A Knowledge and Component Based Multimedia Adaptation Framework

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

Abstract

The rapid evolution in the hardware sector brought up various (mobile) end user devices like PDAs or cell phones on which online multimedia content can be consumed. Due to different capabilities of these devices as well as individual user preferences, the original multimedia resources have to be adapted in order to fit the specific devices' constraints and to meet the users' requirements. Given the high variety of possible adaptation operations both on the format as well as the content level, an intelligent multimedia server must be able to integrate multiple existing and specialized adaptation tools.

In this paper, we demonstrate how the usage of modular software components and declarative descriptions of component behavior enables us to develop extensible multimedia adaptation systems. The precise semantics of the utilized functionality description mechanism as well as the defined vocabulary from existing and emerging multimedia standards also allows us to automatically assemble adaptation chains that are executed on a given resource involving multiple, externally provided software components.

1. Introduction

The achievements of the hardware industry have influenced the area of communication and thus, multimedia access. Today's video and music consumption is no longer bound to special purpose devices like, e.g., analogue TVs and radios. Digital multimedia can be enjoyed on a variety of end devices such as regular desktop PCs, notebooks, web TVs, and even on mobile devices like PDAs and cell phones. These devices use network connections with different characteristics over which they can receive multimedia content. Some of these connections feature high bandwidth throughput, others provide only limited bandwidth. Furthermore, the amount and richness of multimedia content - especially on the Internet - is increasing enormously. The availability of a variety of multimedia compression standards enable content providers to easily publish audio and video clips in addition to standard images and text.

This complex landscape of multimedia content, the diverse set of terminal devices with different capabilities, and heterogeneous networks with dynamically changing conditions, in many cases results in interoperability problems. Therefore, one major goal in today's multimedia research is the development of *Universal Multimedia Access* (UMA) capabilities, where a user is enabled to consume any resource, anywhere, and anytime [16]. To achieve this goal, the multimedia content has to be adapted to meet the limitations of a user's terminal and network. Such multimedia adaptation could be, e.g., transcoding from one video format to another or scaling a video in the spatial domain such that it fits on the terminal's screen. Furthermore, the content must also be adapted such that a user has an equivalent, informative experience anytime, anywhere; i.e., the end point of universal multimedia consumption is the end user and not the terminal [16]. *Universal Multimedia Experience* (UME) might include, e.g., insertion of subtitles into a video to allow deaf users to follow the spoken content in a video.

Multimedia adaptation is a key means for addressing the challenges of UMA. The general idea is to transform the content before or even during distribution, with session characteristics like network bandwidth as well as capabilities of the end user's device being dynamically taken into account. As there is no single piece of software that supports the different forms of possible adaptations, we argue that the only feasible solution for the complex UMA problem lies in the integration of multiple (third party) *software components* into an adaptation framework, which is capable of applying multiple steps of adaptation methods to a multimedia resource.

In this paper, we propose a framework for the integration of multiple isolated adaptation components into a powerful

Proceedings of the IEEE Sixth International Symposium on Multimedia Software Engineering (ISMSE'04)
0-7695-2217-3/04 $20.00 © 2004 **IEEE**

and extensible adaptation system. We base our framework on declarative descriptions of the components' capabilities and effects of the execution of the components's functions. Consequently, we are then able to follow a knowledge-based approach for the computation of general *adaptation plans*. Such plans are chains of required transformation steps on a given multimedia resource to arrive at a desired format for presentation.

2. Multimedia Adaptation and Universal Multimedia Access

Multimedia adaptation is becoming increasingly important in the context of UMA. We define adaptation as the process of producing a variant of a media resource such that it matches the media consumption context. Such adaptation processes are typically driven by several forms of standardized meta-data that describe the media contents, terminals, network connections, client usage environments, as well as user preferences.

The Role of Meta-data in Adaptation. In *MPEG-7* [13], the ISO/IEC standardization group defined powerful XML-based mechanisms for describing and annotating multimedia content as a means for improved management of the vast amount of available multimedia information. The aim of MPEG-7 is to be generic, not targeted to a specific application or application domain. Therefore, the standardized descriptions range from low-level signal characteristics to high-level semantic information. The most important part for our work is *MPEG-7 media information* which contains information on the content like, for instance, storage format and encoding format of audio and video.

On the other hand, the *MPEG-21* standard aims to achieve interoperable and transparent access to multimedia content. The Digital Item Adaptation (DIA) part of the standard [20] addresses, among others, the description of the usage environment, in particular, terminal capabilities as well as characteristics of the network, user, and natural environment. Within the *terminal capabilities*, the encoding and decoding features of a terminal are described, again using the MPEG-7 description mechanism for the media format. Furthermore, input-output capabilities as well as device properties are described. The *network characteristics* describe the conditions and capabilities of a network. Within the *user characteristics*, usage and presentation preferences as well as accessibility characteristics and location characteristics are described. The audiovisual environment, location, and time are described in the *natural environment characteristics* of the usage environment.

The different Levels of Multimedia Adaptation. Adaptation of a multimedia resource can be done on different levels. We identified three main categories of adaptation: selecting content, modality conversion, and scaling/transcoding content.

Selecting content: Modern multimedia content is composed of a set of different media objects such as video, audio, text, synthetic scenes, etc. The main objective of content selection is to choose only parts of a multimedia stream for user consumption. Consider that a user wants to consume the latest CNN news on a car radio. As typical car radios do not support video display, the audio/video news have to be adapted such that only the audio stream is transferred to the car radio. In order to perform this kind of content selection, the adaptation algorithm has to be aware of the individual objects of the multimedia stream. This information can be expressed with MPEG-7 metadata. But content selection can also be applied on a higher, semantic level. For example, one might want to neglect transmission of X-rated scenes. The algorithms can obtain the required nature of scenes via the marker concept of MPEG-21 [19].

Modality conversion: Modality conversion is the ability to transform the type (modality) of the media resource. For example, a video stream may contain important key frames which are transmitted to a client as still images in case the device does not support video playout. Metadata mechanisms for modality conversion are currently being defined in MPEG-21.

Scaling/transcoding content: Scaling or transcoding multimedia content is done on the signal level of the media bitstream. For example, MPEG-4 Part 2 has come up with fine granularity video scalability [17]. The aim of this work is to provide a video format that can be scaled only by truncating the bitstream. In MPEG-21 Part 13 a wavelet based, scalable video codec (SVC) is being developed. Both approaches do not need metadata for adaptation because the bitstream itself is scalable. Transcoding of multimedia resources is done by (partly) decoding and re-encoding the multimedia stream. The resource is decoded, new parameters for the encoder are set to fulfill terminal and user constraints and finally, the stream is encoded again. The required parameter settings for the encoder can be derived by MPEG-21 usage environment descriptions.

Integration requirements for Adaptation in UMA The vision of UMA is that any user can consume any multimedia content anytime, anywhere. The multimedia stream is adapted on a node in a distributed multimedia system (e.g., server, intermediate transcoding proxy, or client) according to standardized metadata that describe the user's terminal, preferences, and usage environment.

Considering the above mentioned adaptation categories, numerous adaptation algorithms have to be implemented in an adaptation engine to fulfill the requirements of UMA.

Research work of the past years has produced many ef-

ficient adaptation techniques. We can find intelligent transcoding proxies which are able to transcode a video to a desired bitrate and which can perform algorithms like, e.g., greyscaling or spatial frame reduction [18]. In addition, sophisticated approaches for temporal video adaptation [12], modality conversion [1], and content selection [19] were developed. However, we think that in a general solution to the complex UMA adaptation problem, the integration of multiple (third party) *software components* into a comprehensive adaptation framework, capable of applying multiple steps of adaptations to a multimedia resource, is required.

In order to be integrateable in such a framework, the required tools have to be developed as encapsulated *adaptation components*, following the development principles of software components: independent and reusable code, well-defined interfaces, and well-defined behavior.

3. Component Based Software Development

Software Components. The idea of assembling systems from modular components with defined interfaces and functionality is omnipresent in many industrial sectors. In the past years, a lot of research work was done in order to apply that approach also to software development.

The core concept of component based software development is a *software component* which is defined as "an independently deliverable set of reusable services" [3].

Reusable services means that a component provides capabilities that many other components may wish to access. Thus, the functionality of a component has to be clearly described. Furthermore, it has to be specified how a component affects the system which is using it. A component can be replaced by another component if both feature the same functionality and behavior. Thus, well designed components are *replaceable* and *encapsulated*.

Independently deliverable means that components may not have structural dependencies among each other, e.g., they must not share a common data structure. Note that one component can depend on another, if both are isolated. For example, component A is capable of separating the audio stream from an audio/video file and component B is capable of decoding an audio stream. According to this, component B can only decode the content if component A has already separated the audio content from the audio/video stream. To realize interoperable components, clear interfaces have to be defined.

Interfaces. Interfaces specify how a user can interact with a component. Note that an interface is pure specification and does not tell anything about implementation. In fact, an interface may exist separately from any component which implements it. As a simple example, suppose an interface specification for a video decoder which is used by a video player before it renders the images. Different software companies can develop decoders for the player by strictly following the interface without touching the player's code. Furthermore, a company which is developing video players can decide which decoder it is using (buying) for the player. Thus, interfaces open up room for competition among software developers.

Behavior Description of Components. The behavior description is a very crucial part of a component. The goal is to offer a component's user a clear picture whether the component is useful for him/her or not. Brown and Short propose that component descriptions may consist of some or all of the following [3]:

- a list of operation names and signatures defining the input and output parameters for each operation;
- an informal textual description of the component's functionality, intended usage scenario, and history;
- an informal description of the component's operating context (e.g., hardware platforms, operating system), expected versions of installed software, and known limitations or deficiencies;
- performance and availability data for typical execution of the component.

The need for Multimedia Software Components for UMA. In order to integrate the various modules, tools, and software components which are required for the idea of universal multimedia access, well designed software components are necessary. For instance, a company that is working on semantics driven content selection does not want to mess around with details of video decoding and encoding which is probably required at the lowest level. On the other hand, a company developing video codecs simply wants to see that content selection is used with their codecs. Thus, a main requirement for complex, integrated multimedia software solutions is the development of exchangeable and encapsulated software components with well defined interfaces and behavior. Each component has to focus on performing *specific multimedia tasks*, like for instance, video decoding, audio encoding, image greyscaling, text to speech conversion, etc.

Currently, we can find a lot of powerful multimedia libraries such as ViTooKi, MPEG4IP, or FFMPEG[1], offering a broad range of multimedia manipulation functionality. For example, FFMPEG is a library supporting audio/video decoding and encoding in numerous formats. Furthermore, it supports multiplexing and demultiplexing of interlaced audio/video streams (objects). At a first glance, one might think of FFMPEG as a multimedia software component, designed for solving different multimedia tasks.

1 All projects are available on SourceForge: http://{vitooki, mpeg4ip, ffmpeg}.sf.net

But the problem with FFMPEG and similar tools is that all the functionality is squeezed into a single library, sharing a lot of common data structures. Such a multi-functional and therefore heavy-weight library is complicated to deploy in special-purpose software systems like, for instance, an audio player, or a video adaptation engine, etc. Even more, we argue that the challenge of UMA is too complex to be solved with all-in-one solutions. The integration of multiple small, encapsulated, and exchangeable multimedia components to larger multimedia software systems has to be the goal of future multimedia software engineering and development.

Figure 1 illustrates the idea of multimedia systems based on single software components. In this example, a CNN news show is streamed to a PDA and to a car radio. A video adaptation system and a content selection system is needed to facilitate UMA. Both systems feature certain multimedia manipulation components which are applied on the incoming multimedia content. The video adaptation engine demultiplexes the incoming MPEG-4 video, decodes it, and applies spatial scaling and grey scaling on every frame. Finally, it encodes the video again, using an MPEG-2 encoder, multiplexes the video and audio content, and forwards it to the PDA. As the video adaptation engine is not capable of preparing the multimedia content for the car radio, it forwards the stream to the content selection system which removes the video objects from the content and forwards it to the car radio.

4. A Component Based Multimedia Adaptation Framework

In this section, we introduce a component based multimedia adaptation framework like shown in Figure 1. The main objective of the proposed framework is to shift the emphasis from *programming* multimedia adaptation systems to *composing* the required steps from existing components. In multimedia adaptation systems, the multimedia resource is typically stored on the server in exactly one defined format and is transformed according to the client capabilities, usage environment characteristics, and user preferences before it is being streamed over the network. Thus, the server has to apply a set of adaptation components to the original stream. Consequently, we developed a knowledge based adaptation *decision taking engine* that allows us to compute adaptation plans for the components. An adaptation plan contains the required software components and their execution order. To leave the framework open and extensible, declarative capability descriptions are used for the plan generation process. For interoperability reasons, only standardized metadata terms and tags can be used for content, capabilities, and preferences descriptions.

Required Metadata for the Adaptation Framework. The adaptation engine requires information about the original multimedia resource it has to adapt. This is basically structural information like, e.g., the video's or audio's encoding format, spatial resolution of the video, color depth, average bitrate, etc. This information is covered by the *MPEG-7 Media Information* tools [8]. Furthermore, the adaptation engine has to know the target environment where the multimedia content is rendered like, e.g., terminal capabilities, network and user characteristics, etc. This information is obtained from the *MPEG-21 Digital Item Adaptation* tools [9]. Finally, the missing piece for our framework is a description of the software components that actually perform the individual adaptation steps. Therefore, we introduce a programming language independent component description which is expressive enough to capture the semantics of the adaptation components and particularly, the effects on the whole adaptation process. Furthermore, when a new adaptation component is assigned to the framework, no implementation effort has to be spent for integration. It suffices to make a proper component description available for the system. These descriptions are essential for the adaptation decision taking engine.

Adaptation Decision Taking Engine. The adaptation *decision taking engine* is responsible for selecting required adaptation components from the set of available components for the adaptation process. For example, in Figure 1 different content selection components like, e.g., audio-, video-, news content selection, etc., are available. The content selection system only picks up the audio selector for the adaptation process because the others are not needed to fulfill the user environment constraints. The output of the decision taking engine is an *adaptation plan*. Therefore, in our framework, we view adaptation as a typical state space planning problem [2]: A state space planner finds a sequence of actions for an initial state to reach a goal state. In our context, actions are adaptation operations contained in software components which are applied on the original multimedia resource to match the terminal capabilities and user preferences. The start and goal states of the planner are expressed as logical facts that use MPEG-standardized vocabulary. The actions of a STRIPS-style planner [7] are described by a set of preconditions and effects. This form of representation is simple but expressive enough for various problem domains and is today also used in the context of Semantic Web Services [14] to describe the semantics of the services.

In the following example we show how the adaptation decision taking engine creates an adaptation plan, relying on the above mentioned descriptions. Note that for the sake of readability, we use an informal notation for the metadata rather than the internal XML representation. Table 1 shows relevant fragments of the content and usage envi-

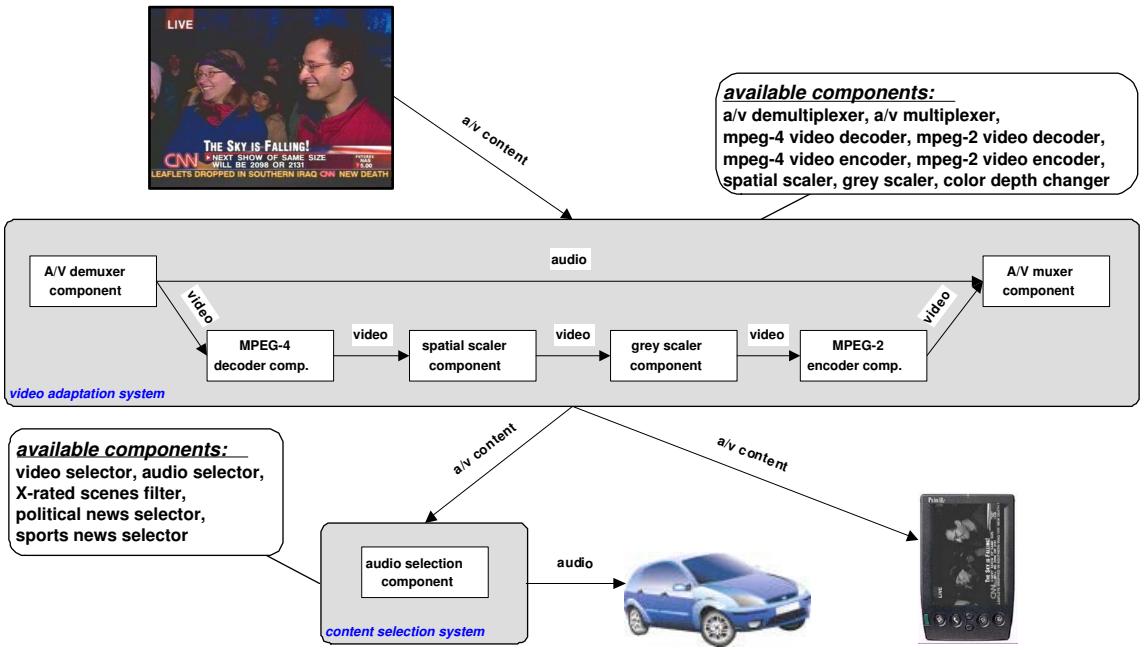

Figure 1. Component based UMA

content descr.	*usage env. descr.*
video codec: `MPEG-4`	video codec: `MPEG-2`
width: `640`	horizontal `320`
height: `480`	vertical: `240`
colordomain: `color`	color capable: `false`

Table 1. Content and usage environment descriptions

ronment descriptions. It can be seen that the original resource is a color video with 640x480 resolution, encoded using MPEG-4. The usage environment, however, does not support color display, the maximum resolution is 320x240, and only the MPEG-2 codec is available for decoding. Thus, adaptation has to be performed. Note that content description and usage environment description are using different terms, e.g., width (MPEG-7) vs. horizontal (MPEG-21).

Table 2 shows the component description of a spatial scaler. The description contains the *name* of the component (e.g., Java class name, C library name, etc.) and a list of *operations* (methods, functions). Each operation has an optional list of *parameters*, *preconditions*, and *effects*. A parameter consists of a *kind* (inout or in) which indicates whether it is a call-by-reference or a call-by-value parameter. Furthermore, a parameter holds a data *type*, and a *name*. We also introduced an anonymous parameter called *return* which represents the return type of an operation.

Preconditions and effects describe the semantics of an

component: SpatialScaler		
operation: spatialScale		
input_parameters:		
kind	*type*	*name*
`in`	`InputStream`	`inFrame`
`inout`	`OutputStream`	`outFrame`
`in`	`int`	`oldWidth`
`in`	`int`	`oldHeight`
`in`	`int`	`newWidth`
`in`	`int`	`newHeight`
`return`	`bool`	
preconditions:		
yuvImage(inFrame)		
width(oldWidth)		
height(oldHeight)		
effects:		
yuvImage(outFrame)		
width(newWidth)		
height(newHeight)		
horizontal(newWidth)		
vertical(newHeight)		

Table 2. Component description

operation and are expressed with logical facts. If all the preconditions of an operation are fulfilled, the operation is added to the adaptation plan. Thus, the `spatialScale` operation from Table 2 requires a decoded YUV image and specified width and height to be accepted for the adaptation plan. `Width` and `height` are defined by MPEG-7 con-

adaptation plan
`decode("http://path/to/res.mp4v", st1)`
`spatialScale(st1,st2,640,480,320,240)`
`greyScale(st2, st3)`
`encode(st3, Codec.MPEG2V)`

Table 3. Example adaptation plan

tent descriptions; the fact `yuvImage` may be defined by another component like, for instance, a video decoder.

Effects describe the consequences of the operation. One effect of the mentioned example is that the image has a new width. Thus, the MPEG-7 descriptor `width` is updated with the value `newWidth`. As a consequence, not only the multimedia resource is adapted but also the metadata for the resource. Furthermore, the MPEG-21 descriptor `horizontal` is set to `newWidth` which means that one goal of the terminal capabilities is fulfilled. Note that, when using this form of action description, the MPEG-7 descriptors are mapped to MPEG-21 usage environment terms. Furthermore, when using such a knowledge representation scheme, the planner does not require changes if new terms are introduced.

Having all this information, the adaptation decision taking engine can produce an adaptation plan like shown in Table 3. In this example, st_i is the input (Java `InputStream`) as well as the output (`OutputStream`) of the adaptation methods. Thus, the decoded video is the output of the `decode` method (st1) which is directed to the input of the `spatialScale` method (st2). The `spatialScale` method makes the scaled frame available at its `OutputStream` st3 and so forth.

5. Implementation

We have implemented a Java prototype of the proposed framework. Currently, our adaptation engine is placed directly on the server node where the multimedia content is stored. Note that it can easily be shifted to any other node in the network like, e.g., a proxy server or multimedia gateway.

In Section 3, we mentioned that too few "real" multimedia adaptation components are available today. To overcome this issue, we implemented Java wrapper components for the FFMPEG[2] multimedia library. Currently, we have single components for demultiplexing and multiplexing multimedia content. We also support decoding and encoding components for a variety of audio and video formats. Furthermore, we wrote wrapper components for the ImageMagick library[3] which are capable of doing still image manipulations such as greyscaling, resizing, rotating, cropping, etc. These adaptation components are regular Java class files which are stored somewhere in the classpath of the system. With the current collection of components, we can perform multimedia adaptation based on transcoding.

The architecture of our framework is illustrated in Figure 2. One can observe that each multimedia adaptation component holds a separate description as outlined in Table 2 which is placed on a defined location in the system. In our framework, we assume that the multimedia content server provides a multimedia resource and a proper MPEG-7 description. Furthermore, the client has to hold an MPEG-21 usage environment description.

If the adaptation engine receives a client request for a multimedia resource, it first verifies the availability of the desired resource and then asks the client to send its MPEG-21 usage environment descriptions. After successful reception, the adaptation decision taking engine sets up the problem space for the planner. We are using a light-weight state space planner, implemented in Prolog [2]. The setup process of the planner basically translates the different descriptions to Prolog facts. The MPEG-7 metadata are transformed to represent the *initial state*, MPEG-21 usage environment descriptions are converted to facts for the *goal state*, and the operations of the components' capability descriptions are translated to planning *actions*.

After successfully set-up of the planning problem, the adaptation decision taking engine is solving the problem. To invoke the Prolog planner from the Java code, we are using the Logic Server API from Amzi[4]. The result of the planning process is a textual representation of the adaptation steps that have to be performed on the original multimedia resource. The decision taking engine then forwards the resulting adaptation plan to the adaptation engine which takes over control.

The adaptation engine's job is to read the original multimedia resource, apply all adaptation steps from the plan to the resource, and forward the adapted resource to the client. Note that the adaptation engine works in streaming fashion, i.e., as soon as the first frames are readily adapted they are forwarded to the client. This behavior makes the system suitable for performing real-time adaptation.

As one of the main objectives of the proposed framework is to minimize implementation effort when new components with new functionality are added to the system. In standard frameworks, specific software interfaces have to be implemented manually. In our case, the adaptation engine is aware of the components descriptions, where also the parameters of all operations are described. Based on this information, the actual operation invocation is performed by dynamic class loading.

2 FFMPEG is available under http://ffmpeg.sf.net

3 ImageMagick is available under http://www.imagemagick.org

4 Amzi! web page: http://www.amzi.com

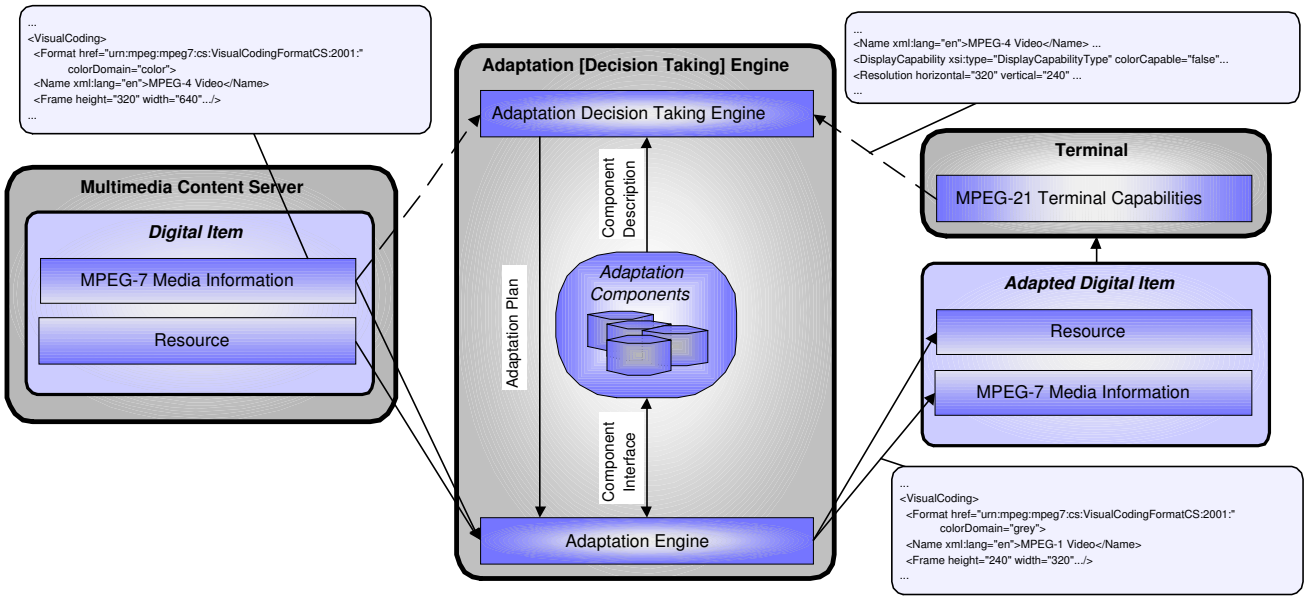

Figure 2. Architecture

Current Limitations and Possible Improvements. From the perspective of search complexity, we know that planning is in general NP-complete. Nevertheless, our experience shows that the lengths of the required adaptation plans are limited to typically five to ten steps. In our test scenario with a non-optimized implementation, the search for an adaptation plan takes less than one second. Note that 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. However, good improvements can be achieved when techniques like plan caching are introduced, where for similar client requests already pre-computed plans can be reused.

Regarding optimization of the adaptation process, however, the incorporation of additional planning heuristics must be considered in the future. If, for instance, the dimension of a video has to be reduced and greyscaling has to be performed, it is obviously better to reduce the dimension first because then the greyscaling component can operate on a smaller image.

6. Related Work

Describing the capabilities of a software component, i.e., its specification, in a formal manner is not new. In the requirements engineering phase, such specifications can for instance be used to (partially) automate the software reuse process [21] [6]. In these approaches, the functionality and behavior of the components of an existing software library are explicitly described using a formal language like Z [5] or in terms of, e.g., input-output pairs.

Another field which relies on component description is component based software engineering (CBSE). CBSE is targeted at composing applications with plug and play software components [15]. Each component is described with properties and behavior by which it can be controlled and interact with other components. Based on this description, new software can be rapidly assembled. ASL (Architecture Specification Language) [4] is a specification language which includes a behavior specification part based on pre- and post-conditions for operations, as well as other extensions supporting the description of software architectures.

Our work differs from these approach with respect to the possibility of automated software construction, i.e., assembly of function calls based on desired behavior and effects. First, such automation is possible because both the specification technique based on predicate logic as well as the allowed terms have a well-defined, precise semantics, which can be automatically processed. Second, the application domain is somehow restricted for a certain class of problems, whereas the above-mentioned approaches aim at general software engineering problems.

In general, our approach can also be seen as a step towards intelligent *multimedia web services* [10]. In particular, it is related to the promising *Semantic* Web Service initiative [5]. In [14], for instance, a scenario is sketched, where an intelligent travel planner uses a set of semantically an-

5 http://www.daml.org/services/

notated web services to compose a suitable travel arrangement (hotel reservations, flights etc.). The adaptation problem described here is in fact a potential real-world application of Semantic Web Service technology, as the notational tools and technologies like OWL-S or WSDL can be directly used to describe our application problem. Modeling the functional behavior of tools/services on pre-conditions and effects is also the standard in the field of Semantic Web Services. Establishing a shared ontology (i.e., vocabulary, terms, and semantics) for the application domain is in fact one of the hardest challenges for successful deployment of real-world Semantic Web Services. In our context, however, we have the advantage that such an ontology for the domain already exists with the help of the MPEG standard documents.

A similar approach to our work with respect to planning can be found in [11]. They use a component based framework in wide area network applications to dynamically adapt to variations in resource availability and client demand. The realtime selection of components is a crucial part of this work. Similar to our proposed framework, planning techniques are used for building adaptation plans.

7. Conclusion

We have presented a framework which is capable of composing multiple isolated adaptation components to build a powerful adaptation system. Our work is based on semantic description of software components which are exploited by a classical state space planner. Furthermore, we are mapping MPEG-7 and MPEG-21 declarations to logical facts to represent the initial state and goal state of the planner. This knowledge based approach enables us to compute adaptation plans independent of specific component implementations. Our work showed that a component based software development approach and declarative behavior specifications are valuable means when developing extensible multimedia systems.

References

[1] M. K. Asadi and J.-C. Dufourd. Multimedia Adaptation by Transmoding in MPEG-21. In *5th International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS 2004*, April 2004.

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

[3] A. W. Brown and K. Short. On components and objects: The foundations of component-based development. In *5th International Symposium on Assessment of Software Tools*, June 1997.

[4] D. Bryan. Exactness and Clarity in a Component-Based Specification Language. In *Object-Oriented Behavior Specification*, 1996.

[5] A. Diller. *Z: An Introduction To Formal Methods*. O'Reilly, 1996.

[6] F. Feiks and D. Hemer. Specification Matching of Object-oriented Components. In *Proceedings of the First International Conference on Software Engineering and Formal Methods (SEFM'03)*, 2003.

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

[8] ISO/IEC 15938-5:2003. Information Technology - Multimedia Content Description Interface - Part 5: Multimedia Description Schemes. 2003.

[9] ISO/IEC 21000-7:2004. Information Technology - Multimedia Framework - Part 7: Digital Item Adaptation. 2004.

[10] D. Jannach, K. Leopold, C. Timmerer, and H. Hellwagner. Toward Semantic Web Services for Multimedia Adaptation. *Poceedings of the Fifth International Conference on Web Information Systems Engineering*, November 2004.

[11] T. Kichkaylo, A. Ivan, and V. Karamcheti. Constrained Component Deployment in Wide-Area Networks Using AI Planning Techniques. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, 2003.

[12] K. Leopold, H. Hellwagner, and M. Kropfberger. QCTVA - quality controlled temporal video adaptation. *Proceedings of the SPIE International Symposium ITCom 2003 on Internet Multimedia Management Systems IV*, 5242, September 2003.

[13] J. M. Martinez. Overview of the MPEG-7 Standard. *ISO/IEC JTC1/SC29/WG11 N4031*, March 2001.

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

[15] J. Q. Ning. Component-Based Software Engineering (CBSE). In *5th International Symposium on Assessment of Software Tool (SAST '97)*, pages 34–43, 1997.

[16] F. Pereira and I. Burnett. Universal multimedia experiences for tomorrow. *IEEE Signal Processing Magazine*, March 2003.

[17] F. Pereira and T. Ebrahimi, editors. *The MPEG-4 Book*. Prentice Hall PTR, 2002.

[18] P. Schojer, L. Böszörmenyi, H. Hellwagner, B. Penz, and S. Podlipnig. Architecture of a Quality Based Intelligent Proxy (QBIX) for MPEG-4 Videos. *WWW2003*, May 2003.

[19] C. Timmerer, G. Panis, H. Kosch, J. Heuer, H. Hellwagner, and A. Hutter. Coding format independent multimedia content adaptation using XML. In *Proceedings of SPIE International Symposium ITCom 2003 on Internet Multimedia Management Systems IV, Vol. 5242*, September 2003.

[20] A. Vetro. MPEG-21 Digital Item Adaptation: Enabling Universal Multimedia Access. In J. R. Smith, editor, *IEEE MultiMedia*, pages 84–87, January 2004.

[21] A. M. Zaremski and J. M. Wing. Specification Matching of Software Components. In *ACM Transactions on Software Engineering and Methodology, 6(4)*, pages 333–369, 1997.