THE QUEEN'S AWARDS
FOR ENTERPRISE
2001

BBC

R&D White Paper

WHP 133

April 2006

Improving workflow in practice for low-cost programme-making using MXF & AAF file formats

P.N. Tudor *and* S.H. Cunningham

Research & Development
BRITISH BROADCASTING CORPORATION

BBC Research & Development
White Paper WHP 133

Improving workflow in practice for low-cost programme-making using MXF & AAF file formats

P.N. Tudor and S.H. Cunningham

Abstract

Tapeless acquisition and efficient post-production workflows are an aspiration for many production teams, because of the potential savings in time spent on non-creative tasks. The marketplace provides solutions using optical disc and memory-card based acquisition devices, direct ingest of live feeds into servers, and server-based editing networks. Increasingly, the interoperability within these solutions is based on open file formats for material and associated metadata (e.g. MXF and AAF).

While the functionality of new systems may be excellent, not every production team has the necessary budget. Where production budgets are low, open file formats provide a new opportunity to integrate commodity IT equipment with existing production tools, to achieve some of the workflow benefits previously only available at high cost.

This paper describes a scratch-built multi-channel ingest system built by BBC R&D, based on off-the-shelf components and open-source software. Using standard file formats, the system has been integrated with existing commercial editing tools to enable an improved tapeless workflow, at low cost, for a BBC production team.

The relevant enabling file formats are examined and a number of design aspects are discussed. This project is an example of how open formats and open source software give users the means to innovate directly to develop improved ways of working.

This paper was originally given in Las Vegas at NAB 2006 on 27th April 2006.

IMPROVING WORKFLOW IN PRACTICE FOR LOW-COST PROGRAMME-MAKING USING MXF & AAF FILE FORMATS

P.N. Tudor and S.H. Cunningham
BBC Research & Development, Tadworth, U.K.

INTRODUCTION

Tapeless acquisition and efficient post-production workflows are an aspiration for many production teams, because of the potential savings in time spent on non-creative tasks. The marketplace provides solutions using optical disc and memory-card based acquisition devices, direct ingest of live feeds into servers, and server-based editing networks. Increasingly, the interoperability within these solutions is based on open file formats for material and associated metadata (e.g. MXF and AAF).

While the functionality of new systems may be excellent, not every production team has the necessary budget. Where production budgets are low, open file formats provide a new opportunity to integrate commodity IT equipment with existing production tools, to achieve some of the workflow benefits previously only available at high cost.

This paper describes a scratch-built multi-channel ingest system built by BBC R&D, based on off-the-shelf components and open-source software. Using standard file formats, the system has been integrated with existing commercial editing tools to enable an improved tapeless workflow, at low cost, for a BBC production team.

The relevant enabling file formats are examined and a number of design aspects are discussed. This project is an example of how open formats and open source software give users the means to innovate directly to develop improved ways of working.

BAMZOOKi

BAMZOOKi is a childen's television programme recorded at BBC studios in Elstree (North London). The programme combines a studio-based set (with presenter and teams of children) and computer generated creatures known as Zooks. The Zooks are designed by the children at home before the show using Zook building software downloaded from the BBC Web site [1]. During the show, the children's Zooks compete against virtual environments and against each other. It is shot using multiple cameras.

BBC R&D has been involved with this programme since its initial development. It makes extensive use of the BBC R&D "free-d" camera tracking system, allowing the virtual viewpoint of the computer generated images to follow the physical viewpoint of the cameras.

Previous work with BAMZOOKi in October 2004 had introduced a gallery logging system for use by the Production Assistant (PA) to record notes and timecodes for all the "takes" recorded in the studio. This information was transferred directly into the edit system using an ALE file (Avid Log Exchange). The ALE file tells the edit system which sections of which tapes need to be ingested, and includes the notes taken by the PA in the gallery. This system worked well and saved considerable time in post-production.

MOVING TO TAPELESS RECORDING

To improve the workflow further, the next development was to enhance the system so that as well as logging what was being recorded, it also recorded the video and audio feeds from the studio. With this capability, the recorded video and audio material, together with the PA's notes, can be made available directly in the edit system, avoiding tapes entirely. Eliminating the need to ingest into the edit system from tape can saves hours of time when preparing to edit the programme together.

The block diagram of the logging and recording system used for the BAMZOOKi programme is shown in Figure 1. The recording system, known as the "Ingex" ingest server, is a PC with multiple SDI video I/O cards. It captures several feeds of video and audio, processes them into a range of different formats and stores the results to disk. The gallery logger allows entry of logging metadata and controls the recording. The uncompressed stored material is processed into MXF files (of various kinds), which can be edited directly by the online Avid Media Composer Adrenaline system used for BAMZOOKi post-production.

The system also provides review copies of the recorded material, generated from the MPEG-2

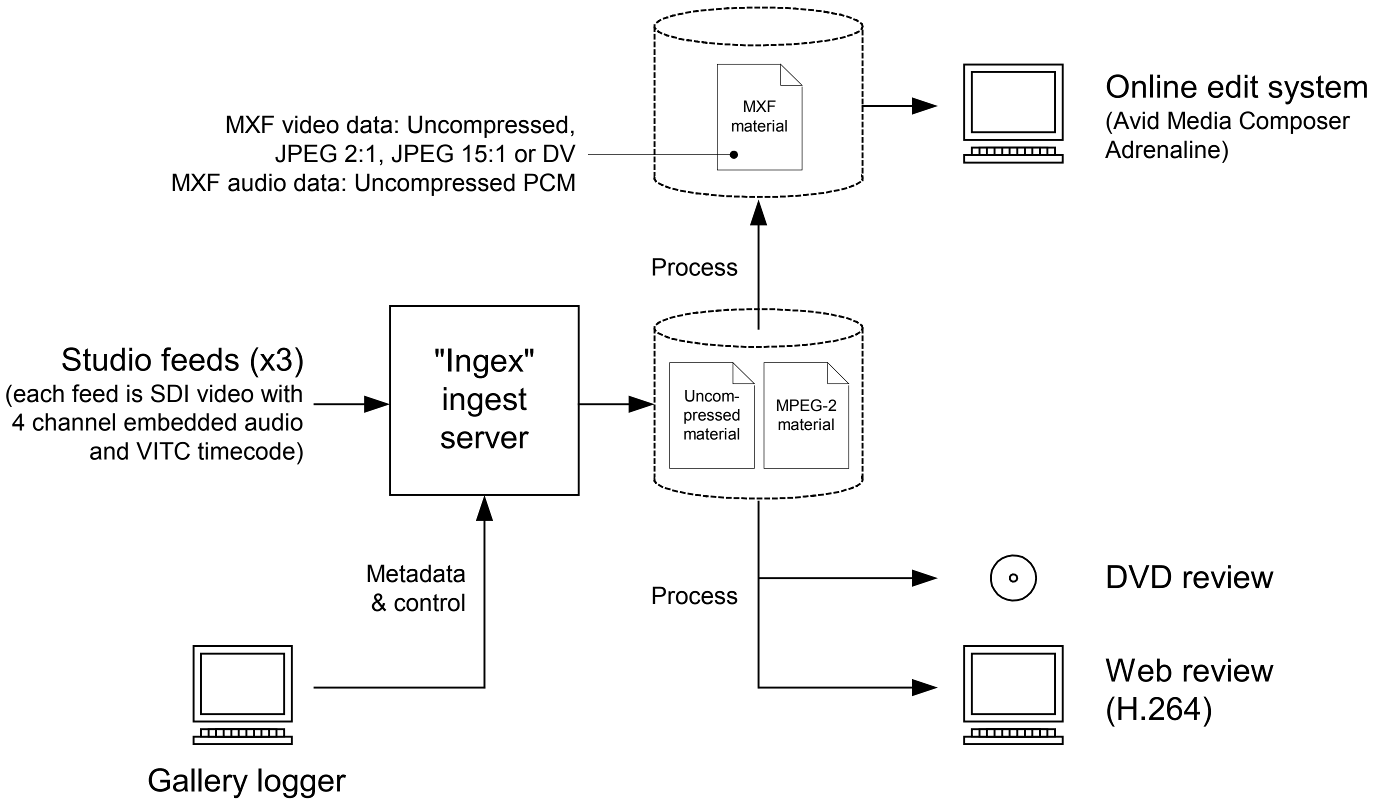

MXF material
MXF video data: Uncompressed, JPEG 2:1, JPEG 15:1 or DV
MXF audio data: Uncompressed PCM
Online edit system
(Avid Media Composer Adrenaline)
Process
Studio feeds (x3)
(each feed is SDI video with 4 channel embedded audio and VITC timecode)
"Ingex" ingest server
Uncom-pressed material
MPEG-2 material
Metadata & control
Process
DVD review
Web review (H.264)
Gallery logger

Figure 1. Logging and recording system used for the BAMZOOKi programme

material stored during the recording. These are available on DVDs and via a Web page. The DVDs of the studio recordings are generated automatically, so that the Director and Producer can take these away and review the material before going into the edit suite. The DVD creation software automatically organises the takes into DVD chapters with convenient menus and provides a "quad split" arrangement of the three video feeds to assist in selecting between multiple cameras. The Web-based review method is an updating Web page with links to H.264 proxies generated as each take is recorded.

On-going work is looking at delivering proxy versions of the studio recordings to portable media players, for self-contained offline viewing. This work includes determining suitable formats and bit rates to support the wide range of available devices, and investigation of different delivery methods. Particularly promising is the "podcasting" approach: an RSS feed is generated containing the logging information and links to the proxy content. The media player is then synchronised either indirectly by means of software such as Apple iTunes, or directly if the device supports wireless HTTP access (for example via a WiFi or 3G connection).

INSIDE THE INGEST SERVER

Hardware and Operating System

To satisfy the requirements of real-time uncompressed video storage and MPEG-2 encoding, the following commodity IT components were used to build the ingest server:

- 2 x dual core AMD Opteron 275 CPUs (2.2 GHz)
- 4 GB RAM
- Tyan motherboard with 4 x SATA-2 ports
- 4 x 500 GB Hitachi SATA-2 disk drives
- 3 x DVS SDStationOEM SDI I/O PCI cards

The total hardware component cost for the ingest server was US$ 10 000.

The operating system used was SUSE Linux 9.3, running the SMP kernel 2.6.11.4.

When recording three feeds with this configuration, the average CPU loading was 45% and sustained disk activity was 25% of maximum.

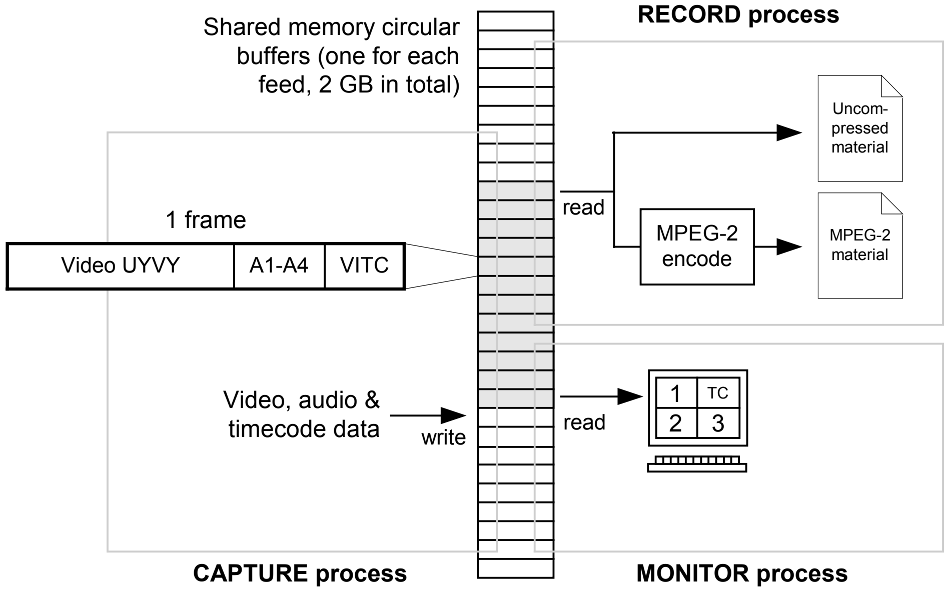


Figure 2. Real-time processes and shared memory within the Ingex ingest server

Software

Figure 2 shows the real-time processes running within the ingest server: CAPTURE, RECORD and (optionally) MONITOR. These are now described.

CAPTURE process

This is a background process started at boot time and runs permanently, copying frames of video and audio directly from the SDI input hardware into a circular buffer using DMA transfers (which are efficient because the CPU is bypassed). The VITC timecode for each frame is stored with each frame (e.g. to enable frame-accurate quad-split arrangement). Status and control variables are stored in a separate structure in shared memory. Since multiple threads may be reading or writing the control variables at any time, mutual exclusion locks (mutexes) are used to prevent conflicts.

RECORD process

This is a CORBA [2] server controlled by requests from the gallery logger. On receipt of a START request multiple threads are created to perform the following tasks:

- for each video source, write uncompressed video and audio to disk
- for each video source, encode to MPEG-2 long GOP video and layer 2 audio program stream
- synchronise the three video sources, down convert, arrange as quad-split mix, and encode to MPEG-2 long GOP video and layer 2 audio program stream
- manage all encoding threads

Recording and encoding continues until a STOP request is received. This causes the recording length to be signalled to all running threads, which continue to record and encode until the specified number of frames have been completed.

MONITOR process

This is a simple monitoring application that displays incoming video on the PC video display. It reads current frames from a circular buffer and displays them in a window. In addition, video from all three video inputs may be displayed in a quad-split arrangement.

Generating MXF files

The processing into MXF formats for use by an edit system (or other formats) begins upon receipt of a PUBLISH request from the gallery logger. The PUBLISH request indicates that the PA has entered all the logging metadata to their satisfaction. Depending on the format, metadata can be either burnt in vision or stored in the header of MXF files. The MXF formats are generated from the uncompressed stored material.

DATA FORMATS FOR INTEROPERABILITY WITH EDIT SYSTEMS

Material

MXF (Material Exchange Format) is becoming a widely supported standardised file format for material. It was developed by the Pro-MPEG Forum, the AAF Association, the EBU and others and is standardised by the SMPTE (SMPTE 377M [3] etc.)

A particular form of MXF, known as Operational Pattern (OP) Atom (SMPTE 390M [4]), is well suited for material intended for use by an edit system. Each track is stored as a separate file (so an edit system does not have to read tracks it doesn't require), the material layout is tightly defined for high-performance access and an index table is always provided.

The Avid Media Composer provides support for MXF OP-Atom as a native file format. MXF OP-Atom files, created by external applications (such as the BBC R&D ingest server) can be directly accessed without any processing delay by the Media Composer, if they are located in an Avid media directory. Furthermore, by using the "alldrives 1" Avid console command to enable access to all network drives, the MXF files may reside on generic networked storage and still be directly accessed by the Media Composer.

The MXF essence mappings implemented by the Ingex ingest server at the current time are:

Video
- SMPTE 384M Uncompressed [5]
- Avid JPEG 2:1
- Avid JPEG 15:1
- SMPTE 383M DV [6]

Audio
- SMPTE 382M Uncompressed PCM [7]

Post-production for BAMZOOKi makes heavy use of graphics, effects, keying and compositing. The production deemed 50 Mbit/s video codecs to have insufficient quality to be used as the online editing format. Using uncompressed video data, on the other hand, would have had too large an impact on storage costs. JPEG 2:1 (approx. 70 Mbit/s) was chosen as providing the best fit.

The move to tapeless recording provided an unexpected benefit for image quality—encoding directly from uncompressed video data to JPEG 2:1 gave a higher quality (measured by PSNR) than the traditional workflow of recording the video to Digibeta first, then recoding as JPEG 2:1. Direct encoding of uncompressed video to JPEG 2:1 gave PSNR values around 4 dB higher than recording on Digibeta and recoding to JPEG 2:1 (for the same final JPEG 2:1 bit rate).

The MXF OP-Atom files are created using the AAF Association's [8] open source toolkit [9], which (in its latest version) implements the MXF file format.

Metadata

To transfer the logging metadata into the Avid Media Composer, two approaches are available:

1. Use an ALE (Avid Log Exchange) file to import the metadata and relink to the MXF material using the tape name and timecodes.
2. Use an AAF (Advanced Authoring Format) file to import the metadata, which then links automatically to the MXF material using the unique material identifiers (UMIDs) contained in the AAF and MXF files.

ALE was developed by Avid and is a simple ASCII file format for logging metadata. ALE files specify sections of source tapes (by tape name and timecode) and logging metadata (e.g. description and comments).

AAF [10] is a standard file format for transferring many kinds of metadata between authoring tools. AAF has a comprehensive data model specifically designed to meet the complex requirements of real-world audio, video and film authoring. AAF files can be created using the AAF Association's open source toolkit [9]. The AAF data model is shared, in part, by MXF, which means that mapping metadata fields between AAF and MXF is simple. AAF is promoted by the AAF Association, a non-profit organisation of manufacturers, user organisations and developers.

Table 1 shows the metadata fields that need to be transferred from the ingest server to the edit system, and indicates whether each is carried or not in ALE, MXF and AAF.

The workflow for bringing the material and metadata into the Avid Media Composer is as follows:

1. Mount the file system containing the MXF OP-Atom material files onto the Avid Media Composer (use console command "alldrives 1" to access generic networked storage)
2. The Media Composer refreshes its media database after scanning the newly mounted file system

Then, if using ALE files:

3. Import the ALE file into the Avid bin

4. The metadata appears, but the media is offline
5. To bring the clips online, relink to the MXF files, matching on tape name and timecode.

Metadata	In ALE	In MXF	In AAF
Clip name	Y	Y	Y
Tape name	Y	Y*	Y
Tape start timecode	Y	Y*	Y
Duration	Y	Y	Y
Track number	Y	Y	Y
Description	Y	Y	Y
Comments	Y	Y	Y
Marked IN/OUT points in clip	Y	N	Y
Clip UMID	N	Y	Y
Kind of video or audio coding	N	Y	Y

Table 1. Metadata support in ALE, MXF and AAF (metadata used for relinking to MXF material highlighted)

For this to work, the MXF files need to contain metadata about the source tapes from which they came (see * fields in Table 1). This is not a standard feature of MXF, although it is supported by Avid through their use of the AAF toolkit for MXF.

If using AAF files:

3. Import the AAF file into the Avid bin
4. The metadata appears and the Media Composer links to the MXF material files directly (matching on common clip UMIDs in the AAF and MXF files).

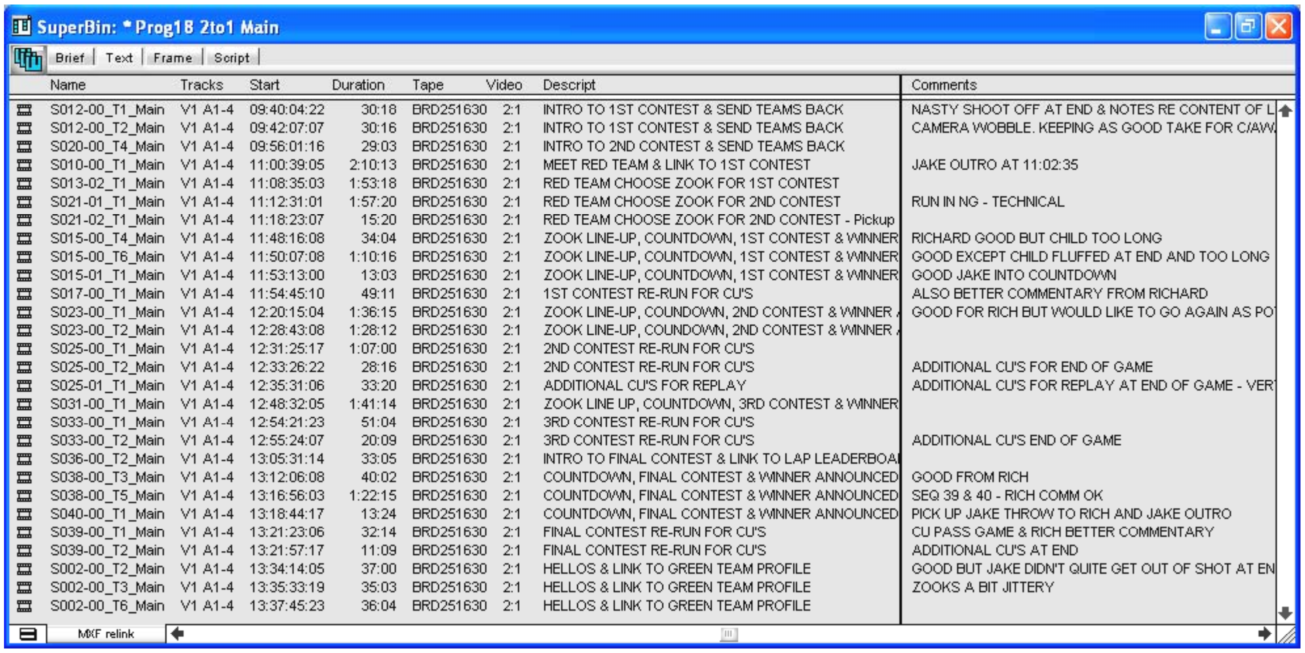
Of the two approaches, the AAF/MXF route is preferable. It avoids the relinking step based on tape names (which may be problematic if the studio recordings were not made on tape, or if a tape name has been incorrectly logged) and it does not require the MXF file to contain source tape metadata, which is a non-standard feature.

Figure 3 shows the clip metadata in the Media Composer user interface. All fields are either generated by the Ingex ingest server automatically or entered by the PA at the time of studio recording. No manual rekeying occurs.

DESIGN ASPECTS

Video capture hardware abstraction layer

The interface layer between the capture of video and its storage and processing consists of a shared memory circular buffer and control structure (see Figure 2). When a new hardware capture card is to be used in the ingest server the only piece of software to be written is the code that communicates with the



Name	Tracks	Start	Duration	Tape	Video	Descript	Comments
S012-00_T1_Main	V1 A1-4	09:40:04:22	30:18	BRD251630	2:1	INTRO TO 1ST CONTEST & SEND TEAMS BACK	NASTY SHOOT OFF AT END & NOTES RE CONTENT OF L
S012-00_T2_Main	V1 A1-4	09:42:07:07	30:16	BRD251630	2:1	INTRO TO 1ST CONTEST & SEND TEAMS BACK	CAMERA WOBBLE. KEEPING AS GOOD TAKE FOR C/AW.
S020-00_T4_Main	V1 A1-4	09:56:01:16	29:03	BRD251630	2:1	INTRO TO 2ND CONTEST & SEND TEAMS BACK	
S010-00_T1_Main	V1 A1-4	11:00:39:05	2:10:13	BRD251630	2:1	MEET RED TEAM & LINK TO 1ST CONTEST	JAKE OUTRO AT 11:02:35
S013-02_T1_Main	V1 A1-4	11:08:35:03	1:53:18	BRD251630	2:1	RED TEAM CHOOSE ZOOK FOR 1ST CONTEST	
S021-01_T1_Main	V1 A1-4	11:12:31:01	1:57:20	BRD251630	2:1	RED TEAM CHOOSE ZOOK FOR 2ND CONTEST	RUN IN NG - TECHNICAL
S021-02_T1_Main	V1 A1-4	11:18:23:07	15:20	BRD251630	2:1	RED TEAM CHOOSE ZOOK FOR 2ND CONTEST - Pickup	
S015-00_T4_Main	V1 A1-4	11:48:16:08	34:04	BRD251630	2:1	ZOOK LINE-UP, COUNTDOWN, 1ST CONTEST & WINNER	RICHARD GOOD BUT CHILD TOO LONG
S015-00_T6_Main	V1 A1-4	11:50:07:08	1:10:16	BRD251630	2:1	ZOOK LINE-UP, COUNTDOWN, 1ST CONTEST & WINNER	GOOD EXCEPT CHILD FLUFFED AT END AND TOO LONG
S015-01_T1_Main	V1 A1-4	11:53:13:00	13:03	BRD251630	2:1	ZOOK LINE-UP, COUNTDOWN, 1ST CONTEST & WINNER	GOOD JAKE INTO COUNTDOWN
S017-00_T1_Main	V1 A1-4	11:54:45:10	49:11	BRD251630	2:1	1ST CONTEST RE-RUN FOR CU'S	ALSO BETTER COMMENTARY FROM RICHARD
S023-00_T1_Main	V1 A1-4	12:20:15:04	1:36:15	BRD251630	2:1	ZOOK LINE-UP, COUNDOWN, 2ND CONTEST & WINNER	GOOD FOR RICH BUT WOULD LIKE TO GO AGAIN AS PO
S023-00_T2_Main	V1 A1-4	12:28:43:08	1:28:12	BRD251630	2:1	ZOOK LINE-UP, COUNDOWN, 2ND CONTEST & WINNER	
S025-00_T1_Main	V1 A1-4	12:31:25:17	1:07:00	BRD251630	2:1	2ND CONTEST RE-RUN FOR CU'S	
S025-00_T2_Main	V1 A1-4	12:33:26:22	28:16	BRD251630	2:1	2ND CONTEST RE-RUN FOR CU'S	ADDITIONAL CU'S FOR END OF GAME
S025-01_T1_Main	V1 A1-4	12:35:31:06	33:20	BRD251630	2:1	ADDITIONAL CU'S FOR REPLAY	ADDITIONAL CU'S FOR REPLAY AT END OF GAME - VER
S031-00_T1_Main	V1 A1-4	12:48:32:05	1:41:14	BRD251630	2:1	ZOOK LINE UP, COUNTDOWN, 3RD CONTEST & WINNER	
S033-00_T1_Main	V1 A1-4	12:54:21:23	51:04	BRD251630	2:1	3RD CONTEST RE-RUN FOR CU'S	
S033-00_T2_Main	V1 A1-4	12:55:24:07	20:09	BRD251630	2:1	3RD CONTEST RE-RUN FOR CU'S	ADDITIONAL CU'S END OF GAME
S036-00_T2_Main	V1 A1-4	13:05:31:14	33:05	BRD251630	2:1	INTRO TO FINAL CONTEST & LINK TO LAP LEADERBOA	
S038-00_T3_Main	V1 A1-4	13:12:06:08	40:02	BRD251630	2:1	COUNTDOWN, FINAL CONTEST & WINNER ANNOUNCED	GOOD FROM RICH
S038-00_T5_Main	V1 A1-4	13:16:56:03	1:22:15	BRD251630	2:1	COUNTDOWN, FINAL CONTEST & WINNER ANNOUNCED	SEQ 39 & 40 - RICH COMM OK
S040-00_T1_Main	V1 A1-4	13:18:44:17	13:24	BRD251630	2:1	COUNTDOWN, FINAL CONTEST & WINNER ANNOUNCED	PICK UP JAKE THROW TO RICH AND JAKE OUTRO
S039-00_T1_Main	V1 A1-4	13:21:23:06	32:14	BRD251630	2:1	FINAL CONTEST RE-RUN FOR CU'S	CU PASS GAME & RICH BETTER COMMENTARY
S039-00_T2_Main	V1 A1-4	13:21:57:17	11:09	BRD251630	2:1	FINAL CONTEST RE-RUN FOR CU'S	ADDITIONAL CU'S AT END
S002-00_T2_Main	V1 A1-4	13:34:14:05	37:00	BRD251630	2:1	HELLOS & LINK TO GREEN TEAM PROFILE	GOOD BUT JAKE DIDN'T QUITE GET OUT OF SHOT AT EN
S002-00_T3_Main	V1 A1-4	13:35:33:19	35:03	BRD251630	2:1	HELLOS & LINK TO GREEN TEAM PROFILE	ZOOKS A BIT JITTERY
S002-00_T6_Main	V1 A1-4	13:37:45:23	36:04	BRD251630	2:1	HELLOS & LINK TO GREEN TEAM PROFILE	

Figure 3. Clip metadata in an Avid Media Composer bin

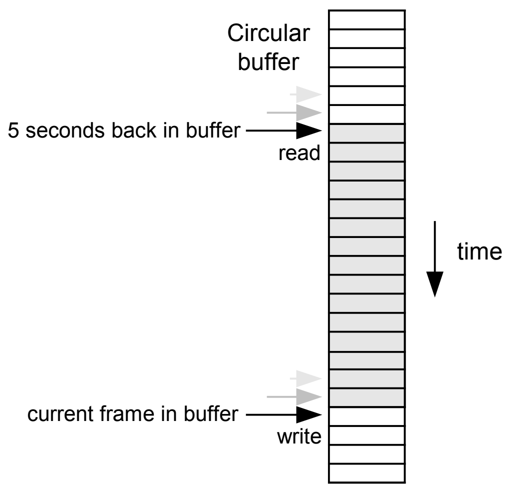


Figure 4. Buffered frames provide a 5 second pre-recording handle

hardware interface and extracts video and audio samples. Whatever approach is used to extract the video and audio, the software will write the data to the shared memory in the same arrangement so that reading processes do not need to be changed. This design allows multiple different hardware capture cards to be used in the PC simultaneously without affecting the execution of the other cooperating processes.

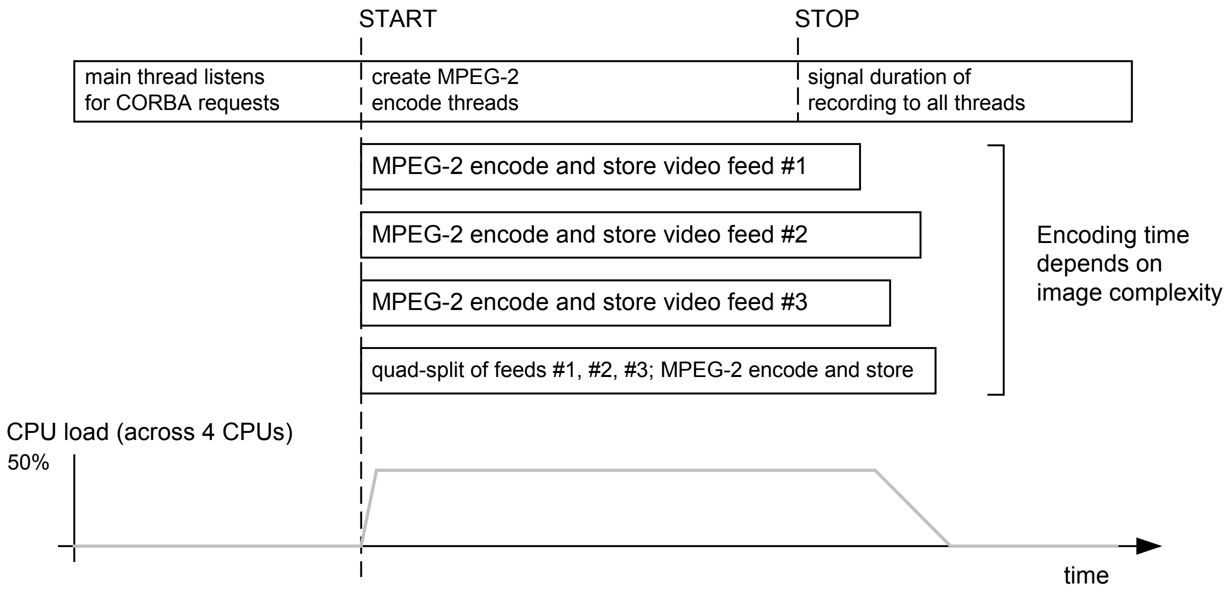
Pre-record and post-record handles

The signalling of START and STOP events is performed by the PA in the studio gallery and is intended to provide accurate logging of the take in addition to recording only the material required. However, it was observed that a suitable edit point was sometimes difficult to find with such tight "handles" before and after the take. The solution was to provide a 5 second pre-record handle, which stores the material from 5 seconds before the PA hit START. Similarly a 3 second post-record handle is stored after the STOP event.

Supporting the pre-record handle required the ingest server to constantly record the latest 5 seconds of video, as shown in Figure 4. The low cost of RAM meant that the required storage was feasible to allocate in memory. The capture process continually copies video and audio data from the SDI card into the circular buffer in shared memory. The circular buffer is managed by keeping track of a write pointer and read pointer. By managing the read and write pointers correctly, other processes can read up to 5 seconds into the past without reading a frame in the wrong order or a frame currently being written to.

Multi-threading and multi-core CPUs for real-time encoding

To meet the requirement of encoding four MPEG-2 program streams in real-time (3 feeds and a quad-split arrangement), a multi-threaded implementation was needed, along with appropriate hardware capable of parallel processing. Using two dual-core Opteron CPUs provided a suitable low-cost quad-CPU platform.



START
STOP
main thread listens for CORBA requests
create MPEG-2 encode threads
signal duration of recording to all threads
MPEG-2 encode and store video feed #1
MPEG-2 encode and store video feed #2
MPEG-2 encode and store video feed #3
quad-split of feeds #1, #2, #3; MPEG-2 encode and store
Encoding time depends on image complexity
CPU load (across 4 CPUs)
50%
time

Figure 5. Creation of MPEG-2 encoding threads and CPU loading during recording

The most intensive use of threads was in the RECORD process, as shown in Figure 5. When a START request was received, RECORD would start four MPEG-2 encoding threads—one thread for each of the video sources and one thread for the virtual quad-split source. The quad-split thread was the most involved since it needed to synchronise video and audio from three separate buffers with video arriving asynchronously, down convert the pictures with filtering for interlace into a suitable quad-split arrangement and burn metadata in-vision before encoding at 5 Mbit/s in MPEG-2 long GOP.

Complications arose from the pre-record and post-record handles since the end of one studio take can overlap with the next take by up to 7 seconds. In this scenario there are 8 MPEG-2 long GOP encode threads running concurrently and care had to be taken to prioritise the appropriate threads to avoid the possibility of dropped frames. Adding more RAM to the system allowed buffers of more than 30 seconds per video feed to be used to smooth out such peaks in CPU demand.

Use of open source software to reduce development time

Software reuse is a cornerstone of good software engineering. A key benefit is the reduction in implementation, testing and debugging time. The design of the ingest server required implementations of various video and audio encoders, cross-platform networked inter-process communication and creation of MXF files compatible with the target editing system. These requirements were satisfied using available open source software, allowing development effort to be concentrated on the specific Ingex design.

The total number of lines of source code for the Ingex ingest server is 1 760 000, with the contribution of pre-existing open source software being 1 750 000 or 99.4%.

The open source libraries used in the Ingex ingest server are shown in Table 2.

Library	Purpose	Source lines of code (SLOC)
ACE	Cross platform CORBA implementation	750 000
AAF toolkit	Read/write MXF files. Read/write AAF files.	550 000
ffmpeg	MPEG codec	240 000
libdv	DV codec	70 000
libjpeg	JPEG codec	50 000
Ingex	Ingex server application (created in the work described here, and made available as open source)	10 000

Table 2. Open Source libraries used in Ingex ingest server

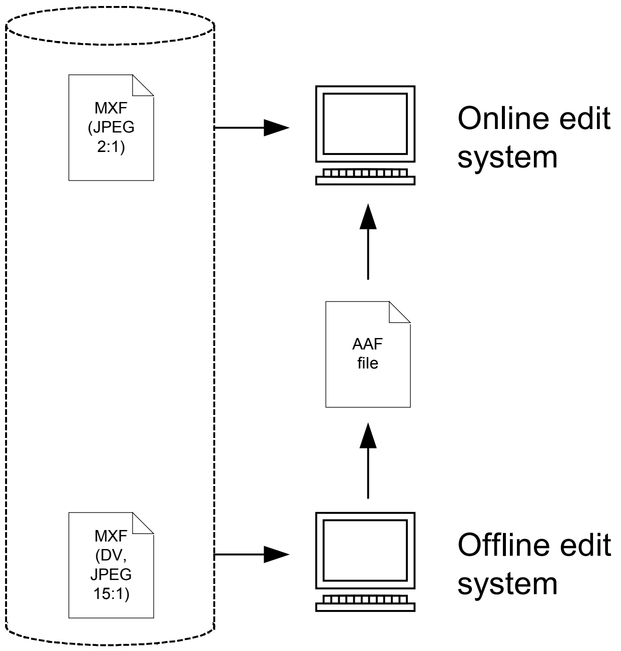


Figure 6. Transferring clip metadata and composition metadata between offline and online edit systems

The availability of such an enormous range of useful libraries, and the corresponding saving on development time, brings development of systems such as the ingest server within the capability of small engineering teams and potentially within user organisations. This gives users the possibility to experiment directly with new tools to improve workflow efficiency, and rapidly develop them based on experimental trials. Furthermore, the increasing deployment of standard file formats means that user-developed prototypes can readily interface to existing commercial production tools, such as the Avid Media Composer in this case.

So that others may duplicate the system reported here, and help develop it further, the BBC R&D Ingex software has been made available as open source software [11].

Supporting offline and online edit systems

For BAMZOOKi, the ingest server generated online quality JPEG 2:1 material files which were used directly by a single Avid Media Composer Adrenaline. However, where an offline edit is needed before the online edit, the ingest system can generate a variety of suitable offline formats (in particular, DV and JPEG 15:1).

The transfer of a basic edit from an offline system to an online system would be via an AAF file, as shown in Figure 6. The AAF file simultaneously carries the clip metadata (referencing the MXF material files) and the basic edit metadata between the edit systems.

Supporting independent extension of MXF OP-Atom partitions

For BAMZOOKi, the generation of MXF OP-Atom material files begins after the PA has completed entering the logging metadata. This means that the OP-Atom constraint that all the metadata shall be written into the header partition before the video or audio essence partition can easily be met.

However, for an environment where logging metadata is being entered during the creation of the MXF OP-Atom file, a strategy is needed for how to manage the simultaneous extension of the header and essence partitions.

A simple approach is to reserve a fixed amount of space in the header partition for metadata, in front of the essence partition. As long as the amount of header metadata does not exceed the reserved space, the metadata can be written to the header at the same time as the video or audio is written to the essence partition. If the header metadata grows beyond the reserved space, compatibility with OP-Atom would be broken if the application overflowed the metadata into the footer.

Another approach would be to store the essence in a separate file and then append it to the header once the ingest was complete, but the file copy operation would be prohibitive for all but trivial video files.

A simple, high-performance solution is offered by the File system in Userspace (FUSE) feature [12] of GNU/Linux and FreeBSD platforms. This approach configures a virtual file system which allows separate files representing the header, essence and footer partitions to be joined together exposing a single compliant MXF OP-Atom file when viewed from the mount point. The underlying separate files are written

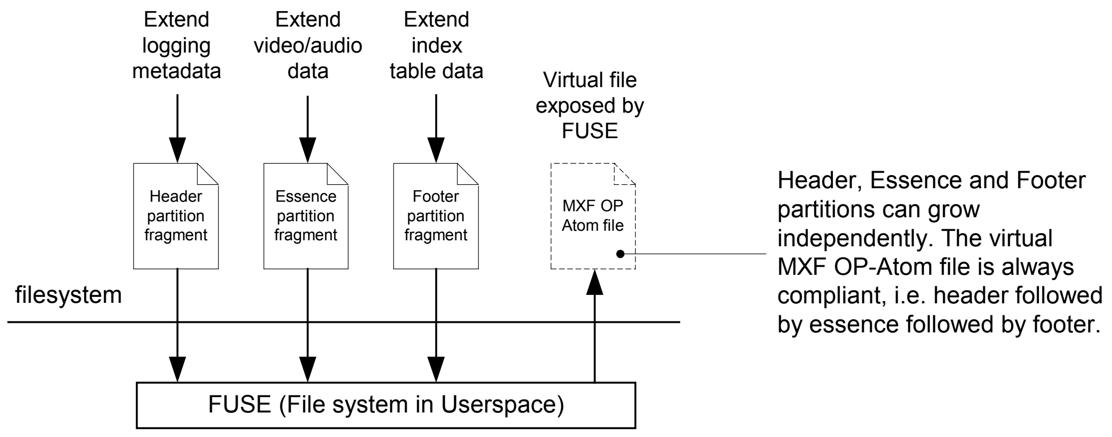

Extend logging metadata
Extend video/audio data
Extend index table data
Virtual file exposed by FUSE
Header partition fragment
Essence partition fragment
Footer partition fragment
MXF OP Atom file
Header, Essence and Footer partitions can grow independently. The virtual MXF OP-Atom file is always compliant, i.e. header followed by essence followed by footer.
filesystem
FUSE (File system in Userspace)

Figure 7. Virtual MXF OP-Atom file exposed by FUSE layer, joins separate partition files together

to a regular high-performance file system. When an edit system accesses the virtual file system it sees only a single correct OP-Atom file. The implementation of the file system is straightforward since only seek and read requests need to be caught and redirected to different positions in the partition fragment files. The performance overhead of simple calculations in this extra layer is negligible. Additionally, since the implementation lies beneath the kernel's VFS layer, all file system buffering is applied to the virtual file system so frequent requests to a given portion of the OP-Atom file may not need to invoke the FUSE implementation.

CONCLUSIONS

A gallery logging and multi-channel ingest system has been developed, based on commodity IT components and open source software. A production-ready prototype was used to record a series (20 shows) of the BBC children's programme BAMZOOKi. The hardware component cost for the ingest server including storage was US$ 10 000.

By implementing the MXF and AAF standard file formats, the recorded material and logging metadata could be made available directly to the online edit system (Avid Media Composer Adrenaline).

The improved workflow eliminated time spent ingesting from tape and made the production assistant's notes directly available in the user-interface of the online edit system.

The availability of a large amount of open source software reduced the development period of the project significantly, bringing it within the capability of a small engineering team within a user organisation. 10 000 lines of new source code were required in addition to 1 750 000 from pre-existing open source libraries. This ease of development gives users the potential to experiment very directly with new tools and systems to improve their workflow, and to rapidly develop them based on experimental trials.

The new software written for this work has been made available as open source.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the contributions of their colleagues Philip de Nier, David Kirby, John Fletcher, Matthew Marks and Peter Brightwell to the work described here. Paul Tyler is the producer of BAMZOOKi. Gerry Lindfield evaluated the generated MXF files on the Avid Media Composer, and used them to edit the latest BAMZOOKi series.

The authors would also like to thank the BBC for permission to publish this paper.

REFERENCES

[1] BBC, BAMZOOKi Web site: http://www.bbc.co.uk/cbbc/bamzooki.
[2] Object Management Group, Common Object Request Broker Architecture (CORBA) Web site: http://www.corba.org.
[3] SMPTE, Material Exchange Format (MXF) – File Format Specification, SMPTE 377M etc.
[4] SMPTE, Material Exchange Format (MXF) – Specialized Operational Pattern "Atom", SMPTE 390M.
[5] SMPTE, Material Exchange Format (MXF) – Mapping of Uncompressed Pictures into the MXF Generic Container, SMPTE 384M.
[6] SMPTE, Material Exchange Format (MXF) – Mapping DV-DIF Data to the MXF Generic Container, SMPTE 383M.
[7] SMPTE, Material Exchange Format (MXF) – Mapping AES3 and Broadcast Wave Audio into the MXF Generic Container, SMPTE 382M.
[8] AAF Association, AAF Association home page: http://www.aafassociation.org.
[9] AAF Association, AAF toolkit home page: http://aaf.sourceforge.net.
[10] AAF Association, AAF Specification: http://www.aafassociation.org/html/techinfo/index.html#aaf_specifications.
[11] BBC, Ingex project home page: http://ingex.sourceforge.net.
[12] File system in User space (FUSE) home page: http://fuse.sourceforge.net.