Understanding and Selecting a File Format for AV

Peter Bubestinger-Steindl

(peter @ ArkThis.com)

ArkThis AV-RD

Speaker notes

No notes on this slide.

Speaker notes

No notes on this slide.

Why bother? - Let’s just have:

- Best quality
- Smallest filesize
- Preserve original properties
- Fast and easy to open/use/edit
- Lasts forever
- +cherries 🍒 & ice cream 🍦on top!
- …

Speaker notes

That list actually makes sense:

They define the desired properties that you have to choose from, which ones and how well you want them implemented.

Tempting…

- Hey, it’s a standard!
- Hey, everyone’s using it!
- Hey, the “big ones” are using it!
- Hey, it’s from a major company!
- Hey, it can do *everything*!
- Hey, it’s so easy to use!
- Hey, it’s gratis!

Speaker notes

Of course all of these are valid reasons to adopt a file format, but when it comes to long-term preservation requirements and independent sustainability, they may not be sufficient, and sometimes even misleading.

And yes, sometimes choosing such a format can be “just fine” too.

At least take some time to align your needs and wishes with any format option.

Digital AV formats...

*What are **your** wishes, needs, expectations of a format?*

Speaker notes

No notes on this slide.

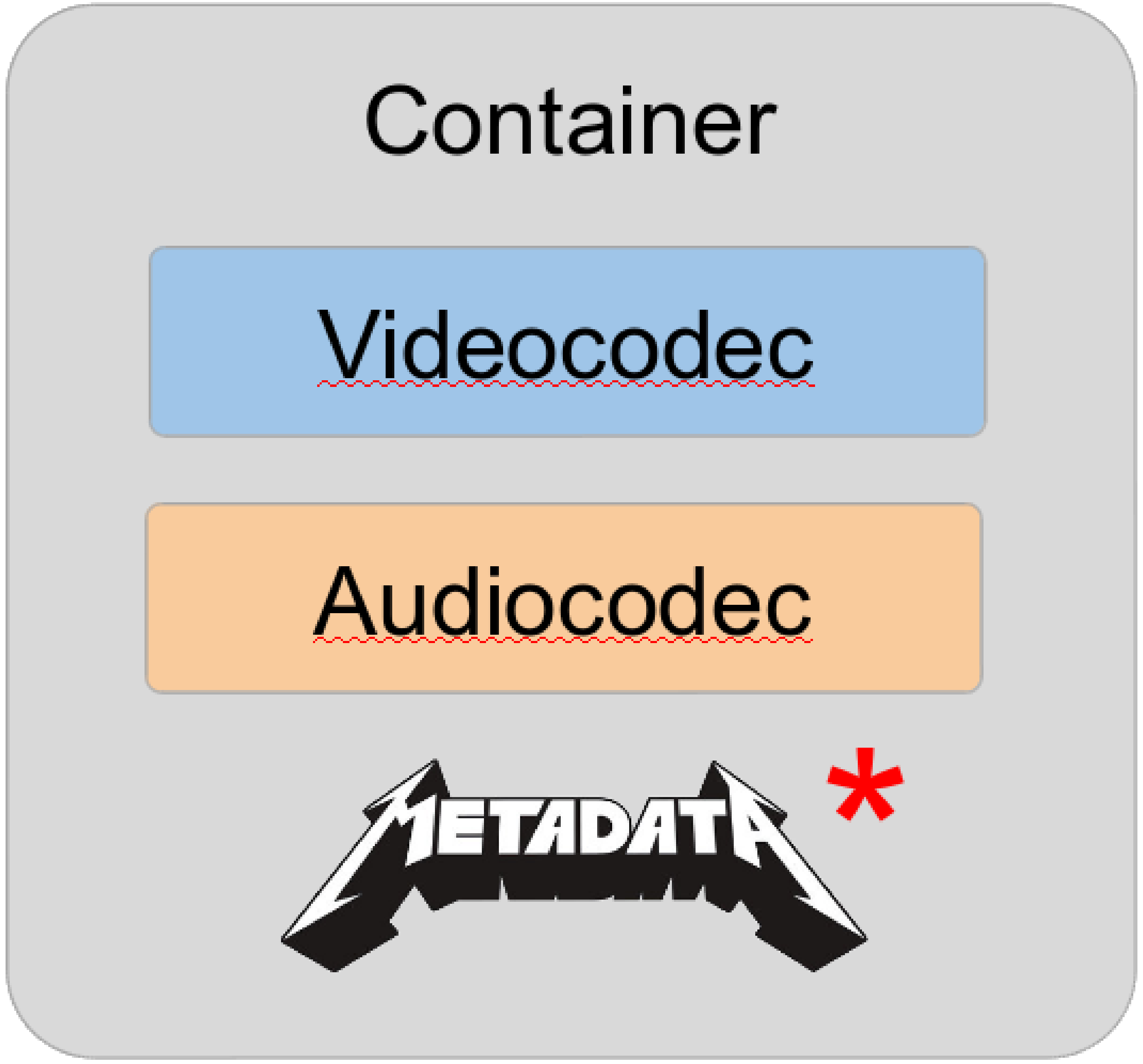
Digital AV formats...

- Which do *you* know?
- Which do *you* use?
- Which would *you* like to know more about?

Speaker notes

No notes on this slide.

Digital Video Trinity



Container

Videocodec

Audiocodec

METADATA *

Speaker notes

The uncomfortable truth. It’s more than “1 format” to choose/consider for AV.

The container format is the one you see in the filename as suffix: mov, mkv, mxf, avi, flv, 3gp, wav, etc.

A/V media files: “The” format of video files usually consists of 3 different formats:

- Container
- Video codec
- Audio codec

Optional:

Embedded metadata of all kinds:

- Descriptive
- Technical
- non time-based / time-based

Additional data streams: (usually “time-based”)

- subtitles
- timecode
- crazy special feature xy
- etc, etc, etc…

What’s a Container?

> *“A container format (informally, sometimes called a wrapper) […] is a file format that allows multiple data streams to be embedded into a single file, usually along with metadata for identifying and further detailing those streams.”*

Source: Wikipedia: Container format (computing)

Speaker notes

No notes on this slide.

What’s a Container?

Think of a regular paper folder…

- It’s a wrapper around content.
- Contains Metadata.
- Structures the content streams.

Speaker notes

Metadata: * Index (which streams are contained, etc) * Descriptive MD (title, language, etc) * as well as technical MD (fps, aspect ratio, color handling, etc) * NOTE: Some technical MD may be stored in the container and the codec/stream. This may be a blessing and a curse… Example: Aspect ratio or fps.

What’s a Codec?

> *“A codec is a device or computer program which encodes or decodes a data stream or signal.”*

Source: Wikipedia: Codec

Speaker notes

No notes on this slide.

What’s a Codec?

Think of a human language…

- It’s coded information.
- There may be dialects.
- Different people may “speak / understand” differently.

Speaker notes

No notes on this slide.

Format Naming

Triplet notation greatly helps reducing confusion:

- **H.264** / **AAC** in **MP4**
- **FFV1** / **PCM** in **MKV** (Matroska)
- **ProRes** / **PCM** in **MOV**
- **DPX** / **WAV** (PCM) in **a folder**
- etc

Speaker notes

When communicating “which video format”, please consider using a triplet-notation, like: “video/audio in container”.

It is unfortunately most common to simply quote the file suffix as “the format”.

Let’s look inside! :)

Speaker notes

No notes on this slide.

VLC / MediaInfo

Website: videolan.org/vlc

Website: mediaarea.net/MediaInfo

Speaker notes

“MediaInfo is a convenient unified display of the most relevant technical and tag data for video and audio files.”

btw: A VLC-related WARNING: There’s a major fraud out there: “www.vlc.de” - aka “VLC Plus Player”. It contains the original VLC, but with unknown - possibly malicious - additions/modifications. Stay away from it.

Characteristics / Properties

	File 1	File 2	File 3
Container	MOV	MOV	MOV
Videocodec	UYVY	H.264	XviD
Resolution	720 x 576px	1920 x 1080	640 x 480
FPS	25	24	30000/1001
-			
Audiocodec	PCM	AAC	MP3
Samplerate	48 kHz	48 kHz	44.1 kHz
Channels	Stereo	Surround 5.1	Mono

Speaker notes

This is just a random example to show that from the “outside” (=file explorer) you would only see that all files are “.mov” - whereas their actual audio/video codecs, as well as their technical properties are completely different. It therefore always makes sense to use proper tools to “look inside” (MediaInfo, etc).

Significant properties

Knowing and deciding which properties to safeguard and which are allowed to change.

See:
LoC FADGI: DRAFT Significant Properties for Digital Video
Nestor (DE): Leitfaden DLTP AV Medien

Speaker notes

For some it’s the resolution, color information, audio quality - for others it’s sufficient to see/understand it “good enough”, or to be able to quickly edit-and-broadcast as the main focus.

Depends.

However, be aware that your recording may possibly be used in a different context in the future, so if possible don’t aim “too low”.

But please make an active decision and possibly document it (which ones and why) somewhere.

Significant properties

Depend on media type (and use case).

Video

- resolution
- framerate
- aspect ratio
- colorspace
- subsampling
- …

Audio

- “resolution”
  (= samplerate, bit-depth)
- channels
- channel layout
- …

Metadata

- language
- title
- author
- rights information
- …

Speaker notes

Definition fuzziness in the preservation community:

Some say “Significant” are the properties that must be maintained as-is and kind of “must never change”, whereas others define it as “should be aware of and decided how to deal with them”.

Further properties: * scan type (interlaced / progressive) * field order * color information * …

“Different strokes for different folks” 😉

- **Digitization:** As-original, as-untouched as possible. Records in realtime?
  (Plus: has headroom for optional restoration/improvements.)

- **Preservation:** Stand the tests of time.
  (Highest ‘original’ quality)

- **Mezzanine:** For daily work. High quality.
  (Optional, if preservation format can be used for this)

- **Access** For quick and easy access.
  (Quality not necessarily best/high, but very convenient to play)

Speaker notes

Sometimes one format can be used for all these cases (merely lower quality/bitrate for access or mezzanine), but with AV it’s not uncommon that these are different codecs (or even containers).

It is perfectly normal to have a preservation copy that requires transcoding before going into other workflows.

The formats for access and editing are more likely to be chanced more often than the preservation format.

For audio: we’re lucky.

PCM/WAV is used from digitization to preservation - and if bandwidth ain’t narrow, even for access.

Why? Because it became “small enough”.

Speaker notes

No notes on this slide.

“Different strokes for different folks” 😉

- **Digitization:** uncompressed/lossless or very-high quality lossy. (eg: V210, FFV1, MPEG-4 / PCM)
- **Preservation:** uncompressed/lossless or (very)high-quality lossy, open & documented, error-robust.
  (eg: V210, FFV1, MPEG-2, MPEG-4 / PCM)
- **Mezzanine:** high-quality/high-bitrate (lossy).
  (eg: MPEG-2, MPEG-4, ProRes / PCM)
- **Access** Most often lossy-compressed currently-popular format combination. (eg: H.264 / AAC in MP4)

Speaker notes

Format choice = A balance of …

- Size vs Quality
- Features
- Performance
- Sustainability
- plus: time, budget, staff
- and of course: convenience

Good starting point for assessing practical usefulness.

Speaker notes

Size: * Smaller = slower processing power * Smaller = less quality * Bigger = slower disk I/O

Quality: * More quality = larger files * More quality = slower processing

Performance: * How "fast" can a format be encoded/decoded? * Which hardware is required/desired? * Making it faster: Often limited by budget * Cameras / recording workflows must satisfy real-time and on-site constraints. * Editing

Features: * Can the format handle information/feature XY? (Can it do start timecode? timecode track? subtitles? extended metadata? etc)

Risks to format longevity

- Data errors
- Obsolescence
- Vendor lock-in
- Interoperability/complexity issues

Countermeasures?

Speaker notes

- Data errors: Files are corrupt. These errors may include filenames or filesystem tech-MD.
  - Obsolescence: Not supported anymore by accessible tools. This is neither God-given, nor irreversible, unless it's a black-box format. Format obsolescense/support is a human-made decision. Documentation/schematics are an important game changer.
  - Vendor lock-in: For long-term preservation, vendor- and technology-neutrality is a must: They will come and go, and with lock-in situations Eternal Migration is hindered or even impossible. Format normalization helps here.
  - Interoperability issues: If a format is read and written differently by different applications, it might "morph". This morphed version might work fine with the tools used in a certain environment, but might be completely broken if read/written by another tool that misunderstands the "dialect".

    A format offering more features, can be considered more "complex". Whereas a format offering less features is considered "simpler". Simpler (and more popular) formats are usually less susceptible to interoperability issues, than complex (and/or less popular ones)

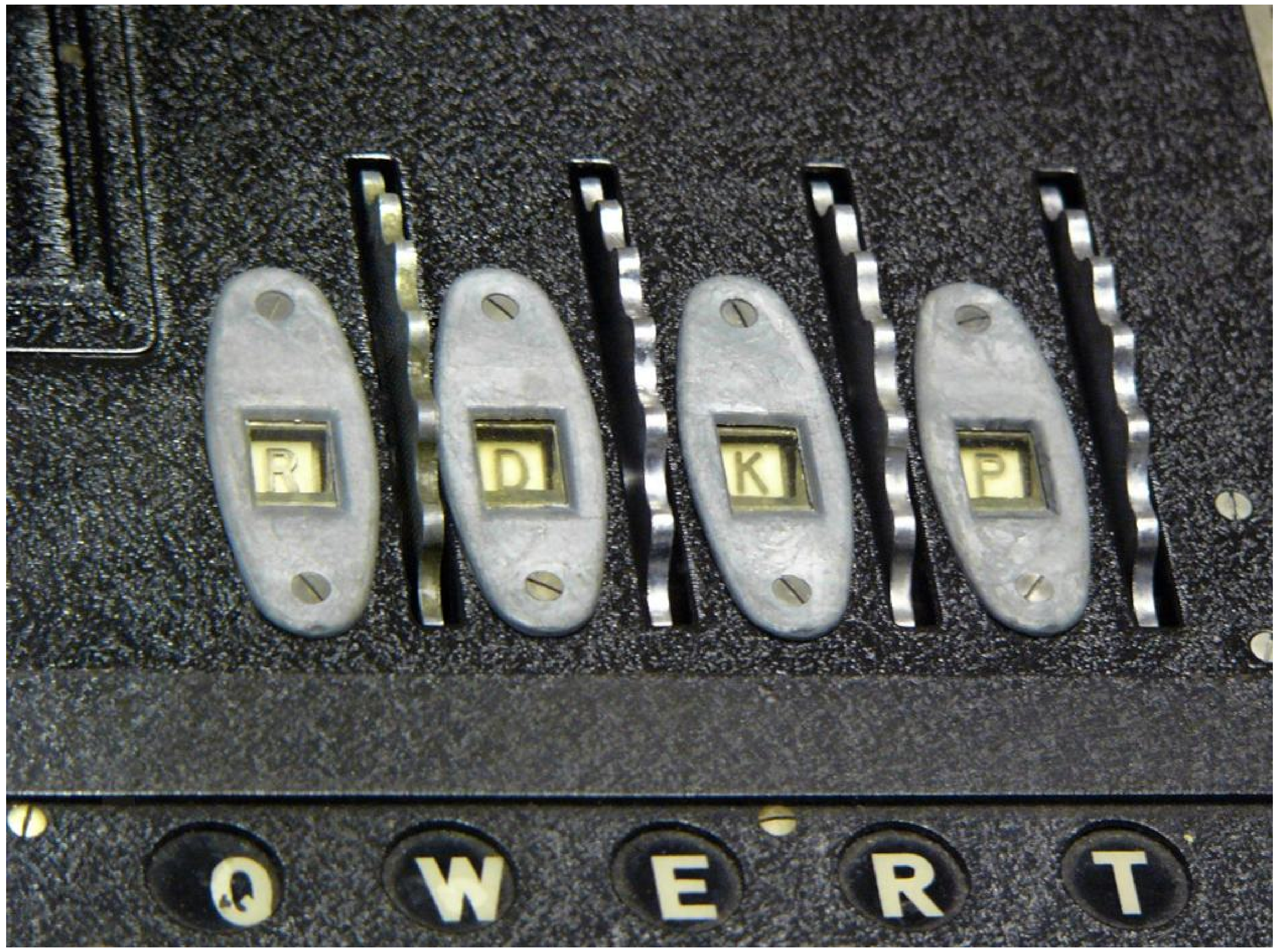
Data errors: Error resilience?

- **Bitstream checksums:**
  Ability to *know* if the content is intact.
- **Error concealment:** Optional choice of decoder to “mask” decoding issues. (decoder specific)
- **Make backup copies!** 😇

Speaker notes

Error resilience of a format is nice, but don’t solely rely only on it: Make backups! :)

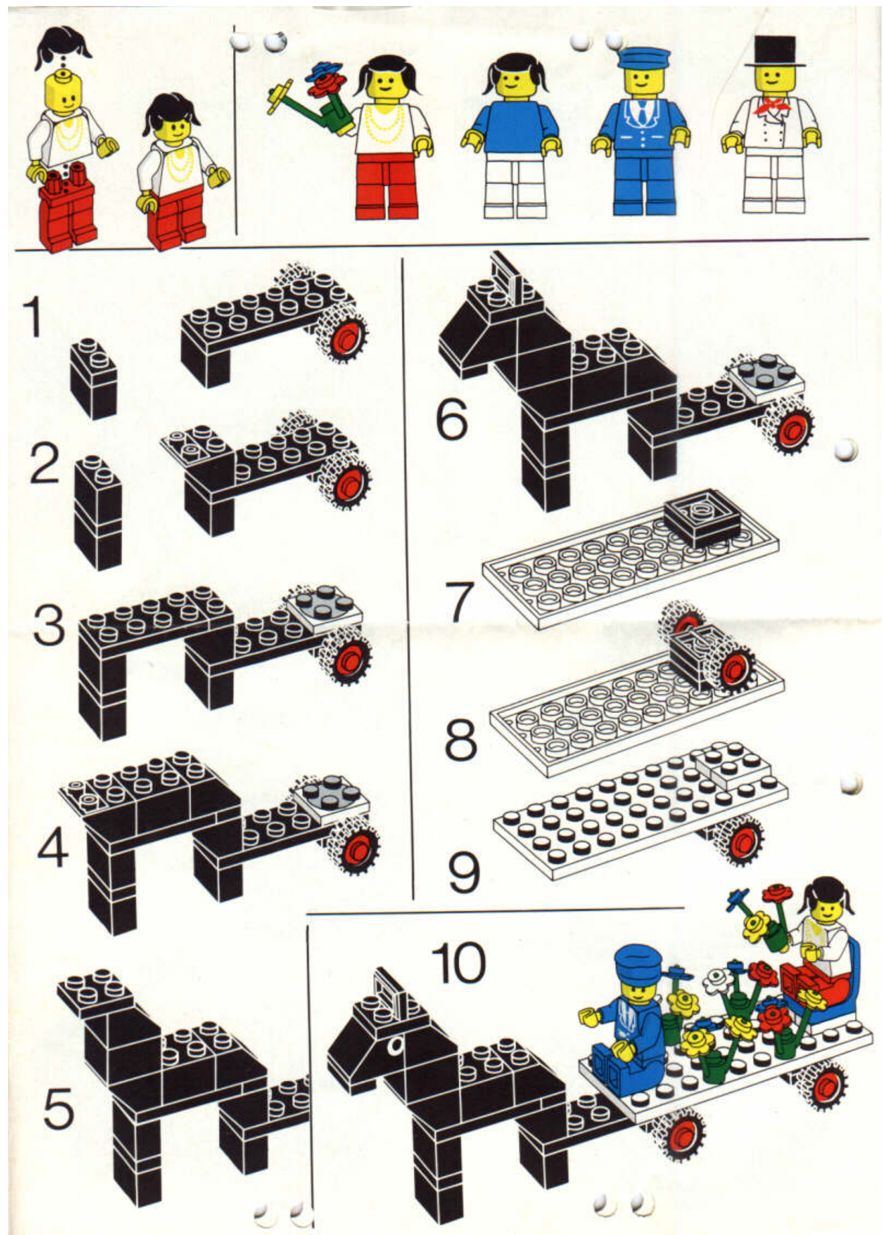
Open vs Closed



Speaker notes

What will be easier (=more likely) to be understood/accessible now and in the future: Documented language or secret code?

Theory vs Practice



“Implementation overrules paper specs. Always.”

Speaker notes

If the implementation is open: you can use, study, share and improve if necessary.

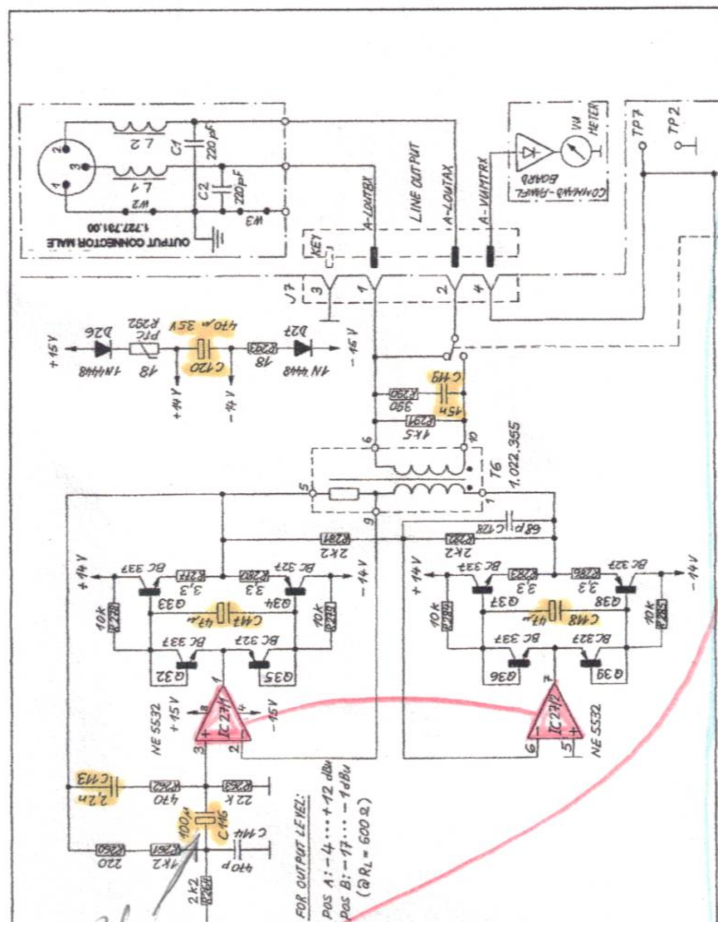
If it’s closed/proprietary then there’s nothing you can do about it: Black box. Vendor lock-in. With digital, this means that it’s quite likely that this as-is binary may not be able to keep running/functioning when its environment changes.

You’ve probably seen how fast even good and stable programs “age” until they stop working, once their native Windows version is “too long ago…”?

Even with official standards, their implementation is what creates the actual encoding/decoding. Regardless of what’s written in any paper manual (=theory): The code implementation (=practice) is what counts.

Therefore, it makes sense to demand an openly accessible reference implementation, with its source code under a license that allows to make use of it as you see fit.

The Eternal Replayer



+

=

Speaker notes

As an example:

- Schematics
- Building components
- +the right to use, study, share and improve them.

= “The Immortal Replayer”

Because it can be kept alive, or rebuilt or adapted to future needs or with future technology (whatever there may be).

Since software is the schematics and building components at the same time, having the source code and the right (=license) to use, adapt and share them - gives you an immortal file format. By definition.

This has been proven in the real world in many different digital domains outside AV or preservation many times already, with different tools and data formats from ancient computer systems. Even popular ones now dead (Atari, Amiga, C64, Amstrad Schneider, etc etc.)

Format Complexity

Speaker notes

Be careful with “one size fits all”: Sporks are good for camping, but there’s a reason why we still have separate tools for the job: spoon, knife and fork.

Considerations:

- More features = more complex / chance that only parts of specifications are supported by tool X.
- Can be non-trivial to judge what is “simple” and what is “complex”
- Find the sweet spot for your use case(s).

Format Complexity: Less is More

Good rule = “Minimalistic Data Format”:

- As simple as possible
- As complicated as necessary

Simpler = more stable, easier to use, keep alive, reconstruct or fix.

Speaker notes

No notes on this slide.

Your use cases/priorities?

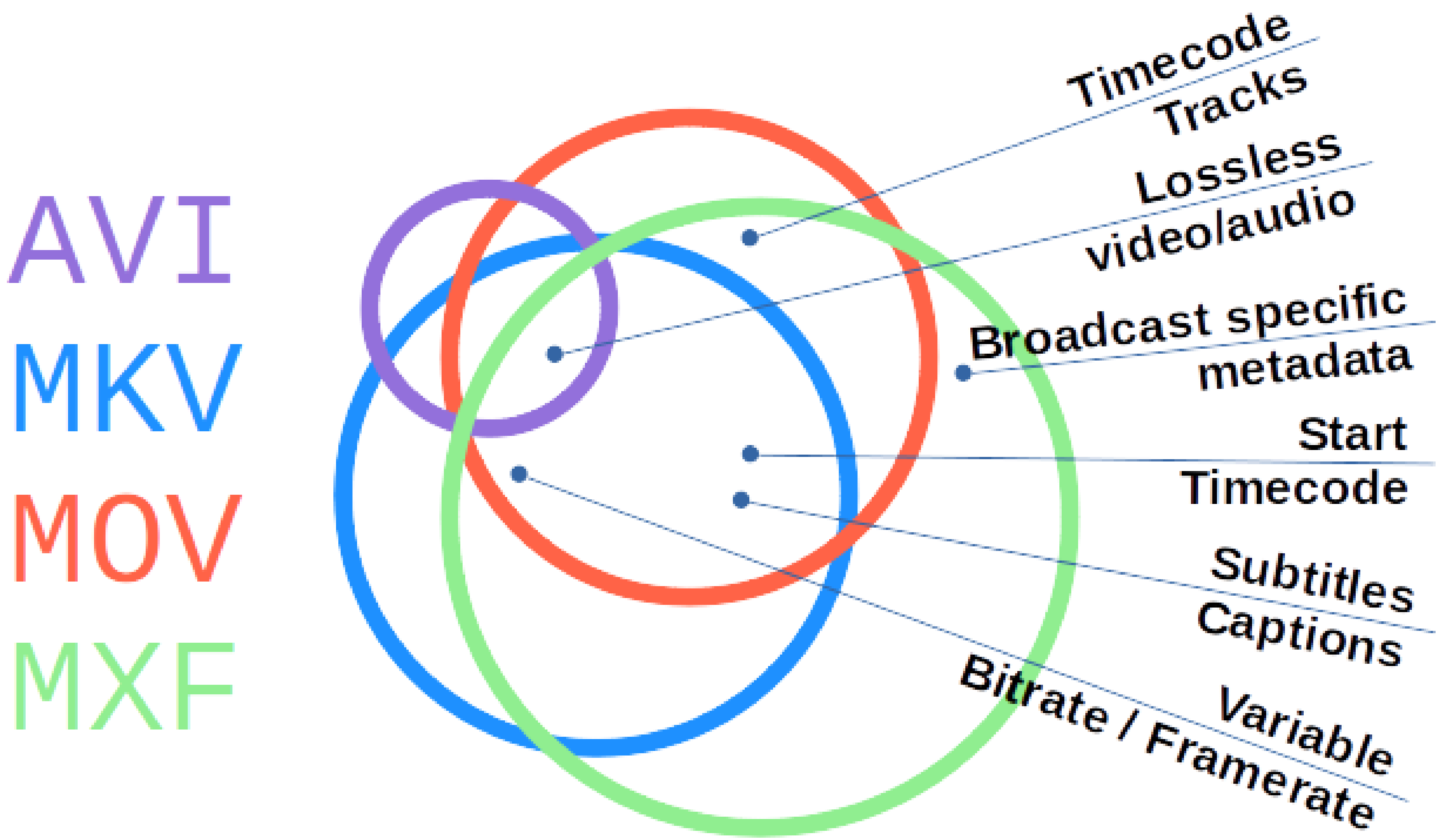
- Who will want/need to work with these files?
- Under which conditions?
- For how long?
- Digitization vs Production vs Preservation vs Access?
- Which properties are significant *to you*?

Speaker notes

It’s good to define (and write down) what you actually need from a file format in which use-case.

Also align these requirements with your restrictions (budget, time, staff, etc)

Yagni Kiss Moscow?



YAGNI / KISS / MoSCoW

Speaker notes

It may look cryptic, but is actually quite simple and useful:

It is an example for finding the “minimalistic data format” that suits your needs.

- **YAGNI = “You Ain’t Gonna Need It”:** It’s supposed to prevent you from selecting a format that is possibly too bloated or unncessarily complex, or less-well supported.
- **KISS = “Keep It Simple, Stupid”:** “The KISS principle states that most systems work best if they are kept simple rather than made complicated; therefore, simplicity should be a key goal in design, and unnecessary complexity should be avoided.”
- **MoSCoW = “Must, Should, Could, Won’t”:** “The Moscow method is a prioritization technique used in management, business analysis, project management, and software development to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement”

The example shows 4 different container formats for AV. Each one has different number and kinds of supported features, represented by the size of its circle.

- Larger circle = more features.
- Smaller circle = less features.
- Overlapping circles = common features.

Exercise: Your Format Policy

Must	Should	Could	Won’t
______	______	______	______
______	______	______	______
______	______	______	______
______	______	______	______

Choose a use-case and try to phrase your “wishes”.

Speaker notes

Now, according to the MoSCoW method, write down which features you:

- must have
- should have
- could have
- won’t have (this time)

And check which format provides them, then draw a dot in the corresponding circle area.

For example: Only MXF may be able to provide support for broadcast-specific metadata/functions, therefore that feature will only have a dot in the MXF circle. Whereas, all (except) AVI can store aspect ratio - so that dot would go into overlap of all - except AVI. The feature of “extremely simple, well-documented and stable/unmodified for ages” would likely to be a dot in either AVI or MKV.

Use this to find out which format fits your needs, while being “as simple as possible and only as complicated as necessary”.

Examples of published Policies

- Guidelines for the Digital Film Collection
  (Austrian Filmmuseum, 2018)
- Digital Preservation & Access Strategy
  (Irish Film Archive, 2016)
- Digital Preservation: Policy, Standards and Procedures
  (Netherlands Institute for Sound and Vision, 2016)

Speaker notes

The above links point to preservation policies published by other AV archival institutions. They also contain information about file format choices - in the context of AV preservation.

As you can see, the length and level of detail of such policies can vary greatly. Often it is great to simply have a rather short written form that at least provides insights in “how” and “why” someone has chosen one (format) option over another.

It is also good practice to write such policies in a way that they can be interpreted and applied properly even if the actual technical options/conditions change. For example, it’s okay to say “we chose format XY” - but without writing down “why”, this choice may seem incomprehensible and outdated once “format XY” is superseded by another format, or something else in the tech-envirionment or the conditions has changed.

Another example: It was common to use “MP3” for some (longer) recordings in the audio archiving domain, instead of WAV/PCM. Now this may seem odd, but at that time 10 MB/Minute WAV/PCM (CD-Resolution) was heavy lifting in terms of digital computer storage.

But back then, a “large” HDD RAID was 80 GB. Consisting of several 20 GB harddisks. Now, this would fit on a USB-Stick.

> *“Nothing ever doesn’t change, but nothing changes much.”*

All the time.

Preservation Checklist: translated! ☆

Sustainability:

1. Documentation openly accessible?
2. Open reference implementation?
3. How likely is it to be supported in tools/devices for which userbase?
4. Which features are implemented/tested/stable?
5. Which choice/requirements do I have to handle it beyond “shelf life”?
6. Is it legal/possible to handle it in future/different situations?
7. Can it contain proper metadata?

Quality and functionality:

1. Preserve significant properties?
2. Sufficient image/sound quality and robustness to multi-generation copies?
3. Interoperability / ease of usage & access?
4. Direct use for editing?
5. How many different formats will I need (pile up)?
6. Handle performance / data size requirements?

Speaker notes

1. Documentation openly accessible? Without documentation (or access to it), a file format is the equivalent of a secret code/language. You can imagine how well “secret code” satisfies preservation requirements?

2. Open reference implementation? Any code that implements a file format can be called an “implementation” of that format. A “reference implementation” is an application that is able to read/write that format in a proper way, used to check the validity of other implementations that read/write the same format. With proprietary (=closed source) formats it is not uncommon that there is no real “offical” reference implementation - but merely a black-box binary provided by the vendor, not necessarily freely accessible and possibly subject to change at their will. Without an open reference implementation (=including source code), interoperability issues are more likely to happen.

3. How likely is it to be supported in tools/devices & for which userbase?

   Adoption of a format plays a major role in how easy (=cheaper) vs hard (=more expensive) it is to use a format. It may be necessary/useful to engage with vendors of developers and ask for supporting a certain format in tools you would like to use.

The userbase is important, as it has a great impact on where a format is supported, and for which use-cases it is real-world tested. If a format is common in a userbase that covers your use cases, it’s good. If a format was designed/intended for a userbase further away from your use cases, you may find that you get less support (or understanding) for your needs.

4. Which features are implemented/tested/stable?

   Developing, testing and supporting a file format is a tricky task. Therefore it is common that the “most popular/important” features are the ones most likely to be tested and working reliably. Other functions that may also be defined (even in a standardized format), may not receive the same attention by its supporters, often leading to (interoperability-)issues with not-so-popular functions. This is especially important for us, since certain features required for preservation are often less important for the main intended use-case of a format.

A popular example is “lossless”: both, JPEG2000 as well as H.264 can produce lossy as well as lossless encodings. However in practice, the lossless feature is often neither tested (and sometimes not even supported) in many of their implementations.

Links

- Primer on Codecs for Moving Image and Sound Archives
- Hex Editing for Archivists
- Comparing Video Codecs and Containers for Archives
- Digital Media Primer for Geeks
- A short guide to choosing a digital format for video archiving masters
- Media Digitization and Preservation Initiative (MDPI)
- Understanding audio bitrate
- Data Compression (Wikipedia)

Speaker notes

No notes on this slide.

- Fin -

Speaker notes

No notes on this slide.

Questions?

Comments?

Peter Bubestinger-Steindl

`Peter @ ArkThis.com`

Speaker notes

No notes on this slide.