on a content object that may pass through the content repository but not through Alice. In a future version, “two-way” connectors will be developed for use with repositories that can route event reporting to another application.

In addition, the connector sub-layer provides additional functionality, including:
(a) virtual content object processing.
(b) extended workflow dispatch.

When asked to check-out a virtual content object, the connection layer recognizes this from the “metadata package”, and executes the query contained in that object against the particular database specified. The query must return a valid XML instance. Virtual content objects are read-only, so a check-in can only update their associated metadata.

On check-in the connector sub-layer can be programmed to recognize special conditions based on the associated metadata and cause the object itself, or a reference to it, plus selected metadata to be passed to another process via the SOAP/HTTP protocol.

4.2.3 Content Repository Layer

Strictly speaking, the content repository layer is not part of Alice per se, but comprises one or more independent content storage servers, which may range from simple file servers to full-scale content management systems. The list of supported repositories is given in section 4.2.2.4.
5. Summary

The objective of this paper is to explain the capsule description of Alice that appears at the beginning. In doing this, we have discussed in detail the design and implementation aspects of Alice, to show:

(a) how its design is based on the nature and use of descriptive metadata, as opposed to a particular type of content production process,

(b) how descriptive metadata can be used to organize and manage complicated collections of content objects over long periods,

(c) how metadata can enhance collaboration through identification and tracking of re-usable content and of content unit changes,

(d) how multi-contextual, placeless views represented by hierarchies of folders both enhance control over the content and enable flexible organization to meet a variety of end-user needs,

(e) how its implementation is based on modular software components and industry-standard service object interfaces so that it can inter-work easily with a variety of existing software applications and repositories, and

(f) how it can be configured and deployed to achieve the maximum effect on content management with a minimum investment in new software and training investment.

In short, Alice is a unique metadata-driven collaborative content management product: a flexible, scalable software suite, designed to facilitate the collaborative, iterative development and management of high-value, knowledge-intensive content through the use of metadata extracted from XML objects or associated with non-XML objects both to represent structured collections of documents by means of multi-contextual views and to track reusable content components and content change units, and implemented as a set of modular services that plug into a multi-tier architecture to connect through standard protocols to an organization’s existing workflow, data and content management applications.
APPENDIX I: Glossary

(I) 1. Metadata
Metadata is data about data, that is, data that names and ascribes properties to other data. The term has two distinct but related uses, as a search of the term in Google will show:

(a) In its older usage, “metadata” refers to the definitions of the names, data types and value domains of atomic data elements, or simple concatenations of such elements, of the sort found in computer programs or in the schema defining the tables and columns of a relational database. This kind of metadata can best be thought of as prescriptive metadata.

(b) In its newer usage, “metadata” refers to named properties ascribed to complex data objects, otherwise known as content objects, usually but not necessarily documents. The values of these properties are either extracted from the content or assigned by authors or editors. The method is similar in concept, and in some cases in implementation, to the properties traditionally used to manage library collections: author, title, subject, etc. This kind of metadata can best be thought of as descriptive metadata.

Alice manages descriptive metadata. However, unlike traditional library practice, Alice is not restricted to a fixed set of metadata elements (i.e. named properties), and each content object may have a subset of the total repertoire of properties known to the system.

(I) 2. High-Value, Knowledge-Intensive Content
The term “high-value, knowledge-intensive content” designates sets of related documents that carry significant intellectual property and/or information binding their sponsors under the law or otherwise exposing them to legal liability, where the set of documents is developed and maintained over long periods of time by a collaborative effort involving a number of professional or otherwise highly skilled subject matter experts (SME’s).

(I) 3. Content Object
A content object is any content-bearing file in any format that also carries attached metadata comprising named properties known to an Alice server. Elements within XML content objects may also carry metadata that can be extracted by XSLT scripts.

(I) 4. Virtual Content Object
A virtual content object is represented in Alice by an XML instance that carries metadata and appears as an ordinary content object in views, but that contains a query against a relational or XML database that returns an XML instance to be used in its place when the content object is retrieved in Alice.

(I) 5. Dynamic Trove (aka Folder)
A dynamic trove is an Alice-specific XML instance carrying metadata and a query against the metadata repository. It appears as a folder in views, and is generally referred to as a folder. It represents and retrieves other troves and/or content objects that satisfy the properties and values expressed in its query.

(I) 6. Collection
A collection in this context refers to the set of content objects represented and retrieved either through a single folder, or as the “leaves” of a hierarchy (tree) of folders.
7. **Classification**

The term “classification” indicates the use of a controlled vocabulary for the values that may be used for one or more properties. The simplest form of classification is a simple list of the permissible values for a given property. Progressively more complex forms include:

(a) a thesaurus, which is a hierarchy of values organized into broader and narrower terms (possibly including weak cross linkages in the form of see/see-also links between terms), that may be used as values of a given property;

(b) a taxonomy, which is a hierarchy of sets of values, organized so that each set is labeled with the name of a property and each value in that set may designate another labeled set of values.

For example, the scientific names of drugs manufactured by a given company might be defined by a simple list, while the names for clinical indications might be defined by a thesaurus, and the more arbitrary relationship between a given drug and its dosage forms as sold in given markets might be defined by a taxonomy:

{drug} a
{dosage_form} x {market} m1,m2;
{dosage_form} b {market} m1, m3;

The current release of Alice supports only the simple list form, but thesaurus and taxonomy tools are planned for future releases.

8. **Multi-contextual Views**

The term “multi-contextual view” refers to the ability to have a single content object show up in different collections, as defined by its properties and the queries inherent in those folders. The objects themselves are sometimes said to be “placeless” because their location in these organizing structures is not tied in any way to storage location.

9. **Reusable Component**

A content object that may be used by another content object by an inclusion reference. This means that the reusable component can be identified in the body of another content object, and when that “host” content object is displayed on the screen or printed the “referenced” object appears in-line in place of the reference. We are all familiar with this practice with respect to graphics included in a text document, but for the most part text re-use is effected by cut-and-paste operations. In Alice, reusable components are either “atomic” graphic objects, or XML content objects whose root element enables them to be inserted at permitted locations in other XML content objects.

10. **Content Unit Change**

Any contiguous deletion or insertion within a document.
APPENDIX II: Alice Functionality

(II) 1. Navigation Through The Explorer Interface

The Alice metadata server is integrated into Explorer, where it appears as a folder including the metadata server (or servers) and the local Alice store on the client PC.

When a folder is opened its contents appear in the list view on the right, showing the object/file type icon, display name and other selected metadata values. The local storage on the PC is used to modify the display by applying the user’s history information from the last time the user opened that folder, such that:
(a) Objects that have been added are displayed with a “gleaming” icon
(b) Objects that have been deleted are displayed with a “shadow” icon.

Note that in this case “added” and “deleted” refers to changes in metadata that cause the object to be included or not included in the set defined by the folder. The duration of these effects are calendar time dependant, and can be set as part of system configuration. The default settings are 5 days for addition and 10 days for deletion.

Local storage shows only the objects currently checked-out to that user.

Navigation is conventional, Explorer-style, with several additional functions available to the user:
(a) Choose columns (metadata values to appear in the detail display)
(b) Alice Search (ad hoc searches on metadata)
(c) Invoke Folder Wizard
(d) Operate On An Object

(II) 1.1 Choose Columns

This option is available in the fly-out from a right-click anywhere on the contents list view. When selected the user is presented with a form that allows them first to select from the named sets of metadata known to the server; when a set is selected all of its properties appear in a check-box pane. The user indicates by selecting/deselecting individual properties whether or not that property is to be included as a column in the detail display.

(II) 1.2 Alice Search

This option is invoked from the Views - Explorer Bar list. It places in the left pane a form with a three column list view that enables the user to perform ad hoc searches using metadata. Its form and behavior is the same as the selection criteria for folders described in section (II) 3.2.2 below.

(II) 1.3 Folder Wizard

This option is available under New in the fly-out from a right-click anywhere on the contents list view and invokes the Folder Wizard, the operation of which is described in section (II) 3.2, Creating Folders.

(II) 1.4 Operate On An Object

Permissible operations on an object are displayed in the fly-out from a right-click on the object. If the user’s privileges or the objects state do not allow the operation on the selected object, the command is grayed-out.
(II) 2. Operations On Content Objects

(II) 2.1 Properties

Displays a tabbed form listing all the metadata held about the object. The tab categories are:
(a) Display properties (those properties that are part of the system’s configured default display)
(b) All properties
(c) Inclusions (if any)
(d) Included In (if any)
(e) References (if any)
(f) Referenced By (if any)
(g) Lineage (history of previous Editions through Copy To New Edition)

Properties are displayed in a two-column list view, with the property name on the left and the
value on the right. If a document is checked-out exclusive to the user, he/she can add or modify
property values. To add a new property the user selects the property name cell in the first empty
row and is presented with a drop down listing the properties available. Required properties and
multi-valued properties are indicated specifically indicated. The user can enter multiple values
where allowed in the property value cell by separating the values with semicolons. Properties
with a controlled vocabulary will present a drop-down in the value cell for selection.

Relationship metadata cannot be modified directly by the user.

(II) 2.2 History

Displays the various versions of this object and the date of their creation. Forks and the versions
arising from them are indented; if no successor version has been created through a merge
operation, they will be the last in the list.

(II) 2.3 Check-Out

The user has three possible check-out options:
(a) exclusive
(b) for edit
(c) for review

They will be grayed-out if the user does not have the appropriate privileges, or if the document is
not in an appropriate state for edit/review.

In all cases a copy of the content object is copied on to the desk top.

(II) 2.3.1 Check-Out Exclusive

This state is recorded in the server and in the CMS. The object cannot be checked out exclusive
to any other user until it has been checked-in or the check-out cancelled.

(II) 2.3.2 For Edit

Applicable only to XML content objects (or RTF or PDF content objects converted directly from an
XML content object) in the appropriate state (by default, “Review”). When opened in the x4o
editor, change unit tracking is enforced.

Objects checked out for review are not recorded as checked out in the metadata server or the
CMS, but carry intrinsic identification that is used by the check-in process to manage forks and
parallel versions.
(II) 2.3.3 For Review

Applicable only to XML content objects (or RTF or PDF content objects converted directly from an XML content object) in the appropriate state (by default, “Review”). When opened in the x4o editor, the user is limited to adding comments. Objects checked out for review are not recorded as checked out in the metadata server or the CMS, but carry intrinsic identification that is used by the check-in process to manage forks and parallel versions.

(II) 2.4 Check-In

Available only if the object is checked-out and stored on the user’s desktop. If the object was checked-out exclusive, the user will be presented with the document properties form to change or add metadata (other than embedded metadata in an XML content object).

If the object being checked-in was checked-out for edit or review, and if the source version (i.e. the one from which it was checked-out) is still the most recent sequential version, the process will create a version fork if one does not exist and a parallel version, or just a parallel version if the fork is already active. If the source version has been checked-out exclusive, merged, deleted, or is otherwise no longer the most recent sequential version, the operation will fail and the user will be notified of the reason.

(II) 2.5 Cancel Check-Out

Available only if the object was checked-out to the user’s desktop, or to the system administrator. The outstanding check-out exclusive is cancelled in the metadata server and the CMS.

(II) 2.6 Merge

Available only on an XML content object in the appropriate state (not checked-out exclusive) and with an active fork (implying one or more outstanding parallel edit/review versions), then the merge process is invoked to merge the comments and individual change units into the XML source version to create a new version containing the aggregation of comments and changes for acceptance or rejection by the author.

(II) 2.7 Copy For New Edition

Displays the current edition property and asks the user to enter a new edition property value, and makes a new content object in the CMS and represented in the metadata server with all the other properties of the source content object.

(II) 2.8 Delete

This operation sets the Deleted property value to True. The object appears as a deleted object (independently of individual user’s local history information, and until it is actually deleted) in all of the folders where it would normally appear. A deleted object cannot be checked out, or have edit/review versions checked in, but can be “undeleted” by the system administrator, or physically deleted by the system administrator or by server-based housekeeping processes invoked on sets of objects when they reach a predetermined state.

(II) 2.9 Check In New

This operation is initiated from an object selected in the file system (on the desktop or network) by selecting Send To – Alice Server. The user will be presented with the same properties form as in an ordinary check-in.
(II) 3. Operations On Folders

(II) 3.1 Properties

Displays the properties of the folder under five tabs:
(a) Display Properties
(b) All Properties
(c) Selection Criteria
(d) Specific Inclusions
(e) Specific Exclusions

(II) 3.2 Create Folder

The create folder wizard leads the user through a series of steps as follows:
(a) Name the folder
(b) Set the folder properties
(c) Define the selection criteria
(d) Define any specific inclusions
(e) Define any specific exclusions

(II) 3.2.1 Set Folder Properties

The folder properties are presented in a list view with two columns, one for property name and one for value. In each row of the name column, a drop-down list of available metadata elements is presented when the cell is selected. When the user selects an item from the drop-down list, s/he can then enter the value for that property in the corresponding row of the adjacent column. Properties with a controlled vocabulary will present a drop-down in the value cell for selection.

(II) 3.2.2 Set Selection Criteria

The selection criteria are presented in a list view with three columns, one for property name, one for relation, and one for value. In each row of the name column, a drop-down list of available metadata elements is presented when the cell is selected. When the user selects an item from the drop-down list, s/he can then select a relation to be applied from the drop down list in the adjacent cell in the relation column. The relations supported are:
(a) Equals (default)
(b) Not equals
(c) Greater than
(d) Less than

The user can then enter a value in the adjacent cell in the value column. Properties with a controlled vocabulary present a drop-down in that cell for selection.

On any one property the user may enter a list of values, separated by semicolons, to indicate a logical OR in the selection.

The query performs a logical AND across all the properties used in the selection.

A given property name may be used more than once in the list of properties. Thus selections on a range of values may be made using the same property name twice, once with a value under a “greater than” relation and once with another value under a “less than” relation.

The user may also restrict the objects retrieved to either folders or content objects by using the Type property with the appropriate value.
If needed, the SQL form of the query can be edited by appropriately privileged users to create special sets not selectable through the selection criteria interface.

(II) 3.2.3 Set Specific Inclusions Or Exclusions

These allow the user to identify, either by display name, object ID or point-and-click, objects that are to be included in, or excluded from, the retrieved set regardless of their properties.

(II) 3.3 Check-Out

Users may check-out folders in order to modify their properties, selection criteria or specific inclusions or exclusions. Only Exclusive check-out is possible on a folder; parallel versions are not supported for folders.

(II) 3.4 Check-in

Available if the folder is checked-out to the user's desktop. The folder is checked in.

(II) 3.5 Delete

This operation sets the Deleted property value to True. The folder appears as a deleted object (independently of individual user's local history information, and until it is actually deleted) in all of the folders where it would normally appear. A deleted folder cannot be checked out, but can be "undeleted" by the system administrator, or physically deleted by the system administrator.