International Color Consortium®

White Paper 25

Level: Advanced

Date: February 2012

Color management implementation classification

Overview

Color management is used and implemented in many ways. As different implementations and specific architectures are proposed it is useful to have a common conceptual framework within which these can be compared. This paper briefly provides a definition of Color Management that can be used in the analysis of different architectural implementations. It then presents a general high-level architecture for Color Management and outlines a continuum for comparing different architectural implementations. In conclusion, different categories of architectural implementations are identified and compared using the presented continuum.

An important point to note is that there is no universal best way to implement color management. Each implementation will have its trade-offs as it achieves its goals related to color management, and the choices involved in these trade-offs are often different for different use cases. This paper is intended to facilitate analysis and comparison of architectural implementations, and as such does not focus on specific workflows.

Color Management

The Glossary of terms in ICC White Paper #5 defines the term “color management” as used in digital imaging as follows:

> **color management (digital imaging)**
>
> *communication* of the associated *data* required for unambiguous interpretation of color content data, and *application* of color data conversions, as required, to produce the *intended* reproductions. [ICC.1]
>
> NOTE 1 Color content may consist of text, line art, graphics, and pictorial images, in raster or vector form, all of which may be color managed.
>
> NOTE 2 Color management considers the characteristics of input and output devices in determining color data conversions for these devices.
>
> *Italics added for emphasis*

Implementations of Color Management involve how four important parts from this definition are achieved: *communication, data, application, and intended reproductions*.

Architectural Layers of ICC Color Management

Generally, the architecture for current ICC Color Management is implemented in layers as shown in figure 1.

Note: Other architectures may exist but these layers can be thought to exist on a conceptual level.

Generally the higher the level in the architecture, the more proprietary the implementation is considered to be. The lower levels are often considered to be more open.

Note: Even though metadata in the lowest levels can be created using proprietary transform generation implementations, the metadata is typically encoded in standard formats that can be used by more open implementations.

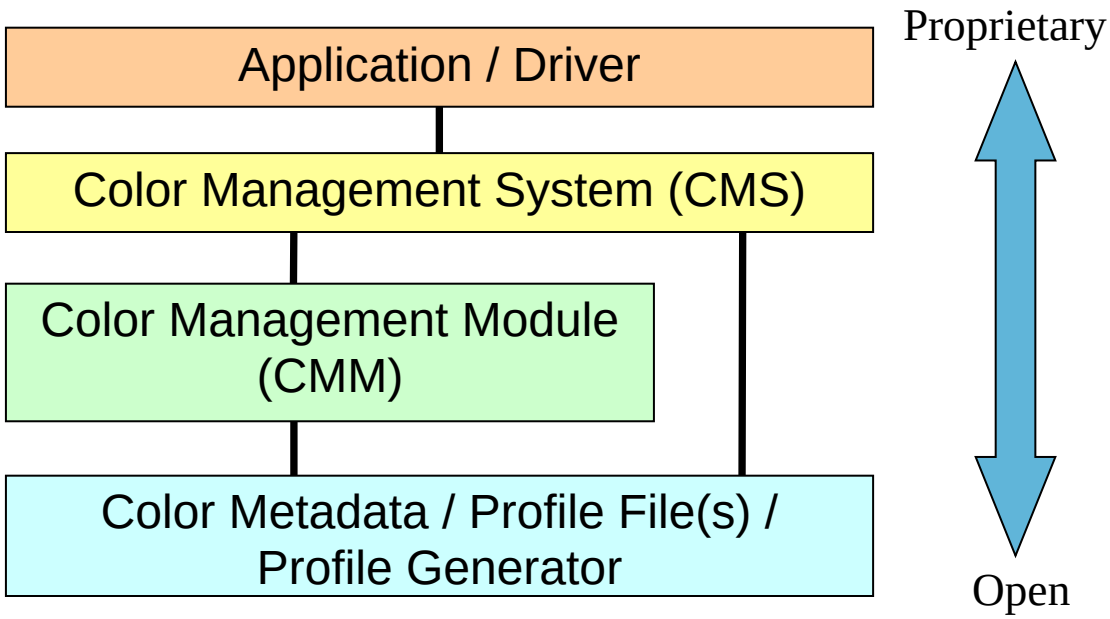

Figure 1 – Color Management Layers

The top layer or Application / Driver layer is the client for Color Management. It ingests source image data and exports destination image data, possibly requesting lower levels to perform color management of the image data. This layer may gather and/or process color metadata, or may defer some or all gathering and processing to lower levels.

The Color Management System layer processes color metadata, not color data. It obtains color metadata from the application level, from devices or their drivers,

or from user input. The CMS determines the class of color metadata (such as OpenEXR CTL or ICC profiles), which in turn determines the class of CMM to use. In some cases, the color metadata can prescribe a preference for a CMM within its class.

The Color Management Module layer assembles and executes color transforms. The CMM takes direction from upper and lower layers in addition to providing its own operational logic to perform transformations of the color data. Some CMMs can be used with only one class of color metadata, while other CMMs can be used with multiple classes. On some systems, multiple CMMs may be available for ICC profiles.

The lowest layer is the Color Metadata / Profile layer that provides information used to assemble and execute color transforms in the CMM layer. Color Metadata may describe the characteristics of a color data source or destination, which are often related to physical or virtual reference devices/media. Color metadata may also provide color transforms, and/or instructions for the application of color transforms. Many metadata formats are in current use. Some metadata have variable digital representations, such as measurement data or transform data, while other metadata are in the form of explicit or implied references to specifications (e.g. sRGB and the Digital Cinema X'Y'Z'). In the ICC workflow, the metadata is encoded as an ICC profile constructed according to the ICC profile format specification. Often, a Metadata/Profile Generator application is used to create the metadata/profile. Metadata/Profile Generators can use their own operational logic in the process of generating the transforms encoded in the metadata/profile.

Note: Since applications and/or drivers make all color management requests through the Color Management System layer, the term "color management system" often refers to the aggregate of the lower layers, instead of the top layer only. The context determines whether a single level or the aggregate is being referred to.

The CMM/Metadata Implementation Continuum

Most of the color transforms in a color management implementation are defined in the bottom two layers. The implementation possibilities can be considered as a continuum of runtime behavior with possible implementations of CMM and metadata layers at the extremes of each end. This can be seen in Figure 2.

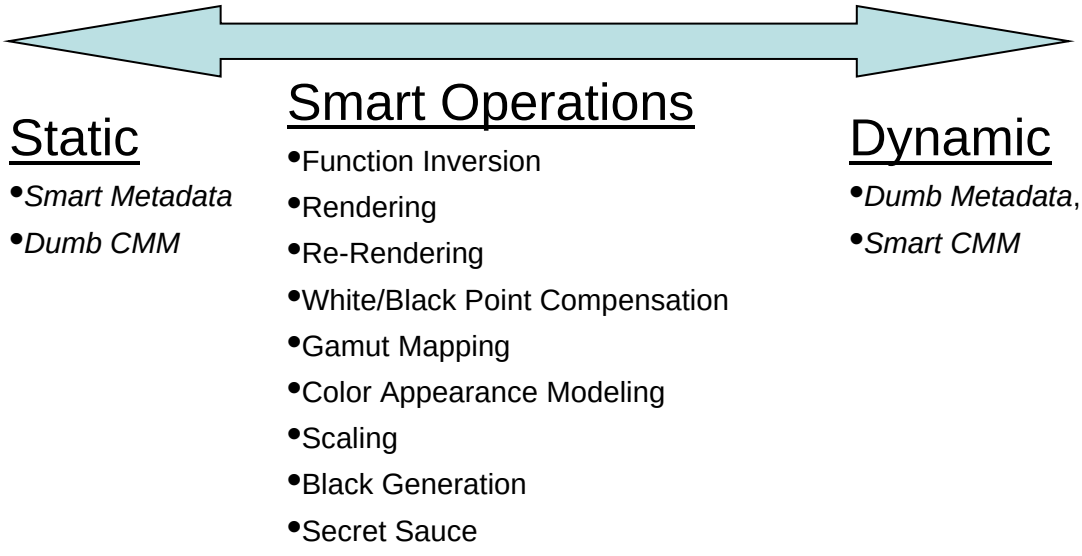

Figure 2 – CMM/Metadata Implementation Continuum

If **the transform operations are defined and controlled well in advance of applying the color data transform(s),** for example, when the color metadata defines the operations to perform, the implementation is classified as Static. In this situation, the color metadata provides the complete operational logic, and the CMM needs no additional logic to determine what transforms to apply to the color data. This generally means that the operational logic in the transformations is assembled and used at the time the color metadata is created.

If **the transform operations are mostly defined by the CMM, user settings, and/or image data,** and not in the color metadata specification, the implementation is classified as Dynamic. In this situation, the operational logic is provided by the CMM. Color metadata, if used at all, provides only basic color measurement information. A Dynamic CMM is free to implement any operational logic that it wishes, but this comes at a cost of interoperability and predictability between Dynamic CMMs with different implementations and/or configurations. This generally means that the operational logic in the transformations is assembled and used at the time the transformations are applied

For most dynamic implementations, accurate color characterization (measurement) data needs to be retrievable for the source and destination color data encodings (which may be for specific devices). The ICC.1:2001-04 profile specification (version 4) improved the ICC profile format to ensure that Dynamic CMMs could retrieve accurate color characterization data.

Note: Current basic ICC implementations are not entirely Static. Rendering Intent linking, the XYZ to/from Lab conversion, and the absolute rendering intent operations to adjust the white point for absolute colorimetry represent dynamic runtime behavior required by the ICC profile specification. Additionally, CMMs that perform black point compensation also provide additional Dynamic runtime

behavior. Dynamic behavior is predictable when it is clearly specified in a standard. Thus the Dynamic behavior is required to be available by the implementation and the specifics of when and how to apply the Dynamic behavior is clearly defined.

Overcoming Limitations

With an understanding of the architectural layers of color management and the CMM/Metadata Implementation Continuum, analysis and comparison between different implementations is possible.

A basic ICC color management-implementation, which supports only the transformations implied by the ICC profile specification, is limited to only those transforms that can be encoded in ICC profiles, or those that the CMM must dynamically implement as defined in the ICC profile specification. Additions need to be made somewhere in the color management layers to go beyond these limitations.

Changes made in lower layers of the architecture are easier to standardize for organizations like the ICC. Though it can be done at higher levels in the architecture, generally the CMM is modified, and possibly the color metadata. Different implementation approaches therefore correspond to movement in the CMM/Metadata implementation continuum.

In a Dynamic CMM implementation the sequence control is centralized in the CMM, but to be open and cross platform, agreement on sequence/control within the CMM is required. In the past, coming to agreement has proven to be difficult. Some reasons include the significant preferential/artistic aspects of cross-media color reproduction, and the estimation of the color appearance of images viewed in different conditions. With such lack of agreement, different CMM implementers have provided additional operational logic to address different use cases, possibly requiring private tags and/or external configurations to go beyond the limitations of basic ICC implementations. However, if private tags are used then they may not be understood by other CMMs. Interoperability between different Dynamic CMMs is therefore limited to the baseline behavior required by the ICC profile specification.

Extending the CMM/Metadata Implementation Continuum

An alternative modification to a CMM would be to define a Pluggable CMM that provides a standardized extendable control architecture using a plug-in method to provide the implementation of pre-defined steps. Some of these defined steps might provide, for example, device modeling, gamut mapping, or device channel separation as plug-ins. Default plug-ins can be prescribed for such an implementation, but they can be replaced to meet specific needs. This allows for

secret sauce to be implemented in proprietary plug-ins while still providing for some level of baseline openness.

(Note: A Plug-in can be thought of as additional form of operational metadata that provides the implementation of transform/control logic not provided directly by the CMM).

In providing plug-ins to a standardized CMM, movement on the CMM/Metadata implementation continuum could be considered to be in a different dimension than the Static versus Dynamic runtime behavior. An additional Fixed versus Programmable dimension to the CMM/Metadata Implementation Continuum allows comparisons to be made between different levels of plug-in capability of Pluggable CMMs. A revised continuum, which replaces that of Figure 2, is shown in Figure 3.

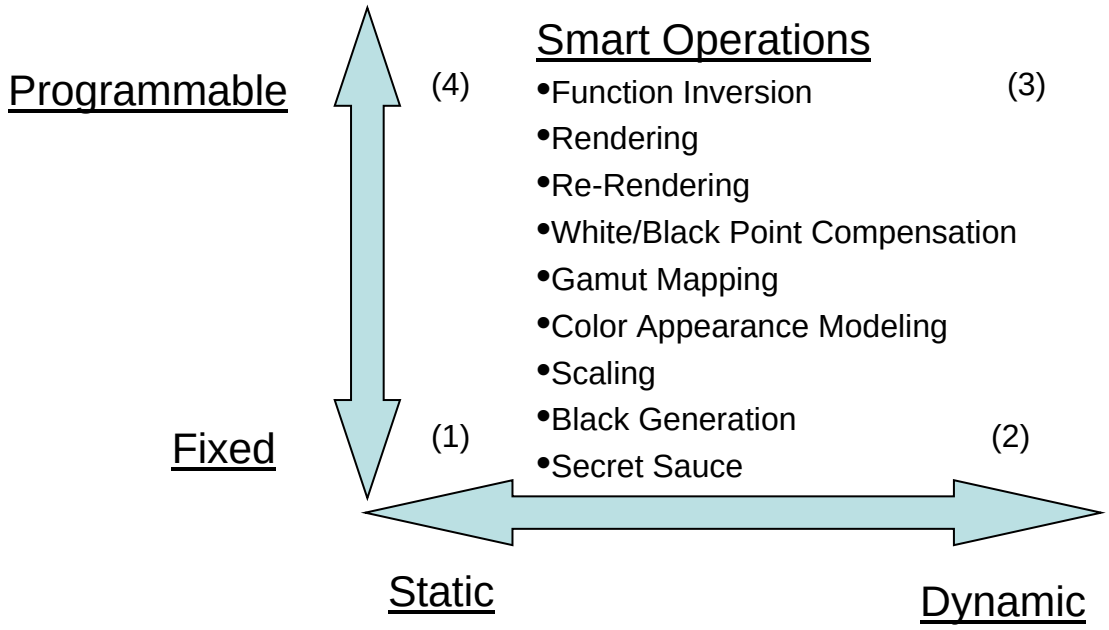

Figure 3 – Revised CMM/Metadata implementation continuum

One serious concern with a Pluggable CMM implementation would be that the unambiguous communication of color requires that all CMMs in a complete workflow are configured the same when asked do the exact same task. Do they all have the same plug-ins installed? Is the same essential architecture implemented on different platforms? Are plug-ins implemented (the same) for every platform? Are the plug-ins all configured the same? With Pluggable CMM implementations, interoperability is a significant concern.

With this revised version of the CMM/Metadata implementation continuum, a Static Programmable implementation is open for consideration. If the runtime behavior is to be Static, then the Programmability needs to be fully controlled by the color metadata. Both the color metadata and the CMM need to be extended to provide more operational options, which are controlled by the color metadata. In this case, the operational logic of both the CMM and the color metadata are extended, but the runtime behavior remains Static.

With a Static Programmable implementation, greater control and flexibility are possible in an open fashion with the CMM understanding little about what is going on. The additional control is open, as it is added to the color metadata.

A Static Programmable CMM can be thought of as a general purpose Color Transform Virtual Machine (VM) which can easily be ported to different platforms. All that is needed is a specification of the basic building blocks of the VM, and the color metadata can then provide the sequencing to implement various workflows. A Static Programmable CMM doesn't necessarily need to understand what the sequence of operations defined in the color metadata is trying to accomplish. Because of this a Static Programmable CMM can be considered to be a more capable Static CMM.

Placing operational sequence control in the color metadata allows for unambiguous *communication* of both *data* and *application* to get *intended* results. For some vendors, the openness may be seen as a weakness – the sequence of operations is openly defined, and any secret sauce is potentially less hidden. However, the increased openness improves the unambiguous communication of color.

Note: ICC.1:2010 includes of a set of optional tags that allow for the implementation of a Static Programmable CMM. (See White Paper 28 – Introducing the mutiProcessingElement Tag Type).

Review and Comparisons

For comparison purposes the four corners of the CMM/Metadata implementation continuum of Figure 3 are now presented with a brief description and general advantages or observations along with disadvantages or concerns.

Note 1: It should be recognized that an advantage to one person might be considered as a disadvantage to another person (and vice-versa).

Note 2: These represent extremes of the implementation continuum and hybrid approaches will combine features with associated tradeoffs.

1. Static Fixed

The operational logic of the transforms involved are placed in fixed sequence in the color metadata. The CMM is responsible for applying the transform steps with limited conversion between transforms.

Advantages/Observations:

- Most of what needs to be specified is in the metadata specification.
- CMM specification not as necessary
- Easy to make open and cross platform
- Predictability fairly easy to achieve between different implementations
- Proprietary know how is encapsulated/hidden in metadata

Disadvantages/Concerns:

- Limited to transforms options provided in the specification
- Little dynamic runtime behavior is implied
- If knowledge of both source and destination is to be used then it is needed at the time the metadata is created.
    - Knowledge of an intermediary can be used if knowledge of either the source or destination is not known.
    - Use of an intermediary requires that it is well specified and used consistently by different implementations
    - Use of an intermediary is not the same as knowing both the source and destination
- Limited to features provided in the specification

2. Dynamic Fixed
   All operational logic of the transform is placed in the CMM. The color metadata only contains characterization/measurement data. Transforms are calculated dynamically at runtime.

   Advantages/Observations:

   - Proprietary color management requirements may be implemented by proprietary CMMs using standard color metadata (Note: Usually no secret sauce is in the color metadata).
   - The CMM may provide an interface for end-user control of results.
   - Dynamic transform generation allows for transforms to be created based on knowledge of data from source and destination as well as image.
       - If knowledge of both source and destination is used then it is not needed until the time the dynamic transformation is generated.
   - Flexibility in metadata/profile connection.

   Disadvantages/Concerns:

- An Open solution requires an agreed upon CMM specification with all operational and transform logic being clearly defined and specified.
    - In practice, solutions are usually proprietary for the reasons noted previously, and Intellectual Property issues come to bear.
- If fixed operational and transform transorm logic is specified, the specification needs to be changed to do things differently
- Difficult to standardize or to get implemented the same on many platforms.
- Predictability between implementations will be difficult due to differences in each implementation and how they are configured based upon the opportunity for end-user control.

3. Dynamic Programmable
   The CMM supports a sequence of operations that can be customized using a plug-in architecture. The sequence can be scripted or standardized. The color metadata contains characterization/measurement data. Operational metatada can also potentially be used determine the sequence of operations and plug-ins to be used.

   Advantages/Observations:
   - Greatest flexibility. - Any color management implementation is possible.
   - Dynamic transform generation allows for custom transforms to be created based on knowledge of data from source and destination as well as image
   - Pre-determined transforms can be provided as plug-ins.
   - If knowledge of both source and destination is used then it is not needed until the time that dynamic transformation is generated
   - Depending on implementation there can be flexibility in metadata/profile connection
   - Proprietary know how is placed in plug-ins.
   - Alternative ways of doing things can be encapsulated in plug-ins
   - Plug-ins can provide interfaces for end-user control of results.

   Disadvantages/Concerns:
   - Open solution requires an agreed upon CMM specification with all transforms clearly defined and specified.
   - Cross platform difficulties - plug-ins (in addition to CMM implementations) should be made available for multiple platforms.
   - Behavior for default plug-ins needs to be specified and implemented on all platforms for predictability mode to be ensured.
   - Workflows crossing multiple systems require that they all support the same plug-in capabilities (where needed) and are configured the same (where needed) based upon the opportunity for end-user control

- Predictability between implementations will be difficult due to differences in each implementation and how they are configured based upon the potential for end-user control.

4. Static Programmable
   The CMM acts as a Color Transform Virtual Machine. Fixed operations are defined by the metadata specification and implemented in a flexible manner by the CMM. The color metadata provides an arbitrary sequence of operations to be interpreted and executed by the CMM. The CMM doesn't interpret meaning between operations.

   Advantages/Observations:
   - Most of what needs to be specified is in the specification
   - New Workflows and behaviors can be implemented without changes to the CMM.
   - Easy to make open and cross platform
   - Flexibility in metadata/profile connection is possible if the options are in the specification
   - Predictability fairly easy to achieve between different implementations
   - Proprietary know how is encapsulated/hidden in metadata

   Disadvantages/Concerns:
   - Repertoire of operations place limitations of programmability
   - Proprietary know how can become more exposed
   - CMM specification is more of an issue than Static Fixed
   - Little dynamic runtime behavior is implied
   - If knowledge of both source and destination is used then it is needed at the time metadata or a profile is created rather than when the metadata/profile is used.
     - Knowledge of an intermediary can be used if knowledge of either the source or destination is not known.
     - Use of an intermediary requires that it is well specified and used consistently by different implementations
     - Use of an intermediary is not the same as knowing both the source and destination
   - Can the programmable behavior of metadata/profiles invalidate capabilities of Dynamic CMMs that assume fixed transform behavior?