# On Decision Machines for Media Transformation

Norwegian Computing Center
APPLIED RESEARCH AND DEVELOPMENT
Norwegian Computing Center / Applied Research and Development

**NOTAT/NOTE**

**IMEDIA/08/02**

Peter Holmes
Knut Holmqvist

Oslo
August 2002

|  |  |
|---|---|
| **Dato**/Date: | August |
| **År**/Year: | 2002 |
| **Notat nr/** Note no: | IMEDIA/08/02 |

**Tittel**/Title:

# On Decision Machines for Media Transformation

**Forfatter**/Authors:

Peter Holmes, Knut Holmqvist

## **Sammendrag**/Abstract:

Abstract

This document is meant to serve as an interim report of one of the channel S subprojects this year. This subproject specifically addresses a problem area which goes under several different names, including: media (or content) transformation, media adaptation, filtering and/or distillation.

This document offers an initial orientation to the problem area, as well as a more extensive review of various dimensions and issues specifically relevant to the area of media transformation. It concludes with a summary of the future work planned within the subproject in 2002.

|  |  |
|---|---|
| **Emneord**/Keywords: | decision machines, media (or content) transformation, media adaptation, media filtering, media distillation |
| **Tilgjengelighet**/Availability: | Open |
| **Prosjektnr.**/Project no.: | Multimedia Multichannel Production (channel S); 11010 |
| **Satsningsfelt**/Research field: | multimedia production |
| **Antall sider**/No of pages: | 17 |

# On Decision Machines for Media Transformation

Peter Holmes
Knut Holmqvist

Norsk Regnesentral
August 2002

# Table of Contents

# 1. Introduction

## 1.1 Purpose

This document is meant to serve as an interim report of one of the channel S [1] subprojects this year. This subproject is organized under the project area for Multimedia Multichannel Production; specifically, the subproject addresses a problem area which goes under several different names, including: media (or content) transformation, media adaptation, filtering and/or distillation.

*The value in addressing this problem area arises from the need for media producers and providers to control the overall volume, variety and complexity of the content they create, administrate and deliver.* In a “perfected” media transformation solution, it would be possible for media producers / providers to store a *single original version* of each media element. The transformation mechanisms would make it possible to adapt any original media element into a form most suitable to the requesting source.

This document offers an initial orientation to the problem area, as well as a more extensive review of various dimensions and issues specifically relevant to the area of media transformation. It concludes with a summary of the future work planned within the rest of the project period.

## 1.2 Background

### 1.2.1 The value chain

A simple value chain for multimedia production will contain the following phases:

- creation (i.e., creation of media elements)
- storage (i.e., storage of media elements in a MM DBMS[1])
- retrieval (i.e., retrieval of media elements from a MM DBMS)
- media element transformation
- media element delivery / streaming

The first phase is related to creating the media files and putting them together into some kind of *presentation*. The result is stored in the MM DBMS for later retrieval. When the presentation is used, some form of transformation might be necessary. A picture might have to be converted from CIELab to CMYK, a DV-video may have to be transcoded to MPEG-1 and so on. Finally the presentation must be delivered to the consumer, on paper, video or electronically.

### 1.2.2 Conserving meaning

As a presentation, media elements may be organized, retrieved and delivered as individual entities. Alternatively, they may be organized into “packages”, and delivered together as a set of more or less semantically-interdependent elements. The degree and kind of such interdependence may be explicitly or implicitly represented, or may instead be a matter which is completely subjective to the author and/or media consumer.

[1] Multimedia Database Management System.

Regardless of the situation, one of the prime objectives of the transformation phase is to strive to conserve the *meaning* of the presentation, to as great an extent as possible. For text, this amounts to the use of transformation operations which do not render the text unreadable (e.g., in size or density), nor renderings which yield text so terse that meaning becomes completely lost. Images, for example, must not become so down-scaled that essential details become obscured; at such a scale, images might as well be discarded.

Another important aspect of a presentation can be layout (e.g., as in advertisements). Here, significant yet sublime layout designs can be utilized to suggest meaning and connote messages which are not immediately obvious through individual examination of each of the elements in the presentation. In multimedia presentations, meaning can be completely dependent upon inter-element timing and synchronization. Disruption of such timing could render a multimedia presentation meaningless.

This kind of problem has come into greater and greater focus over the last decade. For WWW pages, alternative approaches to this problem are summarized in [7]. These are classified as "device-specific authoring", "multiple-device authoring", "client-side navigation" and "automatic re-authoring". In the first approach, it is the media author which pre-specifies transformation. This is accomplished through use of specific page definitions (page templates), each of which is associated with specific information extraction and page-formatting routines.

In the multiple-device authoring approach, document tagging is used to specify well-defined transformations from a single source document to a set of pre-defined document renderings. In contrast, client-side navigation approaches allow users to alter the portion of the WWW document which is visible at any time (e.g., through panning, zooming, scrolling, expanding/collapsing graphic elements, etc.).

Of these various approaches for delivering "specialized" WWW pages, it is only the automatic re-authoring approach which addresses media transformation as *a planning problem*, as we do in this work. In contrast to our effort, the approach described in [7] does not explicitly address temporal media types.

### *1.3 Structure of the document*

The remainder of the document is structured as follows: Chapter 2 introduces the significance of a distinguishing three layers of data representation within a multimedia database management system (MM DBMS). Chapter 3 provides an overview of the media transformation process, along with an extensive review of issues relevant to the design of multimedia production and media transformation facilities.

A concise summary of the paper is provided in chapter 4, followed by a brief description of project work planned for the rest of the project period.

# 2. Multimedia Database Management Systems

Two categories of applications can utilize the services of a MM DBMS:

- applications which present the contents of elements in the database
- applications which enable content editing.

In this paper we will primarily consider the first type of application. Applications which store updated or changed media elements into the database are part of the second category.

We assume that media elements in the MM DBMS can be in one of two logical and explicitly represented states, *final* or *revisable*. Final elements cannot be altered, they may however be the bases of new elements. Presentations that are revisable can still be altered by the second category applications. Elements in a final state can be transformed to another format or size at the request of and for use by applications.

Data in a MM DBMS are represented on several levels. We will base this discussion on a three-level representation. These levels are:

1) Logical level
2) External level
3) Internal level

The logical level is an abstraction of the data in the database, as represented by data models, data types, etc. The external level is the level through which applications interact with the MM DBMS; one example is an API such as SQL. The internal level concerns the MM DBMS-internal representation formats and processing mechanisms. Only the MM DBMS should be concerned with the internal level; no other system element should directly interact with it.

At the logical level, a MM DBMS might have a data type called “photo”, with accompanying methods to deliver RGB- and CMYK-coded versions through the external interface. If the MM DBMS, for whatever reason, chooses to store transformations of a media element (e.g., a CMYK version of an element denoted *the_photo*), this is of no concern to the applications using the MM DBMS. An application asking for a CMYK-coded version of the *the_photo* does not need to know whether the transformation was made on-the-fly or possibly cached by the MM DBMS for future use.

Therefore, *the_photo* is the only externally-known media element, and the only way to get a CMYK version of it is to ask for it through the external interface. If the application stores this CMYK-coded media element into the database, this is considered as a new media element of a different data type.

This three-layered approach can serve to make the database “application neutral”, as discussed by Marder [2]. For a photo database system this approach is practical today, as can be seen e.g. in OPI systems. For temporally-dependent presentations, however, most solutions are not application neutral.

# 3. Issues in Media Transformation

This chapter has two subparts. The first part intends to present an introductory description of the media element transformation process. The second part describes a number of different dimensions and issues relevant to the area of multimedia production, specifically in regard to the task of media transformation.

## *3.1 Media element transformation*

Here, the description of the transformation process intends to be as general as possible. This presentation seeks to describe media element transformation processes which can require *sequences or sets of transformation operations,* in order to convert source media elements into requested target element types and forms. Thus, the process description may appear overly detailed, when only simple, "one-step" transformations are required (such as a file format conversion).

### 3.1.1 Transformation requests and rules

The process is described as follows: A *media element request* is issued by some system entity; typically, this request originates at some client. During processing, the original request may be augmented / modified so as to include additional information useful to the transformation task (e.g., as in [5]).

Here, the media element request is assumed to refer to some *set* of media elements, e.g., a set of elements desired by some requesting source. The request usually contains explicit and/or implicit target constraints (or preferences), for each media element. Such constraints can concern media type, media format, resolution, size, etc.

In order to achieve media transformation, the system includes a set of *transformation rules*. Many of these transformation rules equate to operations which can be invoked upon media elements. Some transformation rules may also exist which are designed to operate upon well-specified collections of media elements, where these elements within a given collection may differ in data type (e.g., [2], [5]).

In this next phase of the process, we shall refer to two distinct stages of media element transformation: these are *transformation planning* and *transformation execution.*

### 3.1.2 Transformation planning

The transformation planning stage concerns the derivation of a plan by which the requested media elements can be retrieved from the MM DBMS and ultimately delivered to the requesting source, in a form which most closely matches the constraints and preferences specified in the request. A complete plan consists of a specification as to:

- which transformation rules should be applied to each media element (or collection of media elements),
- the order in which the rules should be applied and
- upon which local or distributed system entity each rule should be executed.

Transforming a given media element into a form desired by the requesting source can require the utilization of one or more transformation rules. In some cases, the use of *different rules* and/or *sequences of rules* can yield the same transformation result[2]. In complex systems, the space of possible transformation plans can be combinatorially large, and search within that space must be controlled. For this reason, it is necessary for a transformation solution to include some form of "decision machine" [3].

Deriving an "optimal plan" could solely involve minimizing the estimated cost of *plan*

[2] For example, the (rotate $\rightarrow$ blur) sequence yields the same as the (blur $\rightarrow$ rotate) sequence.

*execution.* In practical situations, the goal of optimization is often to minimize some combination of plan execution cost together with the cost of deriving the plan (e.g., the time required to derive it). Search performed by the decision machine can be effectively controlled through use of some form of *evaluation function* [14][3].

Often, an evaluation function consists of the combination of a *cost function* and a *distance* (or *difference*) *function* (the latter is optional); these are described at length in section 3.2.7. During plan derivation (i.e., search), an evaluation function can provide an estimate of the “promise” of any partial transformation plan. Since partially developed plans can be sorted by the evaluation function, this kind of search control makes the job of optimizing plan derivation a matter of minimizing the evaluation function.

Designing the decision machine and evaluation function(s) — as well as the overall transformation approach — is a complex task, and a number of design decisions can be dependent upon factors such as use context, metadata availability and more (see section 3.2).

As an aside, one prominent goal of this work is to investigate the degree to which design of the transformation approach and decision machine can be de-coupled from use context. This is necessary in order to fully achieve application neutrality.

### 3.1.3 Transformation execution

Following the planning stage is the transformation execution stage, where the transformation rules are applied to the media elements, according to the specifications in the transformation plan. This process could involve “shipping” media elements to and from distributed system entities, in order that specialized transformation rules can be applied to those entities.

It should be pointed out that in some well-defined use contexts, it can be useful to interleave the plan derivation and plan execution stages, or to iterate between these stages. That is, it can make sense to derive and execute a transformation plan which is expected to reduce the size of a (set of) media element(s), before deriving a plan by which to convert the elements from one data format to another. This kind of approach implies a need to group transformation rules into sets or classes; this topic is discussed later in section 3.2.8.

## *3.2 Dimensions of the problem area*

A number of issues arise related to media transformation during multimedia production. These are presented and discussed below.

### 3.2.1 Use contexts

The mechanisms required for a given media transformation solution can be influenced by the use context within which the solution is applied. Two sample use contexts are:

- **Content / media creation**: This context concerns work performed with media prior to storage in final state.
- **Content / media delivery**: This MM DBMS context concerns end user and system activities from the time at which (*final*) content is requested by the end user until it is delivered.

[3] Examples of this approach are found in e.g., [7][10][11].

To illustrate how use context affects the media transformation approach adopted, consider the following: In the media creation context, the approach would likely require some form of media element versioning facility (described later below) . For the media delivery context, the media transformation approach adopted would likely include some kind of caching mechanisms. These could be implemented at the server-side (or proxy-side), at the client-side or both.

### 3.2.2 Media element packaging

This problem dimension regards the precise nature of a media element and issues arising when they are packaged together. These issues can apply to logical, external and/or internal element representations. Consider:

- **Media element granularity and nesting**: This concerns the coarseness of the media element, i.e., whether it is atomic and which intra-element structures or properties are defined and accessible. Alternatively, a media element may have the capacity to function as a container for other media elements, in order to create a form of "media-composite". If so, media elements may be able to be nested in several ways: examples could include limitation to a single level of nesting or, alternatively, multiple levels of nesting. In either case, nesting may be restricted to a single parent or, instead, multiple parents may be allowed.

  A Macromedia Flash™ [25] movie can exemplify this kind of complex / nested presentation.  A Flash movie is temporal in nature and, in addition, can contain other temporal elements such as audio, video or other Flash movies.

- **Inter-element structure**: When a media element can function as a container for other media elements (in some greater or lesser way), it may also contain internal information about relationships amongst the media elements it contains. Containers of this kind which significantly *differ* from one another may imply a need to assign them different *data types*, in order that they can be managed in a more efficient way.

- **Temporal synchronization**: When media elements are packaged into a presentation, there is often a need to specify certain kinds of temporal relations and constraints amongst them. These relations can be specified within the top-level container or package or, instead, specified within various subcontainers (when multiple levels of nesting is allowed). Alternatively, these relations can be specified completely *outside* the package and/or subcontainers, through use of metadata associated with individual, yet specific media elements and containers.

- **Layout / presentation**: The use context for media elements may call for the specification of information relevant to layout / presentation on the user's device. Like the issue of temporal synchronization, such specifications could be included within the media elements themselves. Nowadays, however, such specifications are more often included in higher-level media element containers (when nesting is allowed) or, instead, as "external" metadata which is associated with some form of high-level media element container.

- **Representation of inter-element relations / constraints**: Currently, a number of international, industrial and de facto standards have evolved which can serve as the basis for logical representation of  relations and constraints amongst media elements. Among these are XML, SMIL [17] and the MPEG family. For many such standards, corresponding toolsets have been developed, thus easing the burden somewhat when media transformation facilities are created / assembled.

- **Control over media element representation and packaging**: It can easily become the case that one is subject to work with media elements which aren't "homegrown". That is, one might have to work with and manage media elements which are externally represented in a non-standard data format. When the data format is proprietary or closed, there is little one can do with regard to media transformation — often, transformation becomes an "all or nothing" decision.

  With non-standard yet *open* media element representations, a need for specialized transformation facilities arises; these may exist and be openly available, or available yet subject to license. With open solutions, one can of course build such facilities oneself.

It should be clear that the options for media element packaging significantly impact the transformation approach. Multi-parent nesting can require mechanisms for detecting cycles and eventual redundant processing tasks when planning transformations. Differing kinds of media element containers may require new data types and, in turn, specific processing facilities for transformation actions.

The manner in which media elements — as well as temporal- and layout-related constraints — are represented and specified directly impacts the degree to which specialized (e.g., data type-specific) processing mechanisms are required.

### 3.2.3 Data types

This problem dimension concerns the range and kinds of data types (or media element types) to be managed during transformation, as well as the way in which they are organized.

- **Degree of specialization**: In a MM DBMS, there is some set of system-standard data types. Some media element types can be directly represented using the system-standard data types. For other media element types, this is not always the case (e.g., media elements which can contain other media elements). Here, it may become necessary or useful to define specialized data models for efficient representation. Thereafter, new data types can be defined and assigned, and specialized routines / methods can be developed for high-level management and processing of these new data types.
- **Organization**: The manner in which data types are organized plays a role in regard to the degree to which processing software can be made generic. It might be possible to organize various, specialized data types according to an object-oriented data type hierarchy. In this way, object-oriented techniques could be used to declare and define methods at various levels within the hierarchy — methods which can be invoked across a differing ranges of data (sub)types.

  In contrast, planning and executing media transformations may be more flexible and effective (though perhaps less generic) by purposefully avoiding the use of an object-oriented data type hierarchy. Such an approach is employed by Marder [2].

### 3.2.4 Metadata

- **Data types and kinds**: As is the case with media elements, there is no fixed and final set of data types for metadata. Still, many kinds of metadata can be represented using simple data types such as numbers and strings. What is most interesting is the semantic intention of metadata.

  For a given media element, metadata can be stored about the *inherent* qualities of the media element itself (e.g., for an audio segment, its sample rate, number of channels, bits per channel, coding format, etc.). In addition, other information can be stored — information which is not directly related to the media element's inherent nature (e.g., time and place recorded, type of equipment used, person responsible, classification as 'open', 'confidential', 'secret', etc.)
- **Degree of automaticity**: This issue concerns the degree to which metadata is automatically obtained or estimated (and eventually recorded for later use). Furthermore, there is an issue as to *when* this information is obtained. That is, some kinds of metadata (e.g., media-inherent metadata) can be automatically accessed / derived and stored once, prior to the eventual use of the associated media data.

  Other kinds of metadata may require manual entry. This latter kind can often be added at any time, thereby yielding potential "gaps" where metadata about certain media elements is incomplete (and/or inconsistent).
- **Storage and packing**: The decision as to which metadata is obtained, estimated and eventually stored is usually dependent upon the domain of applications within which the associated media data is foreseen to be used. Precisely how the metadata is represented, where it is recorded and how it can be accessed it also dependent upon these conditions.

  Sometimes, metadata is stored external to the raw data (i.e., in tables or files other than the tables or files in which the raw media data is stored). Sometimes, it is stored internally, that is, together with the raw data, as demanded by certain multimedia standards. Depending upon the application, metadata can also be found both internal and external to the raw data, in order to help facilitate certain kinds of application functionality and/or boost application performance.

### 3.2.5 Transformation approach

The precise approach chosen for the transformation of (collections of) media elements involves a number of interdependent factors and tradeoffs. Some of the most relevant issues are included below.

- **Performance requirements**: Quality of result and transformation speed are two requirements which cannot be bypassed. Assessment of result quality could involve any number of objective or subjective criteria, such as error rate for an speech-to-text function; error rate for auto-identification / tracking / extraction of objects of interest in a moving scene; selection of cropping area, for an auto-cropping facility, etc.

  Transformation speed can be assessed across several dimensions as well; for example: the time required to derive a media transformation plan, in contrast to the time required to execute the plan. In distributed solutions, “transformation speed” may also include the time required to deliver transformation results between distributed system entities.

Certainly, there exists any number of other performance requirements; these are often directly associated with specific use contexts and applications.

- **On-demand vs. background transformation processing**: Most media transformation solutions perform a significant amount of transformation activity *on-demand*, that is, when a given transformation request is being processed. Transformation approaches may also include a greater or lesser extent of *pre-processing* and/or *background processing*. Such processing can be performed on (collections of) media elements, transformation rules, metadata, and more.

  Certain pre-processing may be carried out in order to satisfy the design or requirements of the transformation service itself. An example of this kind is provided by Mohan, et. al. [6], where an “InfoPyramid” is first generated and stored prior to use. This data collection contains alternative forms of each media element such that at run-time, the system can efficiently select and deliver the form most appropriate to the requesting client; often, little to no further transformation / customization of the media element(s) is required.

  Other benefits can be realized from background processing which helps the system meet general requirements for transformation speed and/or other performance factors. Here, one can choose to distinguish between background processing tasks having different levels of priority. One kind are “just-in-case” tasks — tasks performed based upon *expectations* of near-term system use. An example here might be the transformation and temporary caching of a new, popular music video into the kinds of formats most often requested by service users.

  Another kind of background processing task may concern the update of information which could help the transformation planner. An example of this sort could include the analysis of the set of available, *individual* transformation rules, in order to derive “composite” transformation rules. Each such composite transformation rule could consist of a subset of individual rules, collected together and organized as a single, well-specified set of operations. Here, the individual rules within a composite rule could be specified to operate in sequence or parallel, in order to efficiently achieve a more high-level transformation. Such analysis could allow for pre-computation of (high-level) transformation operations and their associated *costs*, such that transformation planning could become more efficient at runtime[4].

[4] Furthermore, new high-level composite transformation rules could be derived in the background, whenever new individual rules are introduced. Once derived, the new composite rules could become available to the transformation planner.

Yet another candidate for background processing are "low-level" tasks, such as text indexing and/or automated inter-relation of metadata. Tasks of this kind could be carried out in the background, whenever the system "has nothing else to do".

- **Media element versioning**: In use contexts where media elements are copied (or cloned), edited (or transformed) then stored back within the MM DBMS, it can become vital to include a media element versioning facility within the multimedia production system. Such a mechanism associates with each media element certain metadata about its ancestry.

  In its minimal form, this kind of facility should include information making it possible to track the sources from which each media element has originated. More complex mechanisms could attempt to additionally record certain metadata about the manner in which each media element has evolved from its original sources.

- **Local vs. distributed transformation processing**: This issue concerns the degree to which transformation planning and plan execution are performed across distributed system entities. Consider this: The decision machine must have awareness of — or have the capability to become aware of — all transformation rules which could be efficiently and effectively utilized within a transformation planning stage. This condition implies that relevant transformation rules are available locally — or can become available remotely — during planning.

  Design decisions may stipulate that only rules available locally can be used during planning, and that no new rules may become available to the planner until some specific planning routine(s) are complete. Alternatively, the planner could be designed so as to seek out new or specific kinds of rules from well-known distributed resources, when needed.

  This latter approach is clearly more complex; it requires an agreed-upon manner by which to declare input and output information[5] (or properties) for rules, as well as each rule's estimated execution costs (including eventual media element "shipping costs", when rule execution must transpire on remote entities). Further, it requires common protocols by which to access and acquire this information. Of course, this approach also introduces unavoidable delays during the planning process. In good cases, these extra costs are balanced by increased overall transformation power and flexibility.

- **Use of standards**: When transformation rules and execution resources are accessed and/or employed in a distributed environment, information about these resources must be available in some kind of standardized form. Presently, there is no standardized representation for the *properties* of transformation rules; different efforts have usually produced independent variants of such representations. However, there are certain trends to employ XML-based markup languages as rule property representation languages (e.g., [5]).

[5] This information has also been denoted as "signatures" [2] and, in a stream binding context, denoted as "(flow) type" information [15].

SOAP [19] — along with WDSL [20] and UDDI [21] as they evolve — together have the potential to function as standardized technologies for advertizing and/or exchanging information about rule properties in a distributed environment. For a stream binding context [15] — aspects of which can be likened to the media transformation problem — IDL [22] has been the basis for a media gateway description language (GDL) [16]. GDL allows declaration of the input/output properties of "media gateways", entities which can transcode and/or scale media streams. Further, these GDL-based declarations enable clients to find media gateway services (e.g., transcoding) at runtime [16].

In contrast to rule property representation and localization of distributed transformation services, *distributed execution* of rules is unlikely to be satisfied by standards such as SOAP and iCAP [18] [8]. Use of HTTP by these technologies can effectively prohibit them from handling media elements which can become arbitrarily large (such as high-resolution, full-color video films). In such cases, distributed transformation services should employ transformation rules and mechanisms which can efficiently handle streams as input and output data types. Use of mechanisms such as media gateways [16] could offer such a start.

- **Addressing preservation of temporal synchronization**:
  This issue concerns situations where one of the media elements requested functions as a container (or package) for other media elements. Further the package includes either direct or indirect specifications about temporal synchronization amongst the elements it contains.

  A simple example can be a media element package which contains one element representing a video and another element representing subtitles associated with the video. Should the transformation request include constraints specifying that the video must be reduced in frame rate, it is necessary that the transformation process yields a result in which the video and subtitles remain temporally aligned.
- **Coping with spatial layout specifications and constraints**:
  This issue likens that of the one concerning preservation of temporal synchronization amongst media elements within a containing package. In this case, however, the package includes specifications about the spatial layout amongst the elements it contains. In contrast to the temporal synchronization case, this case opens for a different degree of "transformation strictness". That is, transformation normally includes a requirement to *preserve* temporal synchronization amongst the media elements within a package. Here, spatial layout amongst package elements can usually be somewhat *compromised* when transformed; the degree to which layout compromise may be acceptable is primarily dependent upon the constraints within the transformation request.
- **Acquisition of client profile / preferences information**: Media element transformation requires some kind of specification of target constraints for the media elements desired by the requesting source (e.g., client). Such constraints usually arise from client-side preferences and/or limitations. Constraint

information can sometimes be explicitly found in the request when first issued. Technologies such as CC/PP [23] can also be used to augment the request with data describing client capabilities [4].

Alternatively, it's possible in a "user registration phase", to explicitly request information about the user's terminal set. This information can be stored on the server and later used together with "magic cookies" [24], in order to help automate fetching information about client capabilities when the request is received.

### 3.2.6 Optimization of transformation

- **Atomic vs. global planning**: This issue concerns whether the transformation approach / solution is designed to:
    - derive individual, optimized transformation plans, one for each atomic media element or, alternatively,
    - to derive a transformation plan — optimized across *all* requested media elements — in a single transformation planning stage.

  In the first case, the planner is limited to consider one media element at a time, and can risk delivering plans which contain local optimizations, yet be sub-optimal when considered across all requested media elements.

  Solutions of the latter kind are much more difficult to realize: the search space is more complex, as is the design of the cost and difference functions. In these solutions, it is often necessary to design clever strategies and mechanisms which balance search effort against plan quality.
- **Search strategy for the planner**: The planner's search approach can be more or less complex; this design decision often rests upon two major factors:
    - the expected transformation complexities to be encountered within the use context, and
    - whether an atomic or global planning approach is adopted.

  In contexts where element transformation commonly requires only "one step" (e.g., transcoding a .JPEG image to .BMP), the search approach could be a simple algorithmic routine which selects the appropriate transformation rule using data found in an association list.

  For contexts where more than one transformation rule is required, but atomic planning is used, the runtime performance of an algorithmic planner can still be acceptable when cost functions are accurate and the size of the ruleset is rather moderate.

  When the use context can call for multi-step transformations and/or when a global planning approach is used, there may quickly arise a need to control the planner's search effort using heuristics. For example, one can use an evaluation function which includes both a cost function *and* a difference function [14]. Even with good cost functions, devising accurate difference functions in these contexts can become quite difficult.

Here, heuristic solutions can be employed, where distance functions and/or pruning mechanisms are purposefully designed to reduce the size of the search space examined by the planner. In addition, organizing transformation rules into groups and controlling the planner's access to these groups can also help reduce search complexity.

### 3.2.7 Evaluation functions for plan optimization

Multi-step and/or global planning approaches can usefully employ evaluation functions in order to more derive and *sort* transformation plans; these functions can be used balance the degree of plan optimality against the amount of search effort spent by the planner. In simplified form, these functions can liken

$f(x) = g(x) + h(x),$

where $f(x)$ is the value of the evaluation function, $g(x)$ is the value of the cost function, and $h(x)$ is the value of an (optional) distance (or "difference") function [14]. In this simplified evaluation function, the parameter $x$ represents a partial plan developed by the transformation planner. The cost function utilizes information about that partial plan in order to estimate the cost of applying some (sequence of) transformation rules [11]; the cost function may also include the planner's expenditures used to develop that partial plan.

When present, the distance function is used to estimate the cost difference between a partial plan and an optimal plan; it may also include estimates of eventual planner expenditures.

In short, parameter values used within these cost and difference calculations are usually acquired from metadata relevant to the media elements to be transformed, the transformation rules themselves and/or examination of data about planner activity.

- **Parameter domains**: Evaluation functions can include a variety of parameters. Some examples are:
    - rule-specific parameters, e.g., estimated transcoding time, resource consumption, complexity, quality degradation, etc.,
    - factors relevant to distributed processing, e.g., estimated "shipping costs" for media elements (i.e., costs such as delay), resource availability estimates, etc.,
    - parameters describing an end user's quality demands and preferences (e.g., "maximum wait time", "deliver audio only should video quality drop below a specified threshold", "prioritize audio ahead of graphics and images", etc.)
    - parameters describing the *media author's* quality demands and preferences (e.g., content priorities, content interdependencies, etc.)
- **Availability of parameter values**: At planning time, it is necessary that the parameters to be used in evaluation functions have *values* — preferably values which are as accurate as possible. The issue here is whether these values are the result of information which is automatically derived (either pre-computed or at

runtime) or, whether the values arise from metadata which is manually supplied. In the latter case, absence of data can, in the worst case, cause the planner to halt. Otherwise absence of data (or poor data) can lead to inaccuracies in cost estimation which, in turn, can lead to poor transformation plans.

- **Responsibility for parameter values**: The examples provided above indicate that the availability of planning-relevant data can be the responsibility of the system (e.g., auto-derived data) or the responsibility of one or more individuals. These actors could be persons responsible for system development and performance, persons responsible for content authoring and development and/or the end-users themselves. In different systems and use contexts, there may also be discrepancies between the actors responsible or accountable for relevant metadata and the actors wishing to have *control* over that same metadata.

### 3.2.8 Grouping transformations

As mentioned earlier, certain transformation planning solutions can be made more effective / efficient by interleaving the plan derivation and plan execution stages. Iterative plan derivation and execution can also be a useful approach. When employing evaluation functions, such solutions can benefit from the organization of transformation plans into (possibly overlapping) sets. These sets can be characterized by properties such as:

- rule cost and complexity
- location of rule execution (local or remote)
- whether a rule has a natural, “lossless” inverse (e.g., such as certain pairs of compression / decompression functions)
- whether a rule is commutative and/or associative
- whether or not a rule leaves a media element’s *type* intact (e.g., image cropping rules vs. speech-to-text functions)
- whether rules assemble or dissemble composite media elements
- the degree to which a rule increases or decreases the volume of the media element(s) it transforms.

# 4. Summary

This work has tried to the describe issues important to media element transformation processes from a general perspective. It has had a specific focus upon transformation processes which can require *sequences* or *sets* of transformation operations, in order to convert source media elements into requested target element types and forms.

This presentation has chosen to define this process as involving two different stages: transformation planning and transformation execution. Some of the issues elaborated upon here tend to focus more directly upon the planning stage, while others tend to be more relevant to the plan execution stage or to multimedia production in general.

## 5. Future Work

As part of our own project effort in channel S, we shall investigate alternative architectural approaches for supporting multimedia multichannel content production and services. The preliminary work presented here has been valuable in preparing an understanding as to the variety of technical issues one can face when addressing problems related to media transformation.

Currently, we are beginning preliminary conceptual work upon an architectural framework for support of media transformation. Specifically, we intend to address approaches and techniques for supporting "just-in-case" media transformation. As mentioned earlier in section 3.2.5, these kind of transformations are usually performed in the background. Such techniques aim to help systems meet general requirements for on-demand media services by selecting and performing transformation tasks based upon expectations of near-term system use.

In order be more efficient, decision machines employed in these kind of approaches can utilize information about:

- the expected "window of popularity" for individual media elements
- the media element formats expected to be most often requested
- media element packaging and metadata
- the kinds of transformation resources at disposition
- costs associated with use of a transformation resource
- resource availability
- scheduling, etc.

In our further work, we shall investigate alternative ways of using these various kinds of information within a transformation framework, as part of a more encompassing multimedia multichannel media architecture.

## 6. References

[1] channel S WWW site: http://www.nr.no/channelS/

[2] Marder, U., “Transformation Independence for Multimedia Systems”, Internal Report, University of Kaiserslautern, April 2001 (24 pages). See http://wwwdbis.informatik.uni-kl.de/pubs/papers/Ma01b.pdf

[3] M.P. Wellman, “Rationality in Decision Machines”, Position paper presented at the AAAI Fall Symposium on Rational Agency, November 1995. See: http://ai.eecs.umich.edu/people/wellman/decision-machine.html

[4] L. Suryanarayana, J. Hjelm, “CC/PP for Content Negotiation and Contextualization”, K.-L. Tan et al. (Eds.): MDM 2001, LNCS 1987, pp. 239-245, 2001. Springer-Verlag Berlin Heidelberg 2001. See: http://bim.im.fju.edu.tw/home/paperreading/agnetpaper/cc-pp.pdf

[5] T. Phan, G. Zorpas, and R. Bagrodia, "An Extensible and Scalable Content Adaptation Pipeline Architecture to Support Heterogeneous Clients," To appear at The 22nd International Conference on Distributed Computing Systems (ICDCS 2002), July 2002. See: ftp://pcl.cs.ucla.edu/pub/papers/icdcs2002-cap.ps.gz

[6] R. Mohan, J. Smith, C.-S. Li, "Adapting Multimedia Internet Content For Universal Access," IEEE Transactions on Multimedia, March 1999, pp. 104-114. See: http://www.research.ibm.com/networked_data_systems/transcoding/Publications//ieeemm.pdf

[7] T. Bickmore and B. Schilit,, “Digestor: Device-independent Access to the World Wide Web”, Proceedings of the Sixth International World Wide Web Conference, Santa Clara, California, 1999. See: http://www.fxpal.com/PapersAndAbstracts/papers/bic97/

[8] Wei-Ying Ma, Bo Shen, and Jack Brassil, “Content Services Network: the Architecture and Protocols”, In Proceedings of the 6th International Web Caching Workshop and Content Delivery Workshop, Boston, MA , June 2001. http://www.cs.bu.edu/techreports/2001-017-wcw01-proceedings/143_ma.pdf

[9] Hollfelder, S., Schmidt, F., Hemmje, M., Aberer, K., Steinmetz, A.: Transparent Integration of Continuous Media Support into a Multimedia DBMS. In: Proc. Int. Workshop on Issues and Applications of Database Technology (Berlin, Germany, July 6-9), 1998. See: http://www.ipsi.fhg.de/oasys/reports/ftp/pdf/P1998-08.pdf

[10] M. Cherniack, S. B. Zdonik, “Rule languages and internal algebras for rule-based optimizers”, Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, 1996. See: http://www.cs.brandeis.edu/~cs227b/papers/brandeis-kola-sigmod96.pdf

[11] K.S. Candan, V.S. Subrahmanian V. Rangan, “Towards a Theory of Collaborative Multimedia”, Proc. of the IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan, June 96. See: http://www.cs.umd.edu/projects/hermes/publications/postscripts/ttcm.ps

[12] Chen, J.L., Yang, Y.D., and Zhang, H.J., “An Adaptive Web Content Delivery System”. Proc. AH2000 (Tronto, Italy, 2000) Springer Press, 284-288. See: http://www2.sis.pitt.edu/~jlchen/publications/ah2000.pdf

[13] R. Han, P. Bhagwat, "Dynamic Adaptation In an Image Transcoding Proxy For Mobile Web Browsing", IEEE Personal Communications Magazine, Dec. 1998, pp. 8-17. See: http://www.research.ibm.com/networked_data_systems/transcoding/Publications/IEEE1298.pdf.zip

[14] Nils J. Nilsson, *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto, CA, 1980.

[15] H. O. Rafaelsen and F. Eliassen, “Trading and Negotiating Stream Bindings”, Proceedings of Middleware 2000, IFIP/ACM International Conference on Distributed Systems Platforms, New York, NY, USA, April 2000. LNCS 1795 (Joseph Sventek, Geoffrey Coulson (Eds.), pp. 289-307, Springer-Verlag Berlin Heidelberg 2000. See also: http://www.ifi.uio.no/~frank/papers/paperMW2000.pdf

[16] H. O. Rafaelsen and F. Eliassen, "Design and performance of a media gateway trader”, to be published in the Proceedings of the 4th International Symposium on Distributed Objects & Applications (DOA ’02), Irvine, California, 2002.

[17] Synchronized Multimedia Integration Language (SMIL) 1.0 Specification. See: http://www.w3.org/TR/REC-smil/

[18] Internet Content Adaptation Protocol (ICAP) Forum WWW site: http://www.i-cap.org/

[19] Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000. See: http://www.w3.org/TR/SOAP/

[20] Web Services Description Language (WSDL) Version 1.2. See: http://www.w3.org/TR/wsdl12/

[21] Universal Description, Discovery and Integration (UDDI). See: http://www.uddi.org/

[22] IDL Introduction from OMG: http://www.omg.org/gettingstarted/omg_idl.htm

[23] Composite Capabilities Preference Profiles (CC/PP) Working Group; see: http://www.w3.org/Mobile/CCPP/

[24] Explanation of ‘cookies’, as provided by the O'Reilly Open Books Project: http://www.oreilly.com/openbook/cgi/ch10_08.html

[25] Macromedia Flash: http://www.macromedia.com/software/flash/