Deliverable

Project Acronym:	**ImmersiaTV**
Grant Agreement number:	688619
Project Title:	*Immersive Experiences around TV, an integrated toolset for the production and distribution of immersive and interactive content across devices.*

D3.8 Workflow integration & End-to-end tests

Revision: 0.7

Authors:

Maciej Glowiak (PSNC)

Delivery date: M9

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 688619

Dissemination Level		
P	Public	x
C	Confidential, only for members of the consortium and the Commission Services	

Abstract: This deliverable describes the process and methodology of integration components of the Immersia TV project. It contains information on how the releases were prepared and what functionalities appeared in each release. The document also contains description on testing process and lessons learned from the testing the integration process.

REVISION HISTORY

Revision	Date	Author	Organisation	Description
0.1	22-08-2016	M.Glowiak	PSNC	Deliverable structure, first contribution, list of releases
0.2	30-08-2016	Sz.Malewski	PSNC	Methodology and Testing procedures
0.3	31-08-2016	M.Glowiak	PSNC	Additional descriptions, styling, diagrams
0.4	6-03-2017	M.Glowiak	PSNC	Updated version
0.5	10-03-2017	J.Nunez	I2CAT	Additional improvements
0.6	16-03-2017	J. Lourenço	Lightbox	Tests of production tools
0.7	17-03-2017	M.Glowiak Sz.Malewski	PSNC	Integration, minor improvements

EXECUTIVE SUMMARY

This deliverable describes the workflow integration process and methodology as well as contains information of end-to-end testing procedures. The document is related to other project deliverables, especially D3.1 – Design Architecture, D3.2-D3.6 software deliveries and D4.4 (Technical evaluation), D4.5 (User evaluation). The intention was to create this deliverable incrementally and iteratively, building further versions basing upon this one. This version focuses on integration, delivery software releases and tests done for Pilot 1, what means it contains information dedicated to off-line production and playback workflow until release 0.6.

Chapter 1 contains introduction to the document and Chapter 2 is dedicated to description of components (2.1) and functionalities of particular releases (Section 2.2). In Chapter 3 there are information on workflow integration process (Section 3.1), methodology (Section 3.2) and End-to-end tests (Section 3.3). Document is shortly concluded by Chapter 4.

CONTRIBUTORS

First Name	Last Name	Company	e-Mail
Maciej	Głowiak	PSNC	mac@man.poznan.pl
Szymon	Malewski	PSNC	szymonm@man.poznan.pl
Juan	Nunez	I2CAT	juan.antonio.nunez@i2cat.net
João	Lourenço	Lightbox	joao@lightbox.pt
Maciej	Stróżyk	PSNC	mackostr@man.poznan.pl

CONTENTS

TABLE OF FIGURES

LIST OF ACRONYMS

Acronym	Description
HMD	Head Mounted Display
WP	Work Package
D	Deliverable
T	Task

1. INTRODUCTION

1.1. Purpose of this document

In this report, partners of the Immersia TV consortium describes the outcome of the tests and system integration. The document explains the results of tests, current limitations and suggestions for architecture and implementation improvements. This document will describe the integration process and tests done iteratively across all 3 phases of the work conducted in the project.

1.2. Scope of this document

This document focuses on integration process, test methodology and tools used for management of development process. It contains short description of all releases.

The document bases on the D3.1 (Architecture design) and other deliverables of WP3 (dedicated to particular components provided by partners D.3.2 – D3.6). As some tests were described in D4.4 (Technical Evaluation) and D4.5 (User Evaluation), this deliverable just refer to them if necessary.

1.3. Relation with other ImmersiaTV activities

This deliverable is part of task T3.8 (Workflow integration & End-to-end tests) in WP3 (Immersive Broadcast Platform). The relationship between this task and the other WP tasks is shown below.

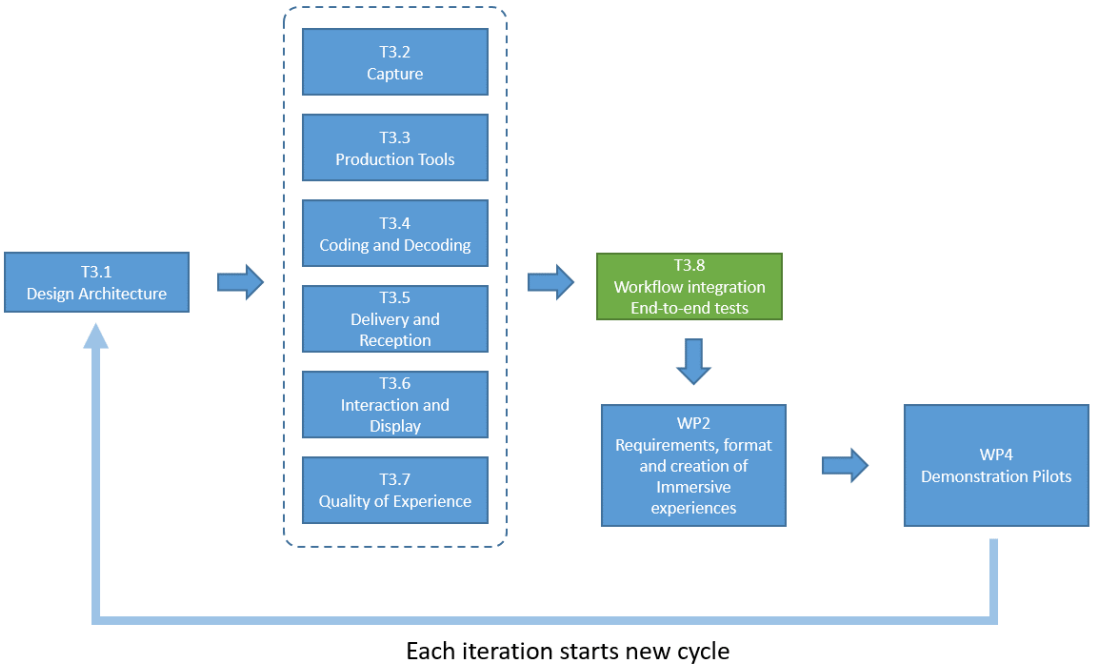

Figure 1: Role of T3.8 within the ImmersiaTV platform

2. RELEASES

2.1. Components

ImmersiaTV aims to distribute omnidirectional and directive audiovisual content simultaneously to head mounted displays (HMD), companion screens and the traditional TV. The content distributed is constituted of one or more omnidirectional videos, complemented with several directive shots, and metadata detailing how to merge these streams in an immersive display, in coordination with directive and omnidirectional videos also shown in traditional TVs and tablets.

As detailed in Deliverable D3.1 - Architecture Design, the ImmersiaTV platform involves the following modules: Video Capture, Video Stitching, Post-Production, Encoding, Distribution, Reception & Interaction and QoE component.

In order to achieve Pilot 1 functionality (off-line workflow), not all components were published as releases available for testing. The content capture was handled using camera rigs, and for stitching the Vahana VR and AutoPano were used, as it was described in details in D3.1. While partners responsible for advanced encoding, live production tools and QoE were developing their modules for next phases, the first releases focused for off-line scenario.

The Release 0.1 was made internally by I2CAT for testing tools and the process of creating release package. It was not published for partners, as it's not mentioned on the list below. The releases 0.2 - 0.6 contained two most important components for Immersia TV system for pilot 1: Player (part of the Reception and Interaction) and Off-line production tools. Release 0.6 contained also web DASH server for use in demonstration environment.

2.1.1. Player

This component handles the end-user's reception side and display. It takes care of selecting proper video streams, receiving, decoding and displaying them. It also handles the synchronization of multiple received streams in order to present them to the end-user. The user can interact with the received content by selecting streams, choosing the device and performing basic playback actions. The client application provides logging functionality of playback parameters, that can be analysed off-line.

2.1.2. Production tools

This component encompasses a set of tools and plugins for offline video editing with functionality of synchronization and combining multiple 2D and omnidirectional video sources and auxiliary data together. These data come from Capture and Stitching blocks. Off-line production tools operate on video files from omnidirectional and directional cameras and allows the editor to select, modify and combine streams using various transition effects and produce metadata. Off-line production tools produce video files as well as XML metadata that needs to be uploaded to Content Distribution server.

2.2. Description of releases

2.2.1. Release 0.2

The Release 0.2 was published **16 Jun 2016** and contained first results of the project. It integrated basic components for testing the system architecture and concept. It was composed of components such as:

- ImmersiaTV player
- Premiere Pro extension
    - Multiplatform content generation (tablet, HMD, tv).
    - Transitions based on luma masks (without interaction).
- Tools for performance tests

It didn't contain Session Manager, DASH converter and containers to consume the Immersia TV services, which were planned for future releases. The ImmersiaTV Premiere Pro Extension was comprised of 2 parts: the Effect Controls and the ImmersiaTV Export Panel.

2.2.2. Release 0.3

The Release 0.3 was published **4 Jul 2016** and extended previous releases. It contained components such as:

- ImmersiaTV player for android, windows and android TV
- Premiere Pro extension with mac and windows installers
    - Multiplatform content generation (tablet, HMD, tv),
    - Transitions based on luma masks (with basic interactions interaction),
    - Automatic generation of DASH files
- Dash conversion and content publication services
- Tutorial explaining how to create a scene and use the immersiaTV services

As the work on Session Manager was on-going, this component was postponed for future releases.

2.2.3. Release 0.4

The Release 0.4 was published **21 Jul 2016** and extended previous versions with more advanced functionality. It contained components such as:

- ImmersiaTV player for android, windows and android TV
- Premiere Pro extension with mac and windows installers
    - Multiplatform content generation (tablet, HMD, tv).
    - Transitions based on luma masks (with basic interaction).
    - Automatic generation of DASH files.
- Dash conversion and content publication services
- Tutorial explaining how to create scenes covering the main cases of ImmersiaTV, and how to use the ImmersiaTV services

As the fully functional Session Manager was planned for future releases, this version doesn't contain this component.

2.2.4. Release 0.5

The Release 0.5 was published 17 Aug 2016 and extended functionality of Production Tools. The release contained elements such as:

- ImmersiaTV player for android, windows and android TV
- Premiere Pro extension with mac and windows installers
    - Multiplatform content generation (tablet, HMD, tv)
    - Transitions based on luma masks (with basic interaction).
    - Automatic generation of DASH files.
- Dash conversion and content publication services
- Tutorial explaining how to create scenes covering the main cases of ImmersiaTV, and how to use the ImmersiaTV services

2.2.5. Release 0.6

The Release 0.6 was published 25 Aug 2016 and was base for first Pilot demonstration on IBC in Amsterdam. The release contained components such as:

- ImmersiaTV player for android, windows and android TV
- Premiere Pro extension with mac and windows installers
    - Multiplatform content generation (tablet, HMD, tv)
    - Transitions based on luma masks (with basic interaction).
    - Automatic generation of DASH files.
- Dash conversion and content publication services
- Tutorial explaining how to create scenes covering the main cases of ImmersiaTV, and how to use the ImmersiaTV services
- Session Manager

2.2.6. Release notes for Player

Release version	Date	New or fixed functionalities	Known bugs
0.2	16 Jun 2016	• Works with GStreamer Unity Bridge 1.2 version • Support for Android Cardboard (Android only) • Support for HTC Vive (PC version only) • New contents json input interface • New content and device selection interface • Base XmlEngine Parser for media content • Base JSON Parser for json content file • Shape Types: Rectangle (directive content), SphereCaps (omnidirectional content)	• The app crashes sometimes while displaying high resolution videos • Render issues with high resolution videos on android devices • Videos with audio have unexpected behaviour • On exiting application crashes (audio issues)

		• Display modes: TV, Cardboard, Tablet, HTC Vive • On PC: Cardboard/tablet move around functionality (pressing ALT + moving the mouse) • Back functionality while displaying media • Exit functionality	
0.3	4 Jul 2016	• add: InteractiveObject script and Interactive base actions • add: InputManager and basic mouse and touch managers • add: callback actions funtionality to Spheres and Rectangles (setVisibility, UnsetVisibility, PlayTransition) • add: set InputManager on Hmd scene • add: hmd input manager for hmd scene • add: target selection by shapeId, add mediaObject to playTransition initialization • add: media video extension constant • add: mediaElement as communication layer between gubs • add: ExternalAlphaColor shader to control fade in & fade out • add: fade in functionality to the rendered mediaObjects • add: media element initialization on callbacks • add: set a marker point to hmd scene • modify: set sphere radio with anchors distance • modify: check visibility in a modified shape • modify: camera background color to black • modify: setup black color to texture when start • modify: StreamManager, works without ApplicationManager • modify: material using the new shader and prefabs using the matShaderColorMask material • fix: remove MonoBehaviour from HmdInputManager • fix: adding the delegates to a list and clear the list after delegates are removed • fix: change state of the interactive object only if there is a callback • fix: get the media element if it's not set yet	• Inputs are not working properly on SamsungGear VR • Render issues with high resolution videos on android devices • Videos with audio have unexpected behaviour • On exiting application crashes (audio issues)

0.4	21 Jul 2016	• add: base for user and world frame reference • add: new crosshair marker for Samsung GearVR scene • add: new input controller for Samsung GearVR device • add: set Samsung Gear scenes with new input • add: new tools: OBJExport, script to export scene meshes to obj CubeSphere generator scripts • add: README file describing the scripts inside Tools folder • add: new projection sphere • add: new immersiaTv icon and splassscreen • add: unsetVisibleAction script • modify: SetVisible action to always be true and set the correct action on "unsetVisibility" callback parse • modify: Use CubeSphere in sphereCaps prefabs, texture are seeing smoother • modify: transform the direction vector with transformMatrix and set the new position to the user frameReference • modify: HmdInputManager to accept Samsung Gear input • modify: move OBJExport to Tools folder • fix: missing dll's for Oculus builds • fix: check if the shape Id is the same as the target of the action • fix: calculate the size of the rectangle being the max with the wide of the reference background sphere • fix: the z position of the panel button • fix: apply the new formula to get the output size of the rectangle shapes • Fix: visualization size needs to be multiplied with a magic number • fix: remove the white point of the screen for Samsung GearVR HMD scene • fix: remove the unused buttons from the menu canvas panel • fix: remove warning messages, replacing Application.loadLevelName and logging the exception message • fix: Readme file for tools folder • fix: sphere correct rotation for uv mapping • fix: change rectangle shape's factor (magic number) to see as a baked video • fix: visibility actions comments	• Synchronization not working properly for offline and online content • The App crashes when coming back from video reproduction • Render issues with high resolution videos on android devices • Videos with audio have unexpected behaviour • On exiting application crashes (audio issues)

0.5	17 Aug 2016	No changes in player application	
0.6	25 Aug 2016	• add: new synchronization with mediaTime inside the player • add: new sync type var to switch between dvbcsswc, hack and no_sync in the application manager • add: MVC for the Menu Manager. • add: Load videos in local avoiding server download • modify: save prefabs setup, broken when the project is cloned for the first time • modify: changes in the scenes with the new mvcViewBase • modify: update comments, use defaults local and remote json strings from ApplicationManager • modify: remove spaces and commented lines • modify: apply sync type on SessionManagerBridge and StreamManager scripts • modify: set the content scene to use the Synchro Hack • modify: remove unused isSamsungGear var, go directly to SamsungGearScene without showing the DeviceSelection panel • modify: setup the starting angle for SamsunGear media menu by buttons length • modify: update menu scenes, removing debugmanager gameobject • modify: add features for non overlap android applicaciotion in same device. • modify: add new methods to check looping • modify: access to the ContentTimeManager from the applicationManager • modify: use of the MediaTime and MediaTimeInSeconds from the ContentTimeManager • fix: recover device buttons • fix: make a pause&play to make it work on android the force-sync • fix: checkers of vars that are not null • fix: events using correct time for sync • fix: we dont have to change the sync type if there is not previous content • fix: remove default uri from StreamManager in tablet scene	• Live Streaming issues, doesn't reproduce • Render issues with high resolution videos on android devices • Videos with audio have unexpected behaviour • On exiting application crashes (audio issues)

		• fix: documentation and sync part onStart event invokation • fix: Control if is local and take the absolute path resources • fix: Back button • fix: update SessionManagerClient in submodule to recover the mediaTime • fix: remove MenuManager as singleton	

2.2.7. Release notes of Production Tools

Release version	Date	New or fixed functionalities	Known bugs
0.2	16 Jun 2016	• Product ready for first tests	• the preview is not correct in size • the video format is not named correctly (equirectangular, not spherical) • The panel in this version has to be run in debug version, the installation notes contained special guide how to enable this feature
0.3	4 Jul 2016	• Add interactive transitions effects and keyframes for Premiere Pro Panel ○ Visible: Whether this shape is visible in the scene or not. Hidden shapes can be activate through interaction. ○ Luma matte: Optional mask to use on the texture media. This mask is also a file name (see mediaFile), so it can either be a video or a still image. It uses a "secondary timeline", independent from the mediaFile, since it can be controlled by the application (it can be shorter and loop, or triggered by user actions). It must be a gray image or video. This allows, for example, to have circles with blurry edges shown in the scene, using only rectangular shapes, or transitions. ○ Overlay: A second texture media (with companion alpha mask in overlayMaskFile) which is rendered on top of the mediaFile. It uses the "secondary timeline" (see maskFile). It can be used to make a "beauty mask", embellishing transitions. ○ Overlay luma matte: Alpha mask for the overlayMediaFile.	• If a project is closed and another is opened, Premiere does not update the QE and preserve the old tracks. • When using the QE to get the number of tracks (getActiveSequence()), it stops working if a track is removed from the list. • All video inputs must have the same resolution to render the content.

		○ OnActivate callback: Player method to call when the shape has been "activated". The exact action that triggers the activation depends on the device and the player (could be a "touch" action on a tablet, and "look for more than 3 seconds" on an HMD). The list of available methods is outside the scope of this metadata description. ○ OnDeactivate callback: Player method to call when the shape has been "deactivated". The exact action that triggers the deactivation depends on the device and the player (could be "stop looking" on an HMD). The list of available methods is outside the scope of this metadata description. ○ Separate switch: ○ Switch latitude: Degrees above or below the horizontal plane. ○ Switch longitude: Degrees on the horizontal plane. ○ Switch size: Relative size of the shape. ○ Switch appearence: Texture media (color) to display on this Shape. ○ Switch reference: Reference frame for the Anchor. • Modify visual interface for Premiere Pro Panel ○ Delete the output path box, now it appears when export button is clicked. ○ Export button is blocked if no render or metadata are checked. ○ Modify the refresh button and change his position over the panel. ○ Add information box into the top right of the panel	
0.4	21 Jul 2016	No new features	
0.5	17 Aug 2016	• Premiere pro panel with after effects integration, interactive transitions, sync • Reorder the effect panel. • Delete the "luma matte" checkbox from Premiere Pro Panel. • Renamed portal effect controls: ○ OnStart Callback --> OnStart ○ OnDeactivate Callback --> OnDeactivate ○ OnActivate Callback --> OnActivate ○ Separate Switch --> Extra Portal	• If a project is closed and another is opened, Premiere does not update the QE and preserve the old tracks. • When using the QE to get the number of tracks (getActiveSequence()), it stops working if a track is removed from the list.

		○ Switch Vertical position --> E.P. Vertical degrees ○ Switch Horizontal position --> E.P. Horizontal degrees ○ Switch Size --> E.P. Size ○ Switch Appearence --> E.P. Appearence ○ Switch reference --> E.P. Reference • Rendering with new transformation function • Portal effect size in range 0 to 100% • Visual size control changed on portal effect panel Bugs fixed • In extra portals, the MediaFile name don't show the number of track. • The distance of spherical shapes is between 0.8 to 1. • Ommnidirectional videos will not exported. • Export different tracks in Premiere Pro when there's no interactivity • Now don't generate xml file if no folder is selected.	• All video inputs must have the same resolution to render the content.
0.6	25 Aug 2016	• Add new preset for 16:9 videos. • Fix support over 10 tracks for type selector • Fix problem with transitions that it will export black in 360 videos. • Fix bug that duplicate the distance parameter in XML file. • (0.6.1) Changed the name of tracks exported in XML file, now it will have and extension "sp_" for spereical tracks, "re_" for rectangular tracks and "tv_" extension for TV tracks. • (0.6.1) The number of tracks in XML displayed on premiere pro panel • (0.6.6) Add new presets for Portals and TV	• If a project is closed and another is opened, Premiere does not update the QE and preserve the old tracks. • When using the QE to get the number of tracks (getActiveSequence()), it stops working if a track is removed from the list. • All video inputs must have the same resolution to render the content.

3. INTEGRATION

3.1. Workflow integration

For the Phase 1, only two modules were released and imposed integration issues: **Production tools** and **Player**. Production tools should create content that is possible to be played by the player. However, there were more components and functionalities to be integrated for Phase 1:

- **Capture** - Recording omnidirectional and associated directional content using GoPro and Elmo 360 rigs and traditional cameras. Various testing clips as well as "Dragon Force" movie were shot and edited by Lightbox. The Capture system used in Phase 1 was described in D3.1 Section 4.2.2.1 (omnidirectional cameras) and 4.2.2.3 (directional cameras), and the evaluation of the capture system is available in D4.4 in section 3.1.
- **Stitching** – Combining video images from several cameras was done by Lightbox in Autopano and VideoStitch Studio with assistance of VideoStitch. The off-line stitching system was described in D3.1 Section 4.2.2.2, and the evaluation of the stitching component is available in D4.4 in section 3.2.
- **Encoding** – for Phase 2, partners of the consortium decided to use standard H.264 coding built-in production tools (Adobe Premiere) and defined common parameters to be adapted and used by all the other components related to encoding (stitching, production tools, distribution and reception). The encoding parameters were described in D3.1 Section 4.4.2.1, and the evaluation of the compression functionality is available in D4.4 in section 3.4.
- **Off-line Production Tools** – as mentioned, this component was crucial for Phase 1. It was used for production and synchronization of omnidirectional and directional clips and for preparation the metadata. The architecture and specific workflow of the Phase 1 production tools were described in D3.1 Section 4.3, and the evaluation of the production tools is available in D4.4 in section 3.3.
- **Distribution** – the web server distributing MPEG DASH streams was implemented in order to stream content prepared with using production tools to the client application (Player). The architecture and workflow of distribution server was described in D3.1 Section 4.5, and the evaluation of the distribution server is available in D4.4 in section 3.5.
- **Player** – the end-user application for Windows and mobile phones and VR googles (both on Android) for presentation the streaming content to the user. The evaluation of the player as part of reception and interaction component is available in D4.4 in section 3.6.

The simplified workflow for Pilot 1 is depicted on **Figure 2** and fully described in D3.1 Section 4.1.

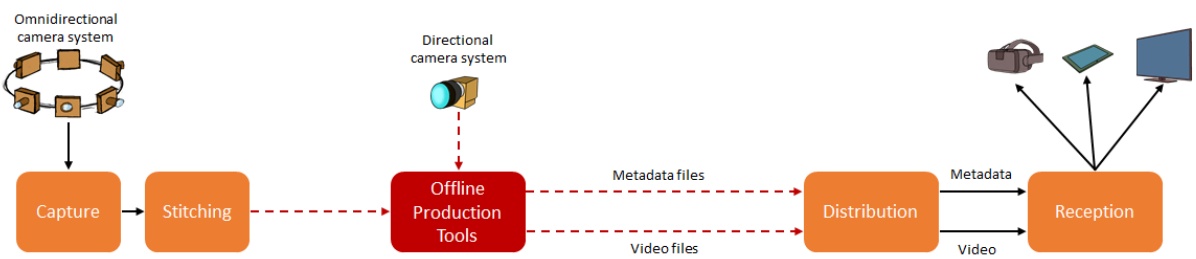

Figure 2: Architecture for ImmersiaTV system for Pilot 1

In all the releases, the workflow and integration of components was tested by functional testing in the software involved and performing qualitative and quantitative tests with media with different resolutions and bitrates.

Modules in the same release were to be compatible, although due to rapid development of new features, the backward or forward compatibility was not guaranteed. On the other hand, the integration of submodules within single module is a constant issue. New features are only merged into component if they don't break existing functionality of a module. This process is depicted on **Figure 3**

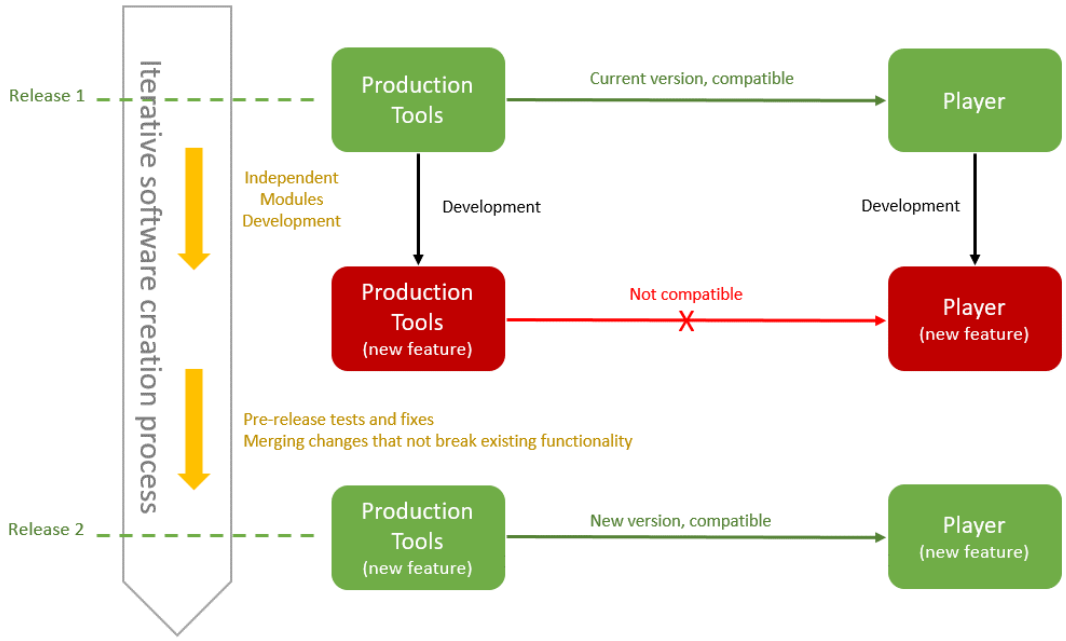

Figure 3: Architecture for ImmersiaTV system for Pilot 1

3.2. Methodology

Development is conducted with **agile approach** in **2 week** sprints. At the beginning of each sprint new tasks are defined – either adding a new feature or modifying existing one. At the end, if the internal tests show the feature to be ready, it is included in the new release.

As the software development process was splitted by several partners, we organized periodically videoconferences in order to define new functionalities and discuss existing problems, we use slack team collaboration tool to resolve doubts and share documentation.

Internal tests are conducted in development environment. Specifics of application makes automatic tests difficult. Developers follow basic usage scenarios defined by requirements, profile applications, monitor available logs and observe parameters, especially quality and synchronization.

Releases are not publicly available and are intended for project partners to use software in real environment and provide feedback. Reported opinions and minor bugs are then addressed in next sprint. Important bugs are fixed as soon as possible.

Management tools: **Slack** team collaboration tool to discuss new functionalities and doubts with the technical team and describing bugs with other partners, **Kanboard** (Kanban project management tool) for fixing bugs, adding new functionalities and following development process.

The packages for release were prepared in a biweekly basis, developing the different functionalities and fixing bugs the first week, testing the solution the second week each piece of software at a time (post-production tool, transcoding and content server and player) and finally integrating all the components to pack the release.

In order to produce a next release (a tag in git version control solution nomenclature) the software has to compile, this code that has to be in a branch is commited to develop version and when it has been extensively tested a new tag is produced, and that code is frozen and assigned a release version number.

The release version number is based in GNU naming structure X.Y.Z where X is the major version and breaks compatibility, Y is the minor version where the code is compatible with the previous minor release version and adds new functionalities and Z is incremented when a bug is resolved for that release.

After committing the tested release the components are packed individually with automatic building tools, with an in-house building solution for post-production software (plugin for Adobe Premiere Pro) and tested in MacOs and Windows, with a docker container for content and transcoder server and Unity3D building environment for unity player.

3.3. End-to-end tests

3.3.1. Internal system tests

Internal system testing was done by development teams during the iterative work on software. Releases were tested by other partners, especially during production of test clips and “Dragon Force” omniditectional movie. Testing team, composed of video producers and editors dealt with two types of testing: the Immersia TV plugin for Premiere Pro and the Dash Server Converter. Both these features were tested in parallel with the pre-production (while we ran tests), production and post-production of Pilot 1. The area of focus was the plugin itself which is an integral part of the production process. The main issues in first releases were related to slowing down the process due to lack of optimization in the plugin. There were also problems with constant preview which made the editing process difficult. In first releases there were also issues with Metadata exportation – not always it was exported in proper way. The Immersia export panel extension also had a couple of problems with refresh and track selection for exportation. Transitions were also troubling to work with at the beginning - using luma matte clips to transition between two omnidirectional images had a complicated logic or organizing and creating a track hierarchy that was counter intuitive to the normal editing thought process inside premiere. These issues arose mainly between pre-production and production and were reported to implementation teams.

Post-production was slightly different at this phase, these initial issues had been somewhat worked out and corrected. An important problem was also having a big project of very high resolutions in demanding and sometimes unstable plugin. At the same time, there were also problems with other the features of the tool - typology of the embedded portals and the process of creating them. All these errors or problems were reported.

Since then, major improvements have been done until version 0.6 - the interface of the plugin and export panel have changed a lot. Luma mattes are no longer needed and everything runs much smoother than previously it did. The process of testing was done iteratively across consequent versions of the production tools, all problems were reported and fixed by development teams.

All major issues related to the player, distribution and production tools were described in Chapter 3 of D4.4 (Technical evaluation).

3.3.2. Testing workflow in demo environment

The Release 0.6 was prepared in order to make Immersia TV system demonstration on IBC fairs in Amsterdam and present first results for Pilot 1. IBC is one of the most known annual event for professionals of audio-video creation, management and delivery content. In 2016 the event was visited by more than 55,000 attendees from 170 countries around the world and was place where 1,600 companies presented their products and services. The Immersia TV demonstration was hosted by European Broadcasters Union (EBU) booth. This demo integrated presentation of two key components developed so far: the android player and production tools and were supported by third component, the web server working in the background and not shown directly during the demo. The Figure 4 depicts the demonstration environment.

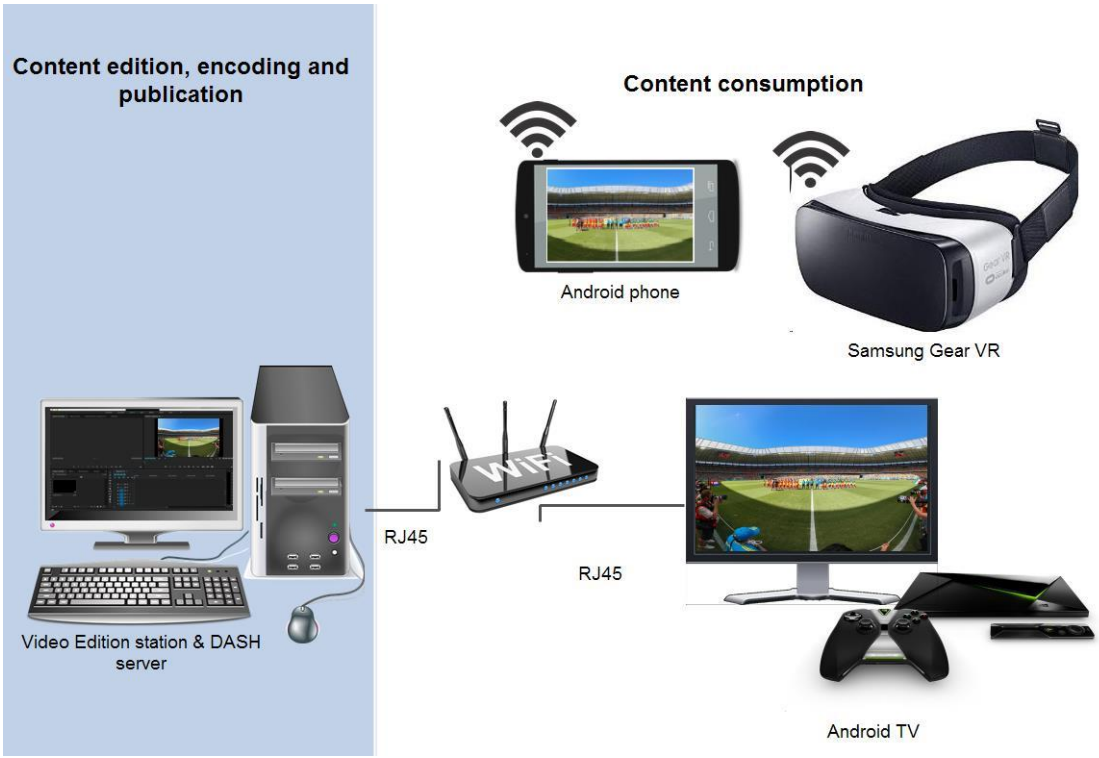

Figure 4: through the immersiaTV services, the content creator can convert his content to DASH and publish the content for synchronized playout during content consumption

The key features demonstrated during IBC were:

- Video-based content delivered synchronously on TV, second screens (tablets) and third screens (virtual reality goggles)

- Portals and video inserts allow defining interactive experiences based on omnidirectional video
- Integration in Premiere Pro for easy content creation

In order to make the demonstration all the components were to be prepared and tested. The "Dragon Force" video clip was prepared and omnidirectional streams were enriched by portals containing directional videos. The result of using production tools were several video clips and metadata file describing relationship between them. They were streamed by local network (WI-FI router) to the end-user devices: TV (standard TV directional clip) and android devices – tablet and Samsung Gear VR (omnidirectional clips). The synchronization between clips was server by Session Manager.

The main technical problems observed during demo were an entry to the discussion and bug fixing process:

- Mobile devices consumes much power during displaying video clips all the day, and this problem must be resolved by constant connection to power suppliers or power banks in order to not run out of battery during presentation.
- On exhibition area there were plenty of wireless networks operating on the same channels (about 30-50 wireless networks with different session id). They interfered to each other decreasing the bandwidth and reliability of the network. The solution is to look for less occupied channel or switch from 2.4Mhz into 5Mhz frequency.
- Several software bugs were observed, especially related to synchronization and previously mentioned problem. They were a subject to further development and bug fixing process.
- Quality, especially this obtained on VR glasses, was not enough for spectators due to problems with network bandwidth in congested and crowded wireless area. Developers need to improve quality in order to present better demonstration of Pilot 1.
- The low-cost devices like google cardboard and Samsung Gear VR have limitations for the human eye to perceive high resolution videos because of the hardware, more high-end devices like HTC Vive and Oculus Rift will give better results.
- We had to check different synchronization mechanisms because first proposed solution (DVBCSS) gave us some problems and took a time to detect how it affected with the different player technologies been used (Gstreamer and Unity3D).

Having a trade-off between synchronization accurate system and high resolution we have to choose frame accurate sync because is what makes ImmersiaTV unique, the software had to adapt to new requirements, processing CPU and GPU limitations on the devices and software updates.

We learned various lessons, mainly that synchronization and different streams does not just depend on the architecture of the solution but has to be extensively tested in different devices. The network was also involved in the tests and the different network negotiations involved in the solution forced us to have a better knowledge of the infrastructure and try to get the best configuration for the testbeds to achieve best results.

4. CONCLUSIONS

The integration of the Immersia TV system which is developed by several partners of the project, require good coordination and communication inside the consortium. Both, the requirements for the system (D2.2 prepared by WP2) and software architecture description (D3.1 prepared by

WP3) need to take into consideration the iterative process of the software development. Requirements on common codec and interfaces between modules and must be perfectly defined in order to avoid integration problems.

Although, both internal testing (done by software developers and testers within teams) and external testing (by other partners making use of the software) are crucial for the development process and bug fixing, good opportunity for finding unexpected problems are demonstrations in production environment. That's why taking part in open pilots, exhibitions and conferences is so important, because it gives a lot of feedback for developers.

A common test and develop environment between partners responsible for components being part of the releases, gives lower response time to bugs and to adapt new functionalities as does a team with version control experience and agile techniques to adopt better strategies when we needed to boost the performance. The process of cooperation and remote collaboration worked pretty well during first releases.

Using external software, such as GStreamer requires specific knowledge and experience. This streaming framework keeps changing all the time, what causes integration problems. That's why I2CAT and PSNC keeps contact to GStreamer community (including taking part in GStreamer workshops) in order to track changes and have an assistance in resolving streaming and synchronization issues.