Overview of UHAPI Architecture and Specifications

John Vugts

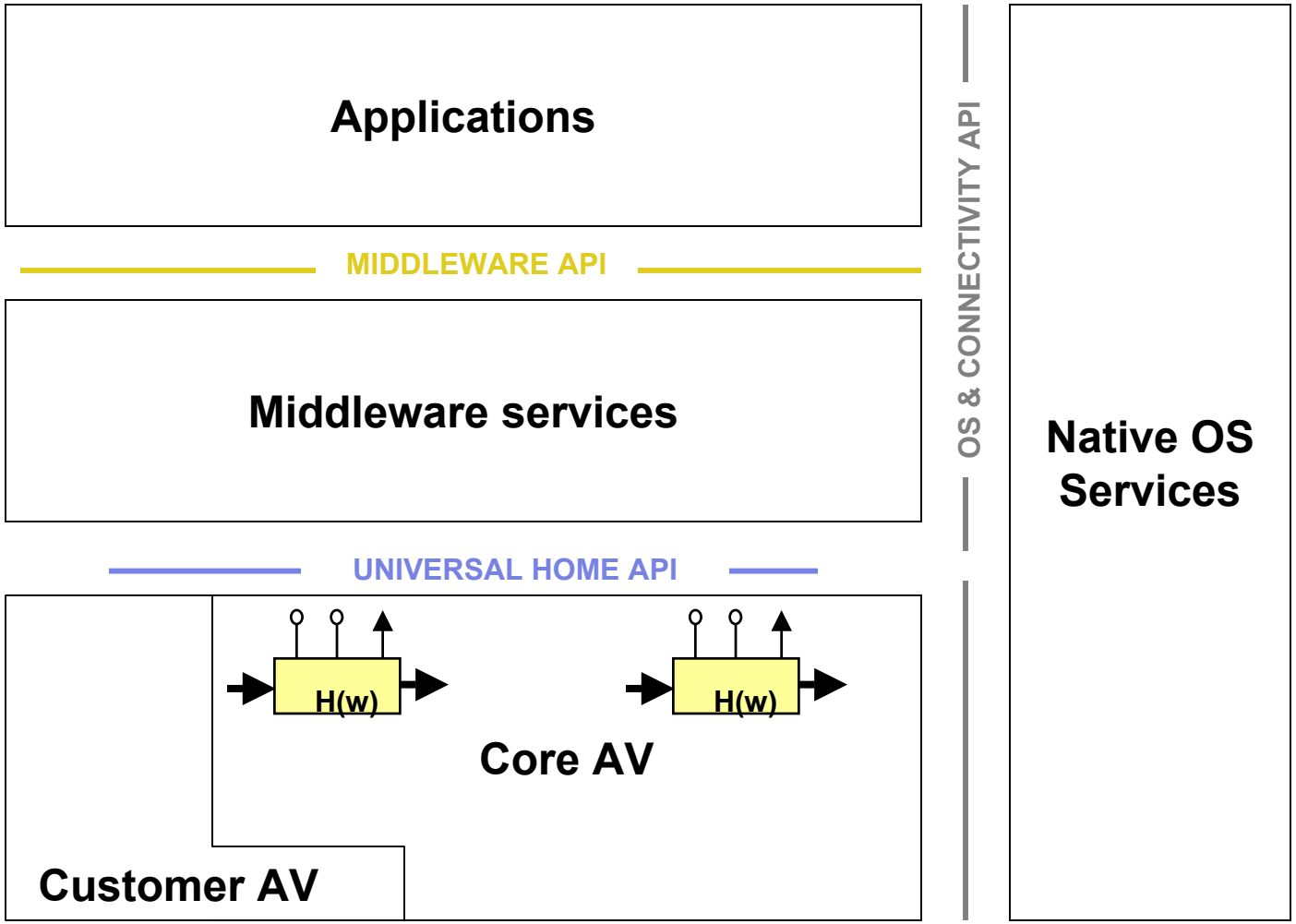
June 14th, 2005

Yokohama, Japan

Outline

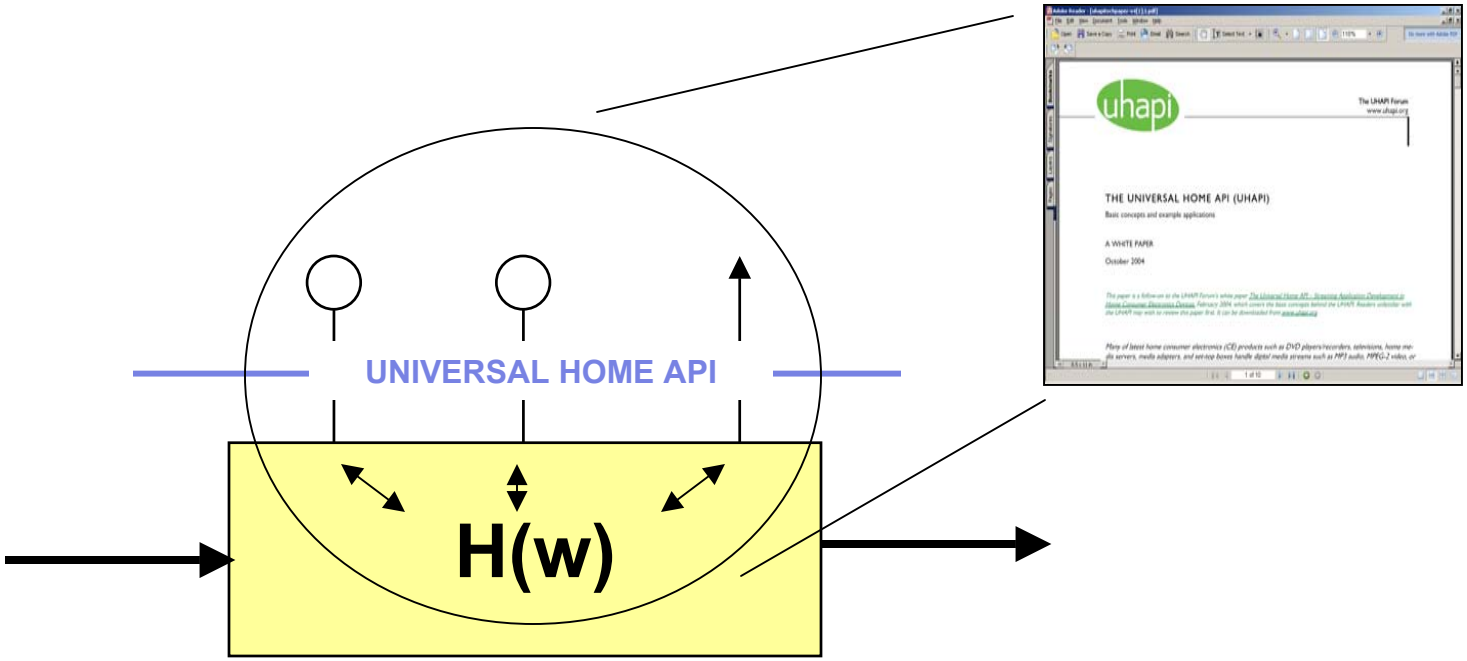
- UHAPI scope
- UHAPI characteristics
- Logical components, roles, interfaces
- Use cases
- UHAPI and variation
- API specification walk-through
- PVR preview

UH Scope: Control AV streams



UHAPI is an API to the Middleware

- Focus on runtime control by the Middleware (ISV).
- It does not specify e.g. streaming interfaces.
- Specification structure deals with diversity.



Example UHAPI 1.0 Logical Components

- **Front-end components (11)**
  - Channel Decoding
  - Tuning
  - HdmiIn
  - …
- **Decoders/encoders (5)**
  - ATSC Decoder
  - Transport Stream Demultiplexing
  - …
- **Video processing components (16)**
  - Basic Video Featuring
  - Color Transient Improvement
  - Sharpness Measurement
  - Video Mixing
  - ….
- **Audio processing components (10)**
  - Audio Bass Enhancements
  - Audio Dynamic Range Control
  - Audio Volume Control
  - …
- **Generic (8)**
  - Connection Management
  - Fatal Error Handling
  - Unknown
  - …

UHAPI 1.0 contents

General documents (7) :

API Specification Reader's Guide
API Naming Conventions
Error Handling
Execution Architecture
Notification
Qualifiers Quick Reference
API Evolution Rules

Type specifications (2) :

Basic Types
Global Types

API specifications (50) :

Front End Components (11)

Analog Audio & Video Demodulation
Analog AV Input
Analog Audio Decoding
Channel Decoding
RF Amplification
Out Of Band Tuning & Demodulation
Signal Strength
Tuning
HdmiIn
SPDIF-in
VBI Slicing

Decoders/Encoders (5)

ATSC Decoder
Image Decoding
SPDIF Decoding
STC Decoding
Transport Stream Demultiplexing

Video Processing Components (16)

Ambient Level
Analog Video Decoding
Analog Video Encoding
Analog Video Encryption
Anti Aging
Basic Video Featuring
Black Bar Detection
Color Transient Improvement
Dynamic Noise Reduction
Histogram Modification
Noise Measurement
Scan Rate Conversion
Sharpness Enhancement
Sharpness Measurement
Video Color Enhancement
Video Mixing

Audio Processing Components (10)

Audio Automatic Volume Leveling
Audio Bass Enhancements
Audio Dynamic Range Control
Audio Mixing
Audio Noise Generation
Audio Program Selection
Audio Volume Control
Equalizing
Speaker Set /Headphones
Output Configuration

Generic (8)

Analog AV Output
SPDIF-out

Connection Management
Fatal Error Handling
I am Alive
Pin Objects
Unknown
URL Source

Outline

- UHAPI scope
- UHAPI characteristics
- Logical components, roles, interfaces
- Use cases
- UHAPI and variation
- API specification walk-through
- PVR preview

UHAPI Characteristics (1)

- **Designed with a middleware view in mind**
  - To support a large set of middlewares
- **API family for AV functionality (analog, digital)**
- **Interface-based programming**
  - Provides a consistent, orthogonal, coherent set of interfaces
  - Well defined (syntax and semantics)
- **Binary interface**

UHAPI Characteristics (2)

- **Hardware and implementation independent interface**
  - Allows freedom in implementation and evolution
  - Supports both HW and SW streaming
  - Supports both on and off chip peripherals
  - Does not expose the implementation software component architecture
- **Processor independent**
- **Used processor not visible to client**
  - Support efficient RPC implementation
- **Operating System independent**

UHAPI Characteristics (3)

- **Uses standard mechanisms for**
  - Notifications (runtime binding)
  - Error handling
  - Connection management (simple to program)
- **Well-defined execution architecture**
- **Uses standard COM like mechanisms (a small subset)**
  - IUnknown
    - QueryInterface
    - AddRef & Release
  - v-tables
  - GUIDs

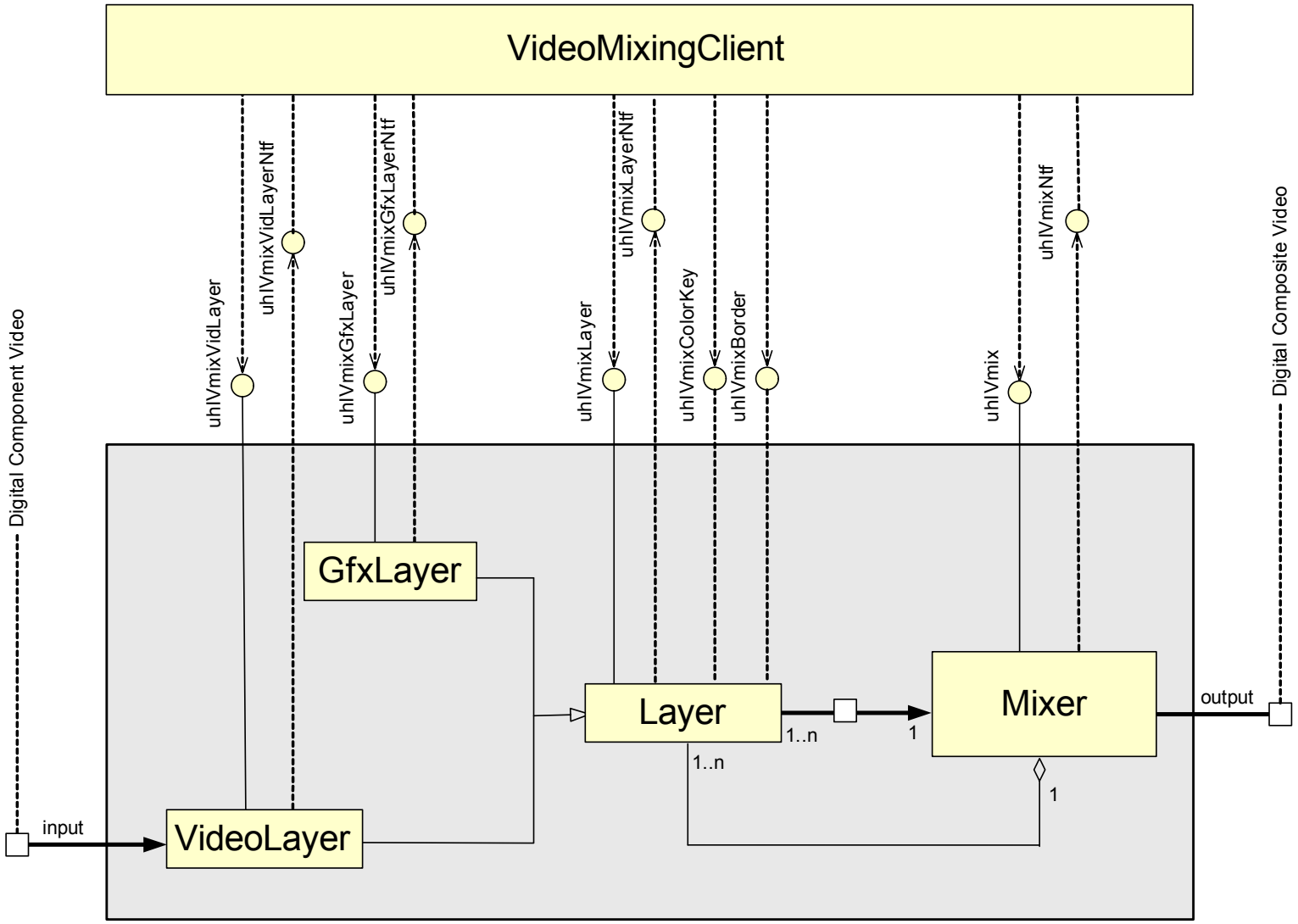
Outline

- UHAPI scope
- UHAPI characteristics
- Logical components, roles, interfaces
- Use cases
- UHAPI and variation
- API specification walk-through
- PVR preview

Logical components, roles and interfaces

- **Each logical component has its own specification document**
- **A logical component can have one or more roles**
- **A role is an abstract class in UML terms**
  - It describes behavior/interface interaction without referencing a particular implementation
- **A role typically provides one or more control interfaces and "requires" one or more notification interfaces**
- **An interface is a coherent set of functions**

Logical components, roles and interfaces



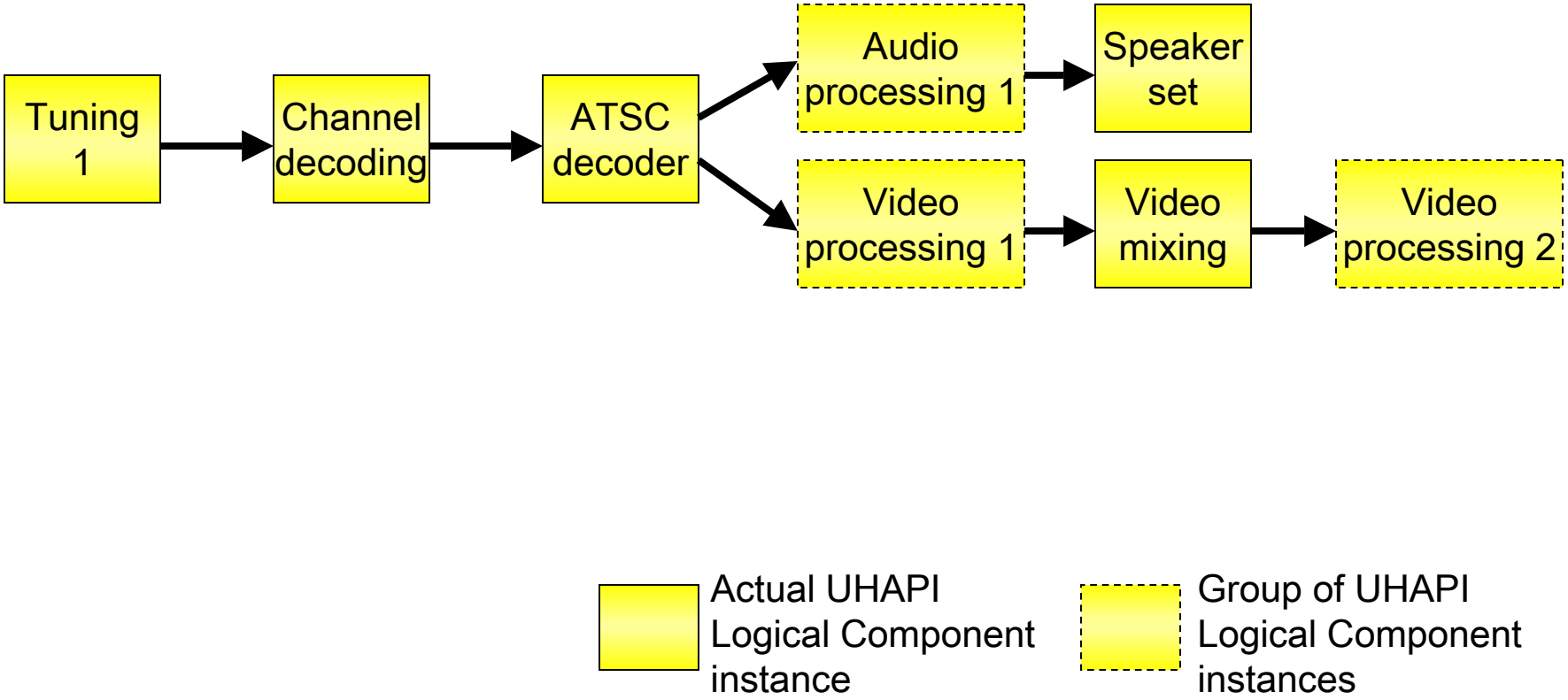
Outline

- UHAPI scope
- UHAPI characteristics
- Logical components, roles, interfaces
- Use cases
- UHAPI and variation
- API specification walk-through
- PVR preview

Use cases

- **Use case: one or more streaming graphs of connected logical component (LC) instances**
- **LC instances in a use case are active simultaneously**
- **UHAPI does not define use cases**
- **Vendor/implementer chooses use cases for its solution (platform instance)**
  - A platform instance may support any number of use cases, including just a single use case.
- **UHAPI implementation realizes the use cases**
  - This relieves the client from the difficult and HW-specific task of setting up and connecting components (priorities, buffer sizes, …)

Example use case: digital input single window



Outline

- UHAPI scope
- UHAPI characteristics
- Logical components, roles, interfaces
- Use cases
- UHAPI and variation
- API specification walk-through
- PVR preview

Logical component diversity

- **Optional interfaces**
- **Parameter ranges**
  - e.g. min/max volume
- **Available resources**
  - e.g. number of section filters
- **Other**
  - e.g. which standards are supported

- **See the “Diversity” section of a Logical Component specification**

UHAPI is an API framework to the MW

UHAPI specifications

Framework level

UHAPI platform instance A specification

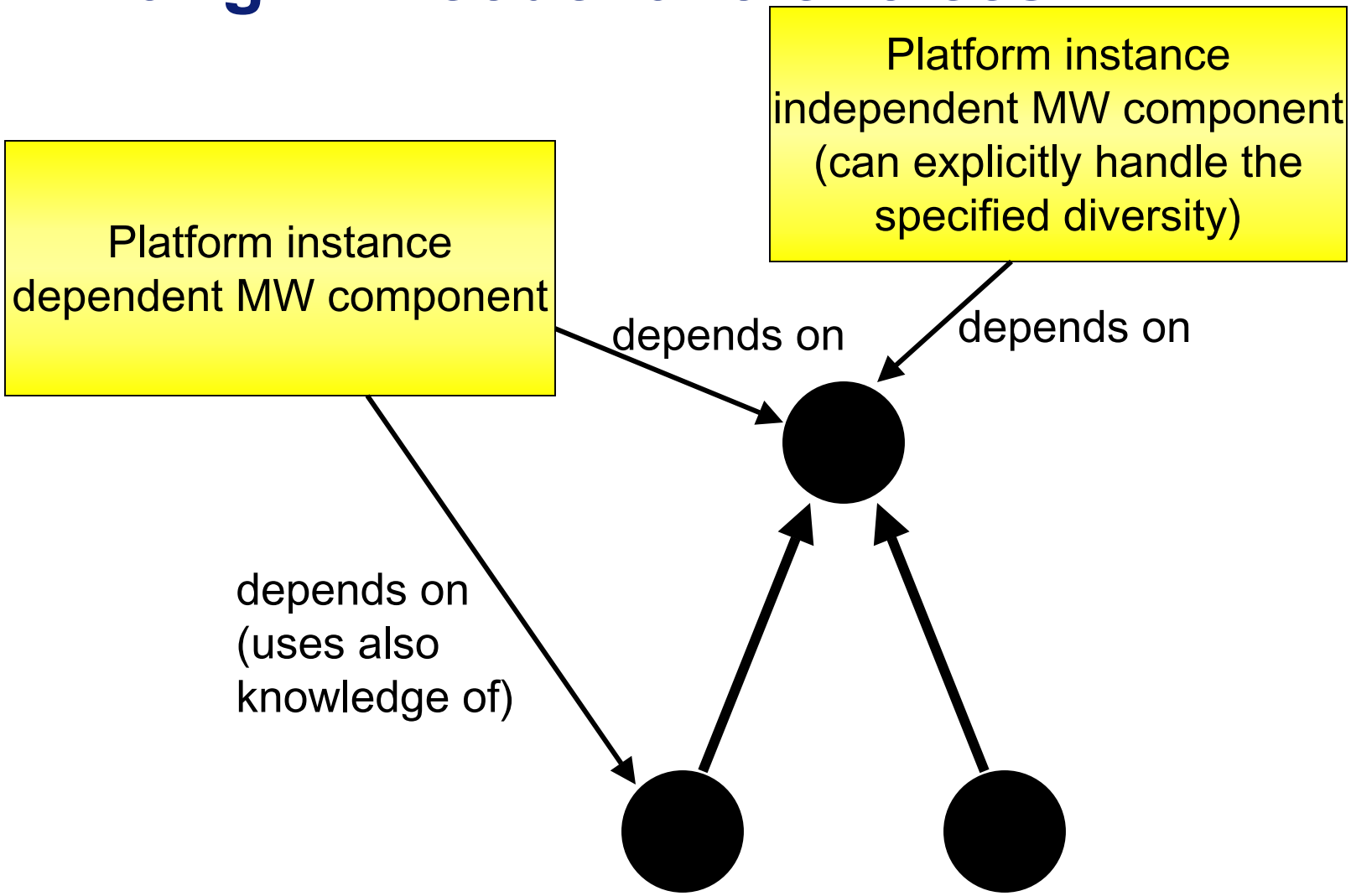
UHAPI platform instance B specification

Platform instance level

What documentation does a vendor need to provide with a platform instance?

- **Which logical component instances are supported**
  - Some (simple) platform instances may only need a few LC instances
  - No need to include all UHAPI LCs in a platform instance
- **Diversity choices for each logical component instance**
  - For example, availability of optional interfaces (= optional features).
- **What use cases are supported**
- **Optional: Resource usage figures (CPU cycles, memory, memory bandwidth, algorithms used, …)**

Writing MW code: two choices



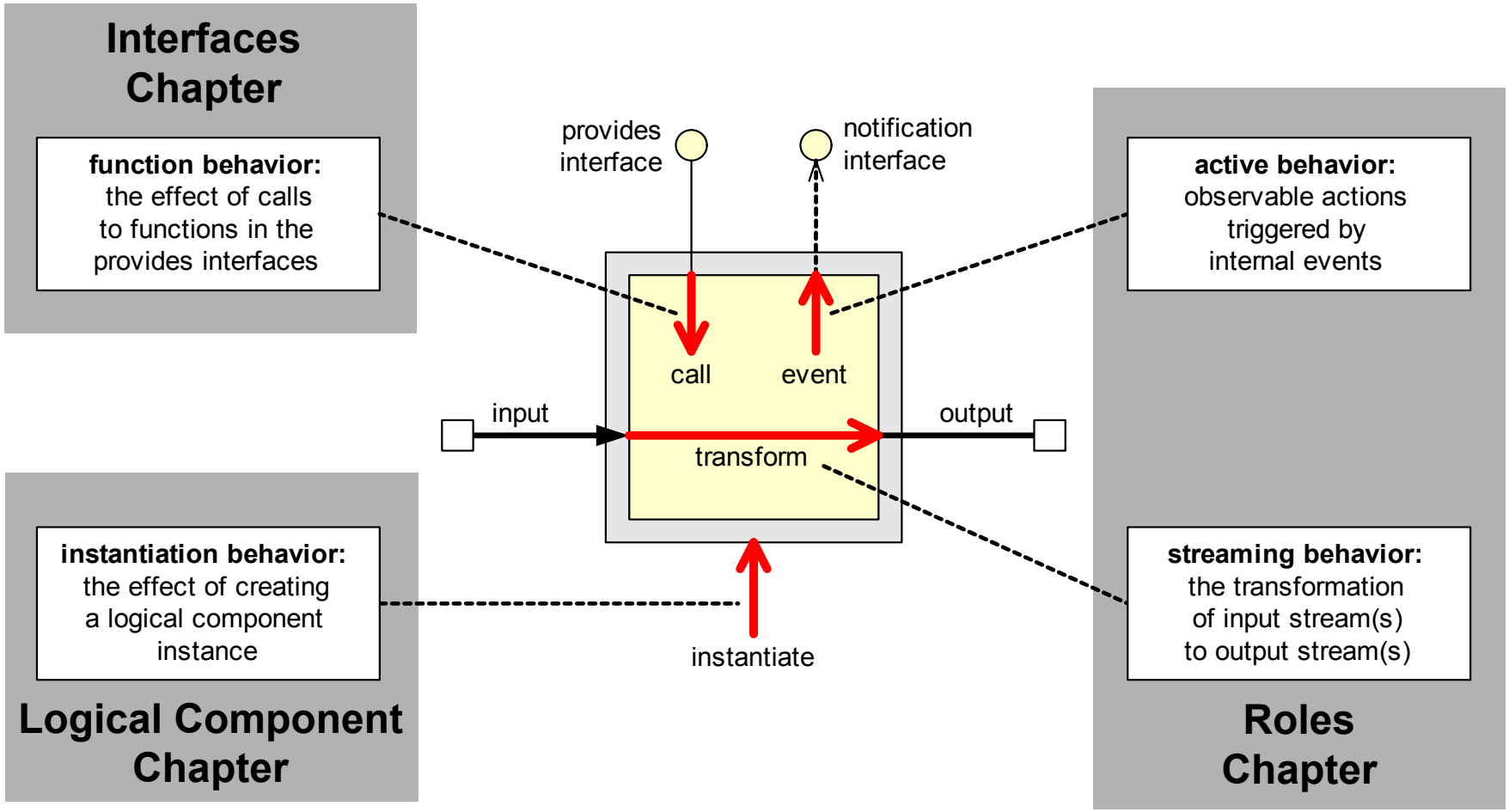
Outline

- UHAPI scope
- UHAPI characteristics
- Logical components, roles, interfaces
- Use cases
- UHAPI and variation
- API specification walk-through
- PVR preview

API Specification

Content of each Logical Component

Specification views



Outline

- UHAPI scope
- UHAPI characteristics
- Logical components, roles, interfaces
- Use cases
- UHAPI and variation
- API specification walk-through
- PVR preview

Personal Video Recorder (PVR) extension

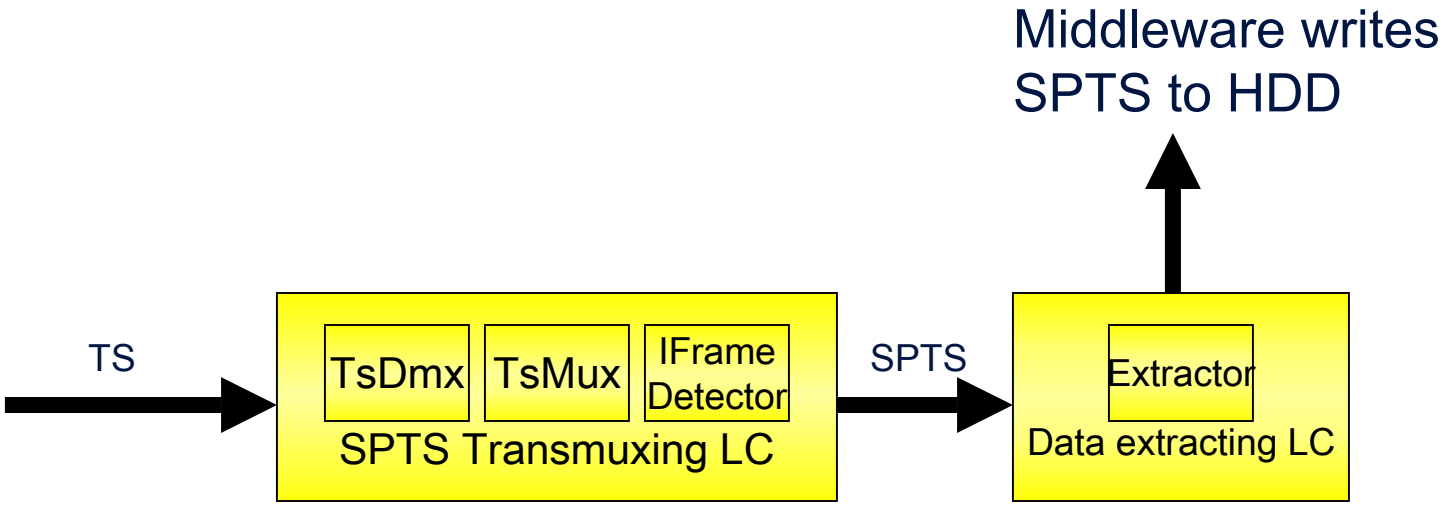
- **New UHAPI PVR workgroup started in April this year**
- **Engineers currently working on PVR:**
  - Philips (2), Samsung (2), NVIDIA (1), HP (1)
- **Process:**
  - Step 1: Outline/architecture total PVR solution
  - Step 2: Outline/design of individual logical components
  - Step 3: Detailed specification of individual logical components
- **Current status: in step 2**
  - 11-page outline/architecture document available
  - Initial Logical Components being written
- **Review of results step 1 are now being reviewed by selected ISVs**
- **PVR specification expected by August 2005**

PVR workgroup charter (some key points)

- **To enable Personal Video Recording on Digital reception**
  - Record live broadcast to HDD
  - Play recorded programs from HDD
  - Play program from HDD while recording program to HDD
  - Various trick play modes
  - Support for storing SPTS (TS→SPTS transmuxing for efficient use of storage space)
- **To enable Networked PVR**
  - Align this with DLNA requirements on e.g. SPTS
- **Middleware responsibility**
  - PSI/SI parsing, PVR EPG database build up, maintain index-files with I-frame positions, execution of trick modes, store and retrieve content from HDD
- **Security and PVR on analog reception are second priority**

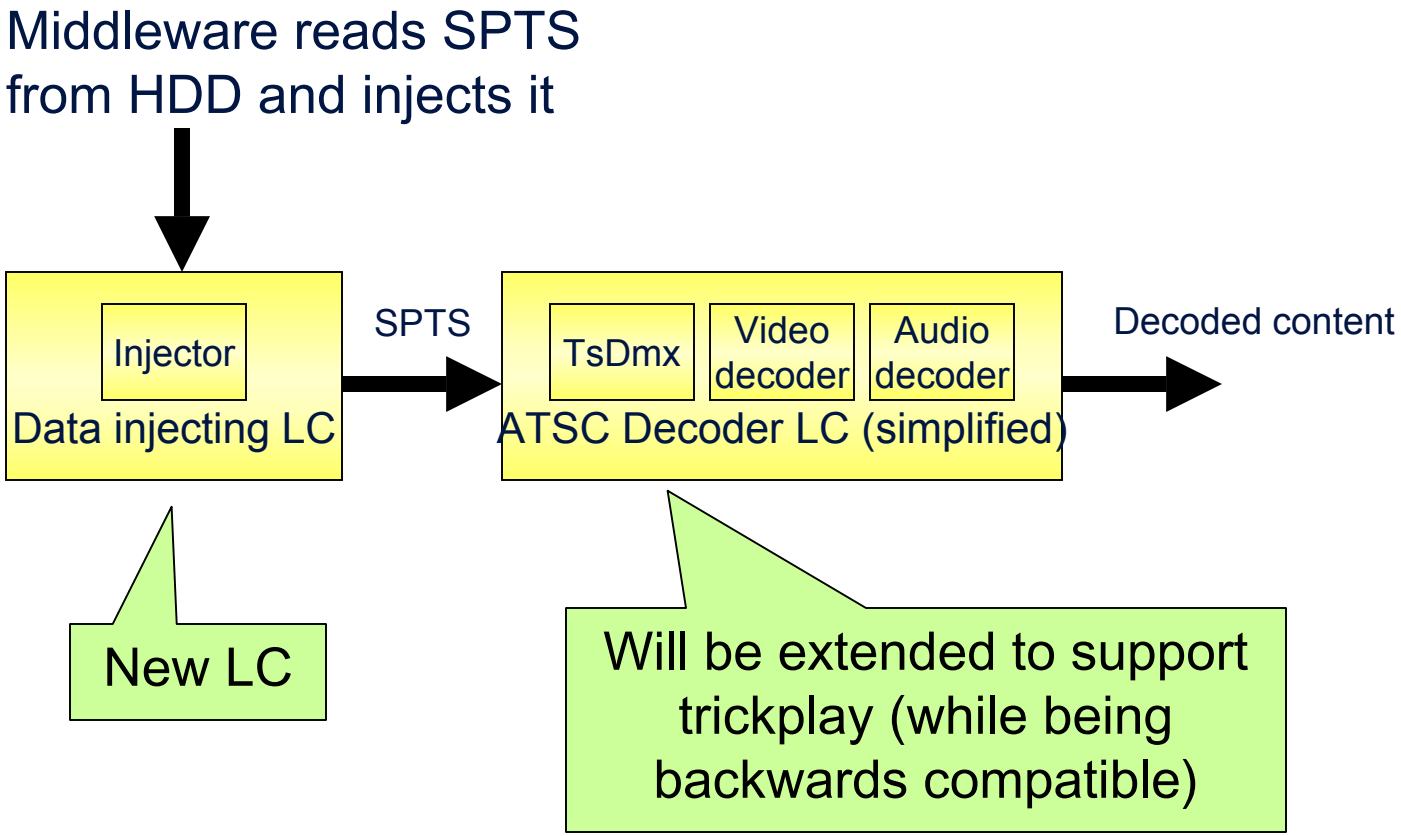
PVR (preview)

- **New logical components for recording an SPTS**



PVR (preview, cont'd)

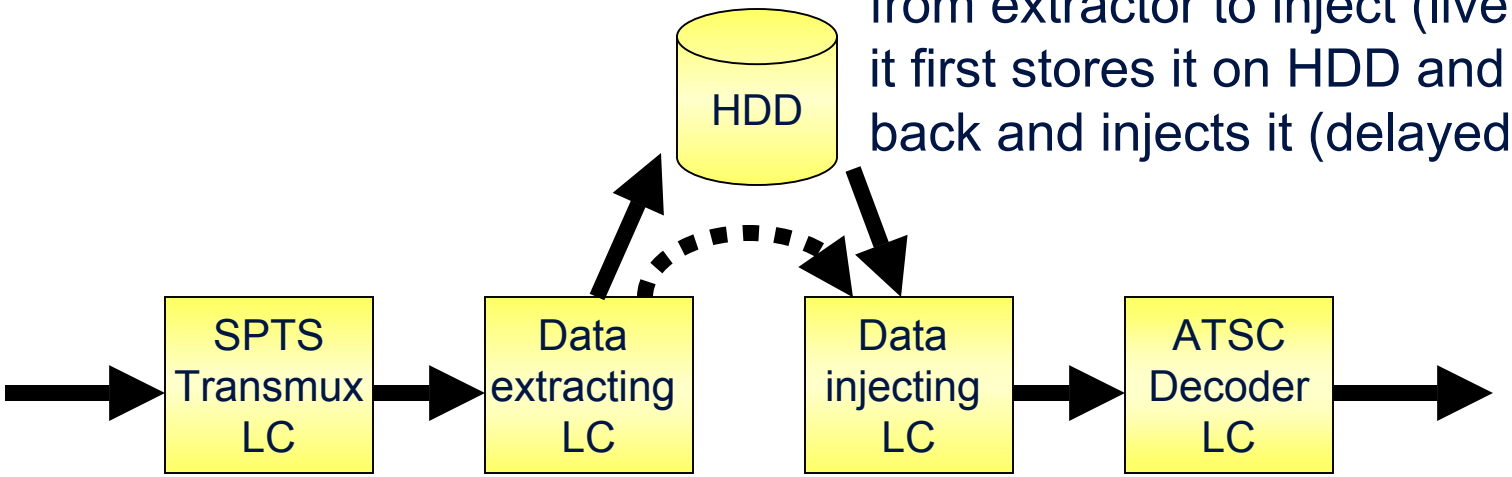
- **Play back of an SPTS from HDD**



PVR (preview, cont'd)

- **Example use case for live/delayed viewing**

Middleware either passes content directly from extractor to inject (live viewing) or it first stores it on HDD and later reads it back and injects it (delayed viewing)



Summary

- UHAPI is complete, consistent and well documented
- UHAPI is hardware independent
- UHAPI has support for variation and scalability "built into its genes"
- UHAPI allows for vendor extensions in a structured way
- PVR specification under development and expected July 2005
- Alignment with CELF via OSS and DirectFB
- Download UHAPI 1.0 at www.uhapi.org