A KNOWLEDGE SUPPORTED APPROACH FOR MULTI-STEP MEDIA ADAPTATION

Dietmar Jannach[1], Klaus Leopold[2], Hermann Hellwagner[2], and Christian Timmerer[2]

[1] Department of Business Informatics and Application Systems
[2] Department of Information Technology
University Klagenfurt, Austria
{Dietmar.Jannach, Klaus.Leopold, Hermann.Hellwagner, Christian.Timmerer}@uni-klu.ac.at

ABSTRACT

In order to enable transparent and augmented use of multimedia resources across a wide range of networks and devices, the forthcoming MPEG-21 standard aims at integrating the various existing technologies for delivery and consumption of digital content in a common multimedia framework. *Digital Item Adaptation* is one of the core concepts of the framework that will support the adaptation of multimedia resources to device capabilities, underlying network characteristics, or the user's preferences. The scope of the standardization, however, is limited to the definition of description tools and does not deal with the internals of the adaptation process itself.
In this paper, we first discuss the requirements of the resource adaptation component of an adaptation engine. These requirements include, for instance, openness for the integration of external multimedia transforming tools as well as intelligent decision taking when determining the set of required adaptation steps. We also present a prototype of a simple video resource adaptation engine that completely relies on descriptions of the resource itself (MPEG-7), the usage environment of the resource (MPEG-21), as well as declarative descriptions of the transformation tools. The prototype employs a knowledge-based engine for finding and executing the needed adaptation sequences.

1. INTRODUCTION

The MPEG-21 Multimedia Framework aims at providing a technological basis for augmented and transparent use of multimedia resources across a wide range of networks and devices [1]. A core concept for such transparent provisioning and consumption of multimedia content is *adaptation*, addressed by the *Digital Item Adaptation (DIA)* part of MPEG-21 [2]. The general idea is that multimedia content is adapted before or even during distribution where session characteristics like network bandwidth or reliability as well as the capabilities of the end user's device are dynamically taken into account. In the MPEG-21 framework, specifically in DIA, only parts of the required adaptation capabilities and activities are standardized. Rather, as depicted in Figure 1, DIA mainly specifies the *tools* that have to be used for describing, e.g., the user's preferences, his/her usage environment or the format of the resource itself. These descriptions are provided in terms of XML documents. The actual engines that perform the adaptation of the content are non-normative and the implementation is left to vendors of adaptation components.

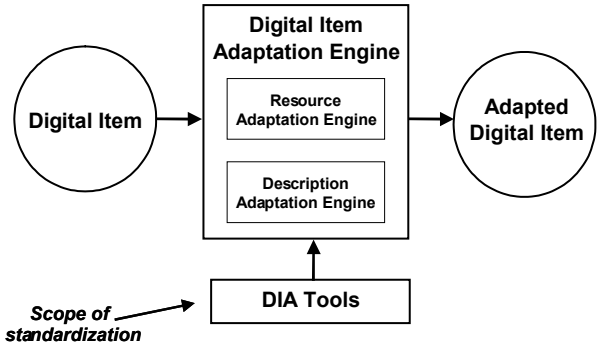

Figure 1 Scope of MPEG-21 DIA (from [2])

Nonetheless, one specific requirement mentioned in [3] concerns multi-step adaptation: the multimedia framework "*... shall support a convenient and efficient way to concatenate multiple adaptation steps"*. This series of adaptation steps will typically be performed on a server, network device, or on the receiving terminal.

Up to now, this specific requirement is only marginally addressed in current standardization efforts. In this paper, we discuss the requirements of a flexible resource adaptation engine that is able to perform multiple media resource adaptation steps. We also present the architecture of a prototype system that makes use of normative DIA tools and extends them with declarative (XML-based) capability descriptions of adaptation modules possibly provided by different vendors. These specific descriptions, together with the standard descriptions of media format and usage environment, allow us to employ a knowledge-based approach for computing and executing a series of adaptation steps on the original media data.

2. APPROACH

In this section, we discuss requirements of an open multi-step adaptation engine and present the rationale and concepts behind our proof-of-concept implementation.

Scenario and requirements. In a typical application scenario, a client using a device with specific properties (e.g., a given display size or a given network bandwidth) requests a multimedia resource from a server. The idea of adaptation is that the resource is stored at the server in exactly one defined format and is transformed according to the client's preferences and capabilities before it is streamed over the network. Given the client's usage environment and the original media format, the server has to apply a set of transformation operations, like removing the color or changing the resolution of the original resource. Note that we cannot generally assume that there exists a single software library or tool that is capable of performing all potentially required transformation operations, let alone in a single step. Consequently, we need an approach that allows us to compute multi-step adaptation plans, where combinations of different external tools may be needed. In order to leave the framework open and extensible, general tool-independent mechanisms must be defined for describing the external tools' capabilities that will be exploited in the plan generation process. Furthermore, these mechanisms must be robust with regard to changes in cases where new tools are to be used or new descriptive metadata for the resource is available. Finally, one prominent goal for such a framework must be that it interoperates with existing standards and technologies.

Metadata needed for adaptation. The information needed for an adaptation process comes from various sources. The description of the original media format, i.e., which codec is used or other properties of the resource, has to be contained in MPEG-7 [4] descriptors that are stored together with the original resource on the server. Within the MPEG-21 framework, the client's preferences as well as the terminal capabilities are contained in a Usage Environment Description [1], [5]. These two types of descriptions are encoded in a normative XML format. Now, the missing piece for multi-step media adaptation is a description of the available transformation steps and possibly also a specification of how these transformation steps can be actually performed on the source media with regard to tool invocation. Besides relying on a defined (XML-based) syntax, the mechanism for specifying the transformation steps must be expressive enough that one can capture the semantics of the transformations. Moreover, the chosen mechanism has to be flexible such that new terms for, e.g., describing the effects of an adaptation step, can be easily introduced.

Semantic adaptation step descriptions. In our approach, we view the problem of finding a suitable set of transformation steps as a classical state-space planning problem [6], where an agent is given a description of its current and desired environment and a set of world-altering actions it can perform. The goal of a planning process (planner) consequently is to find a sequence of actions in order to reach the defined goal state. The start and goal states are expressed as (logical) facts; the actions in a STRIPS-style [7] planner are described by means of a set of pre-conditions and effects of the execution of an action. This form of representation has shown to be simple but on the other hand expressive enough for various problem domains and is also used today in the context of Semantic Web Services [8], where the service semantics are expressed using the same mechanism.

Example. In the following example, we show how the different descriptions work together and can be used for the generation of an adaptation plan[1]. Table 1 shows relevant fragments of the resource format and the usage environment descriptions for a specific request, where we see that resolution, color, and the encoding have to be adapted.

Content description:	Usage environment description:
type: video stream	codec: mpeg-1
codec: mpeg-4	color: false
color: true	resolution: 320 x 240
resolution: 640 x 480	

Table 1 Content and usage environment description

In Table 2, we show how the semantics of two specific actions (grey-scaling and spatial-scaling) are encoded. The preconditions describe the state before invoking the action and the effects describe the state after the invocation, i.e., the fulfilled constraints of the user's usage environment. In this description, semantic features are assigned to the function parameters like, for instance, the mapping of the argument *xdim* to the MPEG-7 descriptor *width*. After the action is performed the width descriptor is invalid and thus, updated with the value of the argument *newxdim*. Another effect is the achievement of a usage environment constraint which is annotated with the MPEG-21 descriptor *horizontal*. Note that, when using this form of action descriptions, the gap between MPEG-7 media information terms and MPEG-21 usage environment description terms is bridged. Furthermore, when using such a declarative knowledge representation scheme, no changes to the planner have to be made, once a new term is introduced.

[1] Note that for the sake of readability, we use an informal notation rather than the internal XML representation.

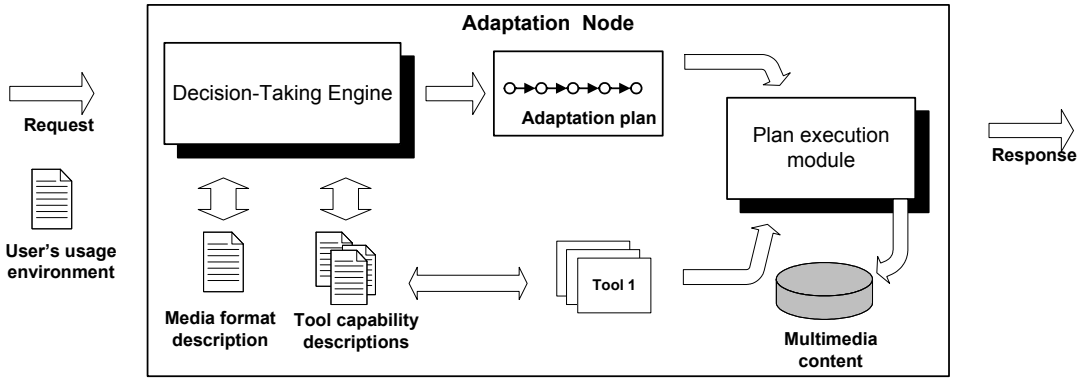

Figure 2 Architecture overview

Spatial scaler	***Grey scaler***
module: Scaler *function: spatialscale* ***Inparameters:*** *fb, xdim, ydim,* *newxdim, newydim* ***Outparameters:*** *newfb* ***Preconditions:*** *isFrameBuffer(fb),* *width(xdim),* *height(ydim),* ***Effects:*** *isFrameBuffer(newfb),* *width(newxdim,)* *height(newydim),* *horizontal(newxdim),* *vertical(newydim)*	*module: Colorutils* *function: greyscale* ***Inparameters:*** *fb* ***Outparameters:*** *newfb* ***Preconditions:*** *isFrameBuffer(fb),* *colorDomain(color)* ***Effects:*** *isFrameBuffer(newfb),* *colorDomain(greylevel)* *colorcapable(false)*

Table 2 Example capability description

Using this information as well as similar descriptions of encode and decode actions, a planner can derive the following transformation sequence for the example problem, where variables with the same name are forwarded from one action to the next one.

Transformation sequence:
decode(fb1, mpeg4, fb2)
spatialscale(fb2, 640, 480, 320, 240, fb3)
greyscale(fb3, fb4)
encode(fb4, mpeg1, fb5)

3. ARCHITECTURE / IMPLEMENTATION

Figure 2 summarizes the architecture of our framework and shows how adaptation is done after a client's request. The main components are the knowledge-based planner, (Decision-Taking Engine – DTE) that computes the needed adaptation plans, and the Plan execution module, that actually performs the plan and invokes external transformation tools like the ffmpeg library2. Besides the needed XML-manipulation tools, the core of the DTE is built by a small Prolog-based state-space planner [6]. From the perspective of search complexity, we know that planning is in general NP-complete. Nevertheless, our experiences show that the length of the required plans is very limited, typically around five to ten steps and the number of possible transformation actions is limited to a few dozens. In our test scenarios with an un-optimized implementation, the search for an adaptation plan in all cases where a plan existed took less than one second. Typically, the time constraints for the plan generation step are not too hard, because the actual transformation is the time-consuming part of the adaptation process. Nevertheless, good improvements can be reached when techniques like plan caching are introduced where for similar client requests already pre-computed plans can be reused. Regarding the quality of the transformation plans, at the moment our planner always produces valid plans.

Regarding performance, however, the incorporation of additional planning heuristics must be considered in the future. That is, it might be better to reduce the size of a frame before performing any other operation, rather than the other way round, although both orders would result in the desired format. Moreover, we will search for techniques that allow us to compute "good" plans for cases where the user preferences cannot be fulfilled exactly.

After the computation of the required adaptation steps, the plan execution module actually executes these transformations before the modified resource is sent to the client. Because the adaptation framework has to be interoperable and extensible for (fast) existing media libraries, the design of this component was chosen as follows. When an action is declared in the style described above, we can extend these descriptions with tool-specific in-

[2] http://www.ffmpeg.org

formation, for instance, the name of the library where some action is implemented and the specific function name. We then utilize these descriptions to generate connector components or at least skeletons of such components that can be loaded at run-time into the plan execution module's Java Virtual Machine without a need for code changes in cases where a new tool is available. When incorporating external, C++ based tools, the actual invocation of a function of the external library has to be implemented by adding tool-dependent code (using, e.g., the Java Native Interface) to the generated connector skeleton.

4. MPEG-21 BSD AND ADAPTATION QOS

The MPEG-21 framework also supports tools for adaptation. In contrast to our approach, this work is based on *Bitstream Syntax Descriptions* (BSD) [9], i.e., an additional metadata layer which describes the high-level structure of a media bitstream. BSD can be used to perform manipulations on a compressed bitstream without costly encoding or decoding. Parts of the BSD framework enable resource format-agnostic adaptations and provide semantic handles to sections of the bitstream (so-called *markers*) that facilitate semantic-based manipulations like, e.g., removal of violent or x-rated scenes [10]. The main limitation of this approach is that one can only perform editing-style adaptation operations like removing, inserting, or updating parts of the bitstream. BSD can be used together with *Adaptation QoS* [2], another tool defined within the MPEG-21 framework which enables users (primarily content producers) to describe the terminal and network quality of service. It specifies the relationship between environmental constraints, media quality, and feasible adaptation operations. For example, a constraint could be the limitation of the network throughput towards the client. Adaptation of a video in this case could be accomplished by performing operations like, e.g., frame dropping, coefficient dropping, or wavelet reduction, which would lead to changes in the quality of the multimedia stream. Consequently, Adaptation QoS is useful to hint adaptation nodes to fulfill given constraints with regard to the quality of the adaptation result.

We see our work as *more general* and on a higher layer than BSD and Adaptation QoS. BSD could be one adaptation step in the presented architecture, providing a fast adaptation mechanism for the compressed domain. Furthermore, Adaptation QoS could be integrated into our approach, providing a basis for deriving a heuristic used by the planner to find, e.g., the best adaptation steps with regard to quality.

5. CONCLUSIONS

We have presented an approach for multi-step adaptation of multimedia resources. Our work is based on semantic descriptions of transformation steps which are exploited by a classical state-space planner. The knowledge-based approach enables us to compute adaptation plans independent of specific tool implementations. Thus, existing tools like, e.g., BSD or Adaptation QoS can be deployed with our framework. Furthermore, our approach is robust against changes in terminology and structure in the MPEG-standardized media and usage environment descriptions. A proof-of-concept implementation showed the feasibility of knowledge-based adaptation planning and includes a generic plan execution module that is open for the incorporation of external libraries.

Our future work includes the development of a stable and fast system that can be deployed on an adaptation-capable network node (like a proxy server) as well as further experimentation in various application settings. We will also investigate whether the described mechanisms can be extended for situations where the *content* has to be adapted, like in situations where the user's preferences are to remove certain scenes in a movie.

6. REFERENCES

[1] ISO/IEC JTC 1/SC 29/WG 11 N6042, Revised Draft PDTR 21000-1, 2nd Edition, Gold Coast, October 2003.

[2] Vetro, A. and Timmerer, C., (Eds.) ISO/IEC JTC 1/SC 29/WG 11 N5845, Text of ISO/IEC 21000-7 FCD – Part 7: Digital Item Adaptation, Trondheim, July 2003.

[3] Bormans, J., (Eds.), ISO/IEC JTC 1/SC 29/WG 11 N5873, MPEG-21 Requirements v 1.5, Trondheim, July 2003.

[4] Martínez, J.M., Koenen, R., Pereira, F.: MPEG-7: the generic multimedia content description standard, IEEE Multimedia, Vol. 9(2), April-June 2002.

[5] Pereira, F., Burnett, I., Universal multimedia experiences for tomorrow, IEEE Signal Processing Magazine, Special Issue on Universal Multimedia Access, Vol. 20, March 2003.

[6] Bratko, I., Prolog - Programming for Artificial Intelligence. Addison-Wesley, 3rd edition, 2000.

[7] Fikes, R. E. and Nilsson, N. J., STRIPS: A new Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, 2, pp. 189-208, 1971.

[8] McIlraith, S., Son, T.C., and Zeng, H.. Semantic Web Services. IEEE Intelligent Systems, Special Issue on the Semantic Web, Vol. 16(2), pp. 46-53, March-April 2001.

[9] Panis, G., Hutter, A., Heuer, J., Hellwagner, H., Kosch, H., Timmerer, C., Devillers, S., Amielh, M.: Bitstream Syntax Description: A Tool for Multimedia Resource Adaptation within MPEG-21. Signal Processing: Image Communication, Special Issue on Multimedia Adaptation, Vol. 18(8), pp. 597-767, September, 2003.

[10] Timmerer, C., Panis, G., Kosch, H., Heuer, J., Hellwagner, H., Hutter, A. Coding format independent multimedia content adaptation using XML, SPIE International Symposium ITCom 2003, Vol. 5242, Orlando, Florida, 2003.