ARQUITECTURA PARA EL DESARROLLO DE APLICACIONES MULTIPLATAFORMA PARA TELEVISIÓN INTERACTIVA

Wilson Arley Vélez Vargas

Universidad EAFIT

Departamento de informática y sistemas

Escuela de ingeniería

2012

ARQUITECTURA PARA EL DESARROLLO DE APLICACIONES MULTIPLATAFORMA PARA TELEVISIÓN INTERACTIVA

WILSON ARLEY VÉLEZ VARGAS

Tesis de grado presentada como requisito para optar al título de Maestría en Ingeniería

Asesor: JUAN CARLOS MONTOYA MENDOZA, MAGÍSTER.

UNIVERSIDAD EAFIT
DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS
ESCUELA DE INGENIERÍA
MEDELLÍN
2012

Nota de aceptación

Presidente del jurado

Jurado

Jurado

Medellín, 22 de Noviembre de 2012

Dedicado a:

A mi esposa Nany, por impulsarme siempre a ser mejor
Y mi hijo Miguel, por su comprensión por el poco tiempo dedicado.
Los amo.

AGRADECIMIENTOS

A mi esposa Nany por su colaboración y recomendaciones con el documento final y principalmente por su apoyo a lo largo de todo este tiempo de Maestría.

Al Magister Juan Carlos Montoya, mi asesor, por haberme guiado en el desarrollo de esta tesis.

A todo el equipo de Ártica, 'beto', Angee, Daniel y especialmente a ‘Gari’ por su aporte en la documentación y a Daniel Camilo por su ayuda durante la implementación del prototipo.

A Lenin Lozano por sus conocimientos, comentarios y observaciones sobre la arquitectura.

A Ramón Martín de Pozuelo Genís por la ayuda referente al estándar PCF.

A Juan Esteban Maya por las facilidades que me dio en el manejo del tiempo de mi trabajo para poder alternarlo con mi proyecto de investigación.

A Juan Felipe Muñoz por los ejemplos suministrados.

A Wander por los arreglos gráficos de la herramienta y a Eliza por su ayuda con algunas traducciones.

A Carito y a Dany por su ayuda en la evaluación de la arquitectura.

Y a todas las demás personas y entidades que me apoyaron de una u otra forma en alguna etapa de la investigación.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

ABREVIATURAS

ACAP-X/J: Advanced Common Application Platform XML/Java

ADD: Attribute-Driven Design

ADSL: Asymmetric Digital Subscriber Line

ARIB: Association of Radio Industries and Businesses

ATSC: Advanced Television Systems Committee

ATSC-M/H: Advanced Television Systems Committee-Mobile/Handheld

BML: Broadcast Markup Language

CE-HTML: Consumer Electronics - Hypertext Markup Language

CNTV: Comisión Nacional de Televisión

CIERTO: Content Reference Identifier

CSS: Cascading Style Sheet

DAE: Declarative Application Environment

DANE: Departamento Administrativo Nacional de Estadística

DAO: Data Access Object

DASE: DTV Application Software Environment

DAVIC: Digital Audio Visual Council

DDL: Description Definition Language

DOM: Document Object Model

DSM-CC: Digital Store Media Command and Control

DTMB: Digital Terrestrial Multimedia Broadcast

DTV: Digital Television

DVB: Digital Video Broadcasting

DVB-T/C/S/H: Digital Video Broadcasting-Terrestrial/Cable/Satellite/Handheld

DVB-T2: Digital Video Broadcasting-Second Generation Terrestrial

EBU: European Broadcasters Union

EPG: Electronic Program Guide

ES: Elementary Stream

ESG: Electronic Service Guide

ETSI: European Telecommunications Standards Institute

ETV: Enhanced Television

GEM: Globally Executable MHP

HBBTV: Hybrid Broadcast Broadband TV

HD: High Definition

HTTP: Hypertext Transfer Protocol

IoC: Inversion of Control

IP: Internet Protocol

IPTV: Television over IP

ISDB-T/C/S/Tmm: Integrated Services Digital Broadcasting-Terrestrial/Cable/Satellite/Terrestrial Mobile Multimedia

ITV: Interactive Television

JDK: Java Development Kit

JMF: Java Media Framework

JVM: Java Virtual Machine

MHEG: Multimedia and Hypermedia Expert Group

MHP: Multimedia Home Platform

MPEG: Moving Picture Experts Group

NCL: Nested Context Language

NTSC: National Television System Committee

OCAP: Open Cable Application Platform

PAE: Procedural Application Environment

PAL: Phase Alternating Line

PCF: Portable Content Format

PES: Packetized Elementary Stream

POJO: Plain Old Java Object

PPV: Pay-per-view

PVR: Personal Video Recorder

RCP: Rich Client Platform

RMI: Remote Method Invocation

RPC: Remote Procedure Call

SBTVD: Sistema Brasileño de Televisión Digital

SI: System/Signal Information

SMIL: Synchronized Multimedia Integration Language

SOA: Service Oriented Architecture

SOAP: Simple Object Access Protocol

STB: Set-top box

TCP: Transmission Control Protocol

TDT: Televisión Digital Terrestre

UDP: User Datagram Protocol

UI: User interface

UML: Unified Modeling Language

URL: Uniform Resource Locator

URI: Uniform Resource Identifier

VDSL: Very High Bitrate Digital Subscriber Line

VOD: Video On Demand

XHTML: Extensible Hypertext Markup Language

XML: Extensible Markup Language

XSL: Extensible Stylesheet Language

GLOSARIO

CUADROS (FRAME): Es una imagen particular dentro de una sucesión de imágenes que componen un video, es también llamado fotograma .

DECODIFICADOR (Set-Top Box, STB): Dispositivo encargado de demultiplexar y decodificar la señal de televisión análoga o digital (incluyendo las aplicaciones interactivas) que van a ser desplegadas en un televisor.

ENRIQUECIMIENTO: Procesamiento mediante el cual se dota metadatos a un recurso.

METADATOS: son datos que describen a otros datos, los metadatos son altamente estructurados.

VIDEO: Es un sistema de grabación y producción de una secuencia de imágenes que representan movimiento.

RESUMEN

Esta tesis de maestría presenta una arquitectura para la creación de aplicaciones multiplataforma para Televisión Interactiva, en las que las aplicaciones utilizan los objetos contenidos en los videos como generadores de interactividad.

Esta arquitectura fue diseñada siguiendo el método de diseño basado en atributos (ADD) y está dividida en dos componentes principales, una herramienta de creación en la que se seleccionan los objetos y se les enriquece con metadatos y un servicio web que transforma las aplicaciones a una plataforma de televisión digital interactiva. La selección de objetos es acelerada mediante el uso de algoritmos de seguimiento de objetos. Adicionalmente se usan los estándares TV-Anytime y PCF para el almacenamiento de metadatos y para la representación de las aplicaciones, respectivamente.

La arquitectura fue evaluada mediante la implementación de un prototipo donde participaron algunos desarrolladores y con la que se pudo verificar que la arquitectura cumplía con los atributos propuestos durante el diseño.

INTRODUCCIÓN

Muchas cosas pasaron desde Mayo de 1954, cuando se emitió la primera señal de televisión en Colombia entre Bogotá y Manizales, hasta estos días en que los canales nacionales, privados y públicos, realizan pruebas de Televisión Digital Terrestre (TDT) para Colombia y en el que la Comisión Nacional de Televisión (CNTV) aprobó la adopción del estándar DVB-T2[1] (*Digital Video Broadcasting-Second Generation Terrestrial*)

> *"Con la TDT, iniciamos una nueva era tecnológica con alta calidad en los sistemas de transmisión de las señales de televisión, que nos permite acceder a nuevos niveles de servicios mejorando las oportunidades de negocio y entretenimiento"*[2]

Se perfila entonces el surgimiento de un nicho de mercado bastante prometedor para la creación de contenido y el desarrollo de aplicaciones [1] con la entrada de estas nuevas tecnologías de Televisión Digital.

Hasta ahora la televisión se entendía como un flujo inalterable de imágenes [5], pero con la Televisión Digital esta idea tiende a cambiar apelando a uno de los niveles de servicio con más valor agregado, la interactividad. Esta fue creada para brindar diferentes experiencias a los usuarios. El video ahora puede ser aumentado, mejorado y/o enriquecido mediante aplicaciones.

Los responsables de estas aplicaciones se enfrentan ahora a la necesidad de distribuir sus contenidos a múltiples tecnologías para la difusión de sus contenidos, generando costos de creación y mantenimiento adicionales.

---

[1]Colombia. http://www.dvb.org/about_dvb/dvb_worldwide/colombia/. Consultado: 4 de mayo 2012.
[2]Televisión Digital Terrestre. CNTV. http://www.cntv.org.co/cntv_bop/tdt/contenido18.html, 2 marzo 2011.

Esta tesis presenta una investigación en el área de la Televisión Digital. Propone una arquitectura para la creación de aplicaciones que utilicen los objetos contenidos en el video como generadores de interactividad para diferentes plataformas de Televisión Digital, atacando los problemas de portabilidad e interoperabilidad y por lo tanto reduciendo los costos de creación y mantenimiento de las aplicaciones.

CAPÍTULO 1 DEFINICIÓN DEL PROBLEMA

1.1 ESTRUCTURA DEL DOCUMENTO

En los siguientes numerales de este capítulo se describe el contexto actual para el desarrollo y despliegue de aplicaciones interactivas en múltiples plataformas, también se formula el problema a resolver, además de los objetivos que se quieren alcanzar y las razones por las cuales esta arquitectura resulta de utilidad.

Los capítulos siguientes de esta tesis están estructurados de la siguiente forma:

El capítulo 2 está formado por dos grandes secciones. La primera es la exploración del estado del arte sobre los trabajos realizados por diferentes autores en las áreas de: televisión interactiva usando el contenido del video como medio de interacción y de portabilidad e interoperabilidad de dichas aplicaciones. Esto para entender de una mejor manera la forma en que otros investigadores se han aproximado a solucionar estos problemas. Y la segunda parte resume el fundamento teórico de esta investigación presentando los conceptos y tipos de televisión digital, aplicación interactiva y *middleware*. También se estudian los diferentes estándares de metadatos involucrados en la solución y se da una vista general del seguimiento de objetos. Además se dan nociones de arquitectura de software y de los métodos de evaluación de las mismas, ambos necesarios para comprender la metodología usada en este proyecto.

En el capítulo 3 se presenta de una manera detallada a través de una serie de vistas el diseño de la arquitectura propuesta resultante de aplicar técnicas de diseño basadas en atributos o ADD (*Attribute-Driven Design*) [57].

El capítulo 4 describe el proceso de la implementación de un prototipo de la arquitectura. También explica las implementaciones de las diferentes herramientas de dibujo, de los algoritmos de seguimiento y de las plataformas de DTV.

En el capítulo 5 se evalúa la arquitectura en cada uno de los escenarios de calidad más importantes usando el prototipo implementado y aplicando una serie de métricas arquitectónicas. Las pruebas van a acompañadas de los resultados obtenidos y del análisis de los mismos.

En el 6 y último capítulo se relatan las conclusiones de esta tesis y hace algunas recomendaciones sobre posibles mejoras y trabajos futuros propuestos para la continuación en esta línea de investigación.

1.2 DESCRIPCIÓN Y FORMULACIÓN DEL PROBLEMA

La llegada de la Televisión Digital (DTV) está causando grandes cambios en la industria [2]. La DTV cada vez se está convirtiendo más en un servicio ubicuo, desplegándose en cualquier lugar, a cualquier hora y en cualquier dispositivo, entre los que se cuentan televisores, computadores y dispositivos móviles entre otros [6][22][30][35][39]. De igual forma, ésta puede ser transportada por diferentes plataformas de transmisión, como por ejemplo cables, satélites, redes terrestres, líneas telefónicas e Internet [2][12][22][23][40].

Una de las principales características de esta nueva TV es la interactividad, dando paso a la llamada Televisión Interactiva o ITV (del inglés *Interactive TV*), con la que se pretende convertir al usuario en un ente más activo [2] mediante el enriquecimiento de su experiencia. Debido a esta nueva propiedad de la televisión la tendencia de los nuevos productores es diseñar contenido interactivo y ofrecerlo en forma de aplicaciones acompañando sus transmisiones y/o programas de televisión.

Hoy en día, un aspecto importante de la mayoría de las aplicaciones para ITV es que son diseñadas para usuarios versados en el uso de computadores, parecen formularios o sitios web superpuestos al video [8], como se puede ver en la Figura 1, debido a que las

interfaces de usuario hacen uso de elementos derivados de los ambientes de trabajo con computadores, como botones, iconos y enlaces [6] entre otros. Pero esto no debería ser así, los televidentes están interesados en disfrutar su tiempo libre y no en gastarlo entendiendo complicadas interfaces como si estuvieran trabajando. En este sentido el desarrollador enfrenta retos tales como la ubicación del televidente con respecto a la pantalla, la limitación en resolución de pantalla y sus diferentes aspectos, la escasa habilidad para desplegar imágenes, el área donde es seguro mostrar información, las diferencias en el tamaño y en el color de las gráficos y la tipografía entre otros [30][51].

uk sky - sky music channels

TV3, Catalonia's Television – European Football

Figura 1 Ejemplos de aplicaciones ITV típicas

Otro aspecto para tener en cuenta de las aplicaciones para ITV es el grado de relación con el video, la mayoría de estas no están estrechamente relacionadas con el contenido desplegado. No tienen en cuenta el movimiento, las acciones, los personajes o la historia, esto debido en gran medida a que no permiten una manipulación del flujo de video o de los elementos contenidos en él [7].

Para crear las aplicaciones interactivas de las que se ha hablado, los productores utilizan paquetes de *software* llamados herramientas de creación o de autor, más conocidos por su nombre inglés como *Authoring Tools*. Estas herramientas han sido diseñadas para

ambientes o estándares específicos como las propuestas en [5] y [20], sus interfaces hacen uso de los cuadros del video y de líneas de tiempo como paradigma de edición y no permiten usar el contenido como medio de interacción [7].

Como se muestra en la Figura 2, una vez la aplicación está desarrollada, el productor de contenido o el desarrollador desea desplegarlas en diferentes medios, como TDT, cable y satélite, entre otros, enfrentando problemas de **portabilidad** [12][41] y de **interoperabilidad** [31][32][33][34][39][41] con sus aplicaciones**,** debido al gran número de estándares involucrados, tanto en el desarrollo de aplicaciones como en la representación del contenido o de los metadatos asociados al video [35][36].

En consecuencia, el productor o desarrollador debe construir y mantener códigos fuentes diferentes para cada uno de los estándares [35]. De igual forma debe hacerlo con los metadatos.

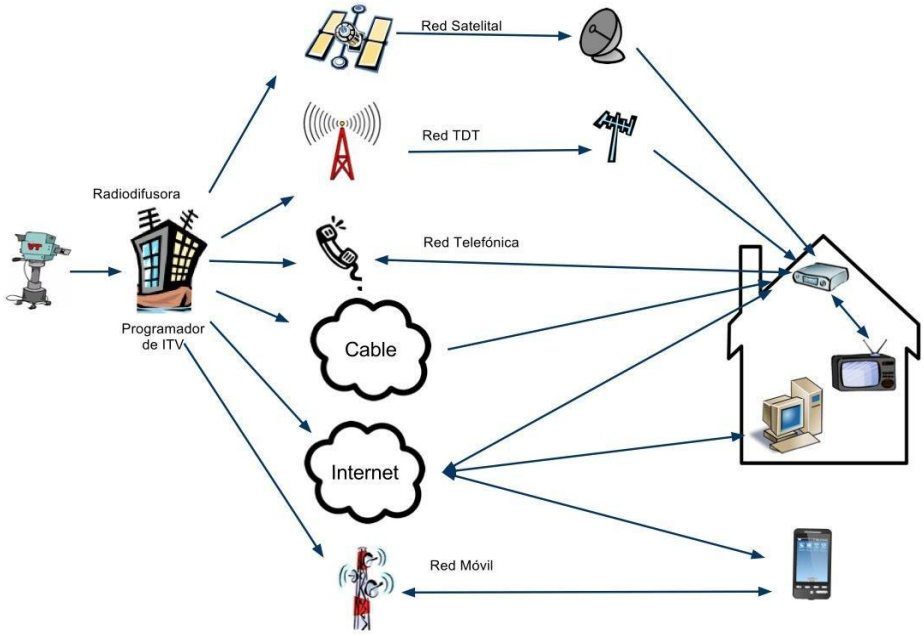

Figura 2 Transmisión de Televisión Digital

Según la problemática expuesta anteriormente, los nuevos productores de contenido y de aplicaciones de televisión interactiva se ven en la necesidad de diseñar interfaces relacionadas con el contenido de los videos que se puedan desplegar sobre cualquier plataforma de DTV [32][33].

Teniendo en cuenta lo enunciado en los párrafos anteriores el autor de este documento se formula el siguiente interrogante:

¿Qué arquitectura facilita el desarrollo de aplicaciones multiplataforma para Televisión Interactiva que permita el uso de los objetos embebidos en el flujo de video como mecanismos de interacción con el usuario?

1.3 OBJETIVOS

1.3.1 Objetivo General

Diseñar una arquitectura para el desarrollo de aplicaciones multiplataforma para Televisión Interactiva que permita el uso de los objetos embebidos en el flujo de video como mecanismos de interacción.

1.3.2 Objetivos Específicos

- Identificar los componentes encargados de la manipulación de los objetos contenidos en el video.
- Establecer el modelo de metadatos para transportar el contenido descriptivo del video y el de sus objetos a múltiples plataformas.
- Establecer el estándar o lenguaje para describir la funcionalidad de las aplicaciones interactivas.
- Valorar la viabilidad y efectividad de la arquitectura para el desarrollo de aplicaciones interactivas para televisión digital multiplataforma.

1.4 JUSTIFICACIÓN

La organización DVB anuncia que la llegada de la TDT a Colombia impulsará al sector audiovisual, alentando la producción de contenido dinámico y fortaleciendo la producción nacional. Razón por la cual, se espera que haya una buena inversión en el talento nacional en todas las capas del ecosistema de televisión digital, pero sobre todo en las creativas y las tecnológicas. Esto se traduce en nuevos retos para la academia y para la industria, que tienen la necesidad de adoptar y adaptar nuevos conocimientos y tecnologías, generar emprendimiento, generar nuevos contenidos y desarrollar nuevas aplicaciones entre otras.

En términos de los beneficios esperados, este proyecto permitirá el mejoramiento de algunos servicios y/o aplicaciones interactivas, brindando herramientas para diseñar interfaces con usos más naturales para los usuarios.

Otro beneficio esperado de esta arquitectura es el mejoramiento en términos de los recursos necesarios para producir, desplegar y mantener aplicaciones para ITV en diferentes plataformas de DTV debido a que se pretende no tener que repetir totalmente la implementación para cada una. De igual forma se espera eliminar la redundancia de metadatos para cada sistema.

La realización de este proyecto contribuirá a la evolución del Grupo de Investigación, Desarrollo e Innovación en Tecnologías de la Información y las Comunicaciones (GIDITIC) de la Universidad EAFIT. Este grupo realiza proyectos, que facilitan la incorporación de conceptos y tecnologías que aporten a la comunidad científica del país. Aunque este grupo ya ha desarrollado proyectos con Televisión Digital, esta propuesta pretende brindar nuevos elementos de investigación en el área de ITV con una nueva arquitectura que podrá ser utilizada en ambientes de producción y emisión de televisión.

Finalmente este proyecto se presenta como requisito indispensable para obtener el título en Maestría en Ingeniería de la Universidad EAFIT por parte del autor.

CAPÍTULO 2 MARCO DE REFERENCIA

2.1 REVISIÓN DE LITERATURA Y/O ANTECEDENTES

A continuación se presenta una revisión de los antecedentes más relevantes encontrados durante la etapa de levantamiento de información, estos están divididos en dos campos, el primero tiene que ver con la implementación de interfaces que usan los objetos del video y el otro campo tiene que ver con los atributos de portabilidad e interoperabilidad de las aplicaciones de ITV.

2.1.1 Objetos del Video

Los trabajos relacionados con las interfaces que usan los objetos del video se pueden dividir en tres grupos, según la plataforma donde se despliegan: Televisión Digital, Video en computadores personales e Internet TV

2.1.1.1 Televisión Digital

- *LiveLink System* [11] es una implementación comercial de TeleClix para ITV, específicamente para la Plataforma Multimedia para el Hogar (MHP) que usa el seguimiento de objetos como fuente de interacción. Su solución es mediante *hardware*, consiste en poner sensores en los objetos y o personas a rastrear y un equipo pegado a las cámaras que detecta y registra la posición espacio-temporal de los sensores. Esta información es enriquecida y enviada con datos adicionales previamente adquiridos. Muy útil para transmisiones en vivo.

- GMF4iTV [9] *Generic Media Framework for* ITV es el más completo de los proyectos evaluados, su propósito fue desarrollar técnicas para crear programas de televisión con objetos interactivos [10]. El prototipo fue una plataforma de

extremo a extremo que proveía interactividad a nivel de los objetos. Permitía a los proveedores de contenido y servicios crear, administrar, sincronizar y distribuir flujos de video lineal pregrabados en conjunto con contenido adicional no lineal empleando esquemas de metadatos. Hace uso de técnicas avanzadas de codificación de video (MPEG-2, MPEG-4), de metadatos (MPEG-7), y modelado semántico mediante ontologías y procesamiento de reglas usando un motor de inferencia. Las aplicaciones cliente resultantes tienen dos componentes una que es implementada en MHP y que es radiodifundida y la otra que se ejecuta en dispositivos móviles y es transmitida por banda ancha.

2.1.1.2 Para edición de video en computadores personales:

- Goldman, Gonterman, Curless, Salesin y Seitz [7] proponen un *framework* que usa el movimiento de objetos de video como generadores de interacción. Usan técnicas de visión por computador para entender el movimiento de varios puntos en una escena y luego los segmentan en objetos independientes. En particular proponen 3 tareas a realizar, anotación, navegación y composición de imágenes. Este *framework* es para el desarrollo de aplicaciones interactivas para computadores.

2.1.1.3 Para internet TV: El conjunto de herramientas para video interactivo en Internet se destacan por que su principal funcionalidad es la creación de hipervideo y su uso es el de la publicidad y el comercio electrónico. Su principal desventaja es que cada uno define su propio reproductor y su propio formato de interactividad.

- Adivi ProductionKit [15], Quick.tv [16], Overlay.TV [17] y WireWAX [18] permiten definir regiones sensitivas en cualquier escena, área u objeto en el video como punto de entrada para anotar o desplegar información adicional o para ir a un enlace dentro o fuera del video.

- VideoClix [19] es el mejor ejemplo de esta gama. Es una solución de extremo a extremo para distribución de video interactivo pensado para proveedores de contenido y anunciantes. Su editor usa seguimiento de objetos para detectar

automáticamente personas, lugares y productos con lo que crea puntos sensibles que son usados por los usuarios para interactuar. Incluye un administrador de contenido para los objetos y un administrador de campañas. Permite desplegar el video en diferentes plataformas (web, móvil y TV).

2.1.2 Portabilidad e Interoperabilidad

En los párrafos siguientes se describen los proyectos relacionados con la portabilidad e interoperabilidad de aplicaciones ITV:

- Rodríguez-Alsina, en [12] describe un flujo de trabajo para desarrollar y mantener aplicaciones para televisión interactiva para Web usando el estándar de Formato de Contenido Portable o PCF [25].

- *The Cossack System* [31] describe una plataforma para ITV. Comprende una estructura de extremo a extremo que reemplazaría a los sistemas actuales. Es un administrador de servicios que ofrece control de la interacción entre los usuarios finales y los proveedores del servicio. Define el concepto de Hiperprograma, en el los programas de televisión son mejorados con enlaces a servicios relacionados con una o varias imágenes o a todo el programa. No hay restricción sobre el tipo de clientes que pueden ejecutar los servicios. Es un esfuerzo académico importante pero casi imposible de implementar en la industria al querer reemplazar a los actores actuales.

- Tsekleves y Cosmas [35] presentan una metodología para la creación de servicios interactivos. La innovación de esta propuesta está en la creación de interfaces independientes del lenguaje en el que se va a desplegar, propone el uso de herramientas de edición de video muy conocidas, como Macromedia Director, para generar vistas que luego son descritas usando formatos XML. Estos formatos luego son presentados en otras herramientas que permiten hacer la traducción a lenguajes propietarios.

- En [39] se propone un estándar para el desarrollo de aplicaciones en las plataformas DTV, Web y móvil, dentro del contexto de Brasilero. Se centra en el reuso del contenido, propiciando la adaptación de este a las diferentes plataformas. Hace uso de los tipos de datos comunes a todas las plataformas. Se optimiza la producción y el mantenimiento del contenido al ser único y se reduce la necesidad de espacio.

- El proyecto BEACON [40] define una arquitectura genérica para garantizar interactividad basada en MHP Ejecutable Globalmente (GEM) que permite que las aplicaciones sean independientes del estándar de transmisión. Incluye la definición de un navegador que corre en el decodificador, siempre y cuando este soporte GEM. Este navegador traduce o interpreta un XML definido por el proyecto y creado mediante su propia herramienta de autor. El problema de esta propuesta es que aunque la mayoría de estándares están basados en GEM no todos los decodificadores lo soportan. Otra problema es la interpretación del XML en el cliente ya que esta es una operación costosa computacionalmente.

- Song y Park [41] proponen en teoría una arquitectura integrada para la transmisión de datos basada en componentes. Integran las características de cada *middleware* actual a los componentes diseñados. Definen perfiles que proveen información específica de cada *middleware* así los receptores pueden manejar las aplicaciones basadas en otras especificaciones. Si se implementara esta arquitectura la ejecución de aplicaciones podría ser lenta debido a la interpretación de los perfiles.

2.1.3 Conclusión

En conclusión no existe una solución que integre todos los conceptos planteados en la descripción del problema, la utilización de los objetos del video como medio de interacción y la portabilidad e interoperabilidad de las aplicaciones de ITV. Los mejores proyectos exponentes de cada concepto son GMF4iTV [9] y BEACON [40]. Adicionalmente en algunas de las soluciones planteadas para resolver los problemas de portabilidad e interoperabilidad mantienen un límite muy borroso entre ambos conceptos [12][39], no queda muy claro la diferencia y la forma de resolver cada uno .

2.2 MARCO TEÓRICO

2.2.1 Televisión Digital - DTV

Los detalles técnicos de esta tecnología como lo son los tipos codificación de audio y video, estándares de compresión, multiplexado, modulación, por mencionar algunos, están fuera del alcance de este proyecto (Para leer sobre los aspectos técnicos de la DTV ver Collins [46]), por lo que este documento se concentrará en aspectos de alto nivel que sirvan para entender conceptos más abstractos y de capas superiores como lo son aquellos relacionados con la interactividad.

La DTV usa técnicas de codificación digital para todo el ciclo de vida de la televisión, desde la producción, la transmisión hasta la recepción [28]. Al ser digital ésta permite la adición de datos al flujo normal de audio y video. Lo que se transmite es una señal de datos digitales modulados sobre una señal portadora análoga [2].

Sus orígenes fueron la necesidad de conversión entre estándares análogos, por ejemplo de NTSC (*National Television System Committee*) a PAL (*Phase Alternating Line*), la gran cantidad de ruido existente y la necesidad de compresión [27].

Por el solo hecho de ser digital tiene ventajas sobre la análoga [2][28]**.** Una que ya se viene disfrutando desde hace tiempo, así los operadores aun no transmitan en digital, es el manejo del contenido en este formato evitando muchos problemas con el almacenamiento, ocupa menos espacio, se puede catalogar más eficientemente, puede ser encriptada y respaldada, entre otros.

Durante la transmisión [2][27], desde el punto de vista técnico, es más versátil permitiendo un mejor uso del espectro. La representación numérica permite el uso de compresores, filtros digitales y es menos sensible a errores, desde el punto de vista del consumidor estas características técnicas se traducen en: más canales, mejor calidad de imagen (Alta

resolución, sin interferencias), un formato de pantalla más amplio (16:9), mejor calidad de sonido (Sonido envolvente, varias pistas para diferentes idiomas), televisión personalizada (Grabadoras personales de video o PVR, Pague por ver o PPV y Video por demanda o VOD) y la más atractiva de todas, el factor diferenciador, ofrece servicios de datos adicionales y aplicaciones que no son posibles desde la tradicional tecnología análoga (Guías electrónicas de programación, servicios móviles, aplicaciones interactivas).

2.2.2 Estándares y Tipos de DTV

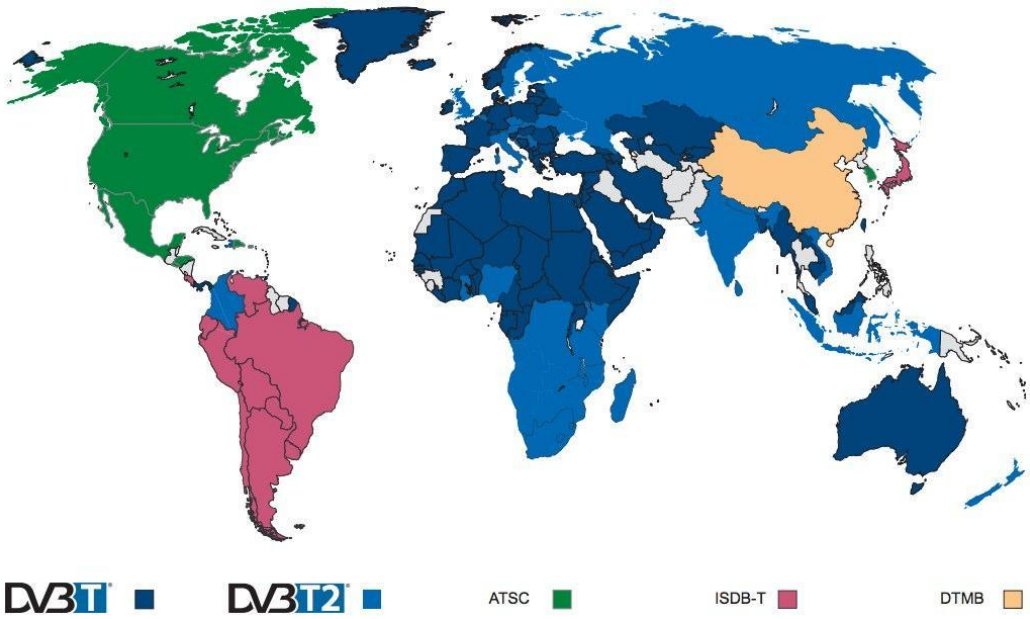

Figura 3 Sistemas de TDT.[3]

Desafortunadamente la DTV heredó la fragmentación de especificaciones, tal como se puede ver en la Figura 3, dando paso a un modelo desigual de transmisión [2]. A continuación se presenta una lista de los diferentes estándares [2][28]:

[3] DVB Worldwide. http://www.dvb.org/about_dvb/dvb_worldwide/DVB-T_map.pdf. Consultado: 5 de mayo de 2012

- DVB-T (*DVB-Terrestrial*). Estándar Europeo para transmisión terrestre.
- ATSC (*Advanced Television Systems Committee*). Estándar desarrollado en Estados Unidos para transmisión terrestre. Actualmente también usado en Canadá y Corea del Sur.
- ISDB-T (*Integrated Services Digital Broadcasting, Terrestrial*). Estándar Japonés para transmisión terrestre.
- SBTVD (Sistema Brasileño de Televisión Digital) o ISDB-Tb. Es una adaptación a la norma Japonesa. Ha sido ampliamente aceptado en los países de la región.
- DTMB (*Digital Terrestrial Multimedia Broadcast*). Desarrollado en China.
- DVB-C (DVB-Cable), OpenCable y ISTB-C. Son algunos de los estándares usados para la transmisión por cable.
- DVB-S (DVB-Satélite), A/80 y A/81 de ATSC y ISDB-S. Son algunos de los estándares usados para la transmisión satelital.
- DVB-H (*Handheld*), ATSC-M/H (*Mobile*/*Handheld*), ISDB-Tmm (Multimedia Móvil Terrestre) y DTMB. Se usa para las redes celulares.
- HBBTV (*Hybrid Broadcast Broadband TV*). Es un nuevo estándar desarrollado en Europa que combina servicios de radiodifusión y de banda ancha. Puede ser usado sobre diferentes tecnologías.

Dos casos especiales de DTV son la Televisión sobre IP o IPTV y la Televisión por Internet o Internet TV, aunque sus nombres son muy similares son productos diferentes. A diferencia de los otros tipos de DTV los canales no son difundidos en su totalidad para optimizar el ancho de banda, solo es transmitido el canal seleccionado por el televidente [23].

La IPTV se ha desarrollado sobre las técnicas de reproducción continua o *streaming*, usando las líneas telefónicas ADSL (*Asymmetric Digital Subscriber Line*) o VDSL (*Very High Bitrate Digital Subscriber Line*) [8][23]. Esta tecnología se da en redes cerradas, lo que le permite a las empresas brindar mejor servicio y modelos de negocios similares a los de la televisión por cable.

La Internet TV es la distribución de servicios de televisión vía Internet. Existen 2 formas: el usuario selecciona el programa que desea ver de un archivo de programas o el canal desde una página web y este es distribuido por *streaming* a un reproductor multimedia. La otra forma es la descarga del programa directamente al computador.[4]

2.2.3 Televisión Interactiva - ITV

Al combinar datos binarios en la misma señal de video y audio [22], se hace posible llevar aplicaciones multimedia para ser ejecutadas en un televisor digital o en un STB. Adicionalmente, la convergencia [3][22] entre diferentes infraestructuras tecnológicas como la radiodifusión, la televisión digital, las telecomunicaciones e Internet, han hecho posible incorporar una vía de retorno al tradicional canal unidireccional de televisión.

La definición más acorde con el propósito de esta investigación es la presentada por Lu [30] y la Unión Europea de Radiodifusores (EBU) [21] que dicen que la ITV es el uso de servicios mejorados o de algún tipo de interactividad incorporados a la DTV.

La ITV tiene las siguientes propiedades, en términos de experiencia del usuario: 1) Mezcla de videos fijos (preeditados) que tienen una narrativa lineal 2) los niveles de entrada del usuario van de bajos a leves, y 3) se emplean gráficos dinámicos principalmente para recubrir el video.

2.2.3.1 Tipos de Aplicaciones ITV. Los nuevos servicios y aplicaciones son probablemente las características más interesantes para muchos espectadores y para la industria. La transmisión digital puede ofrecer muchos tipos de servicios que simplemente no son

[4] Internet Television. Wikipedia. http://en.wikipedia.org/wiki/Internet_television. Consultado: 4 de mayo de 2012

posibles con la analógica. Dentro de los que se pueden incluir información adicional en el flujo para mejorar la experiencia de ver televisión, o puede incluir aplicaciones descargables que permiten a los espectadores interactuar con su televisor de nuevas maneras. Estas aplicaciones pueden ser simples mejoras a la televisión actual como subtítulos en varios idiomas o las Guías Electrónicas de Programas (EPG) que muestran una mejor información de la programación. Puede haber mejores servicios de información como noticias, o información vinculada a los programas específicos (biografías del elenco, o las estadísticas en una transmisión deportiva). Otros de estos servicios pueden ser en áreas tales como el comercio electrónico, anuncios interactivos o aplicaciones de participación.

En [8] y [38] se hace una clasificación según el grado de interacción o su relación con el contenido.

Según el grado de interacción son:

- **Locales**: No necesitan canal de retorno, por ejemplo, el teletexto o algunas aproximaciones a VOD. También caen en esta clasificación las funciones que ofrecen algunos televisores como cambiar la relación de la pantalla (4:3 o 16:9), seleccionar los canales de sonido (mono, estéreo) y hasta activar subtítulos y *closed caption.*
- **Simples**: Integran un canal de retorno, por ejemplo compras, juegos y PPV.
- **Completas**: Incluyen video conferencia

Según su relación con el contenido son:

- **Servicios Permanentes o dedicados**: Servicios que no están directamente relacionados con la programación transmitida. Incluyen servicios con información disponible permanentemente como el clima, titulares, noticias de deportes y entretenimiento y ***front-ends*** para *e-Government, e-Learning* o *e-Health*.

- **Aplicaciones para ETV**: Son aplicaciones que directamente mejoran la experiencia de los usuarios de un programa o evento. Puede ser tan simple como capas de texto o complicados como escenarios con gráficos 3D que se sobreponen a los programas con los cuales los televidentes pueden interactuar.

De [2][4][30][38] se puede tomar una lista con casos concretos:

- **Servicios de Información:** EPG, ESG, Noticias/Eventos, Pronóstico del clima, Servicio de tráfico y *Digitext/teletext*.
- **Servicios de Comunicación:** *T-Mail, T-Chat.*
- **Servicios de Entretenimiento:** *T-Games*, VOD.
- ***T-Commerce:*** *Tele-Shopping*, Publicidad Interactiva.
- ***T-Government:*** Portales de Información Regional, Votaciones.
- ***T-Learning***
- ***T-Health / T-Care***

2.2.3.2 Edición de Aplicaciones. Como lo cita Lu [30], para que un servicio interactivo tenga éxito es necesario un diseño interdisciplinario de especialidades como diseño gráfico, interacción hombre-maquina, etnografía, entre otros. Y se resalta mas esta necesidad ya que la interactividad debe estar concebida desde el nacimiento de los programas, en conjunto con los escritores y productores .

Herramientas de autor multimedia

El termino es muy general y hace referencia a un programa que permite crear contenido u otros programas[5] para ser entregados a usuarios finales. Se puede categorizar basado en 1) Quien lo va a usar? 2) funcionalidad y 3) paradigma que usa [3].

[5]Authoring System. http://en.wikipedia.org/wiki/Authoring_system, 23 feb 2011

La funcionalidad de estas herramientas incluye la agregación de objetos de diferentes medios, como subtítulos e imágenes y un mecanismo para manejar la interacción con el usuario [3]. Adicionalmente algunas permiten adicionar información semántica para describir el contenido [36] [37].

El resultado es una estructura multimedia sincronizada con una distribución espacial del contenido y con enlazados a acciones y otros materiales [36].

2.2.4 Middleware

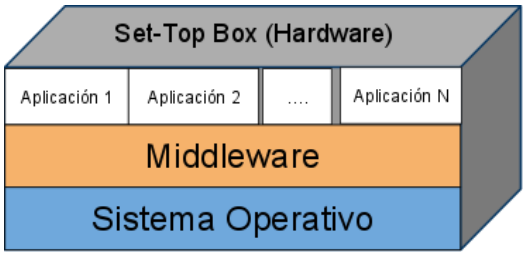

Figura 4 Arquitectura Básica de las capas de *software* de un STB. Adaptado de la figura 1 de [22].

El *middleware* es una capa de *software* que corre sobre el sistema operativo de los decodificadores, ver Figura 4. Su propósito es proveer una interfaz estándar para la ejecución de aplicaciones [2][4] independiente de la marca del STB [23]. Define las funciones que permiten enviar y recibir datos de programas, estos datos pueden ser aplicaciones multimedia, aplicaciones interactivas o contenido de Internet. También define el lenguaje de programación en el que deben ser implementadas las aplicaciones [20].

El *middleware* puede ser procedimental o declarativo, propietario o abierto. Las soluciones propietarias [2][22][23] van de la mano de grandes empresas de la industria, por ejemplo: OpenTV Core de OpenTV (ahora NAGRA) [6], MediaHighway de NDS [7] y Microsoft

[6] OpenTV. http://www.nagra.com/dtv/products-and-solutions/client-technologies/opentv-middleware/ Consultado: 4 de mayo de 2012

[7] *MediaHighway Set-top Box Software*. http://www.nds.com/solutions/mediahighway.php, Consultado: 4 de mayo de 2012

Mediaroom de Microsoft[8]. Entre los abiertos se encuentran: “Multimedia Home Platform” (MHP)[9], “OpenCable Applicaction Platform” (OCAP)[10], “Advanced Common Application Platform” (ACAP)[11], “Application Execution Engine Platform for Digital Broadcasting” (ARIB B23) y “Broadcast Markup Language” (BML) (ARIB B24)[12], Ginga[13]. Los estándares abiertos son usados por que garantizan que los sistemas sean capaces de trabajar juntos sin importar el fabricante [2][22]. La mayoría de estándares incluyen dos tipos: los basados en lenguajes declarativos o de scripting y los basados en lenguajes procedimentales.

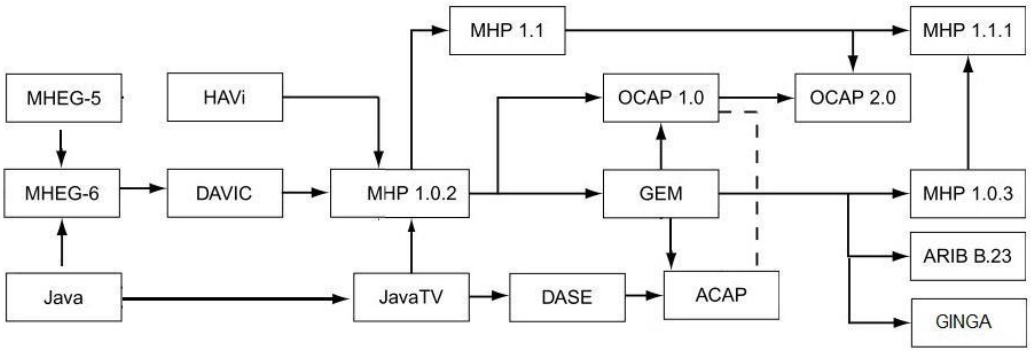

Figura 5 Evolución y relación entre los diferentes *middlewares* procedimentales. (Adaptado de las figuras 1.2 y 1.4 de [2]).

En la Figura 5 se muestra la evolución y relación de los *middlewares* procedimentales y a continuación se da una breve explicación de los mas relevantes.

2.2.4.1 MHEG (Multimedia and Hypermedia Experts Group). El Grupo de Expertos en Multimedia e Hipermedia de la ISO definió un estándar para almacenar, intercambiar y visualizar presentaciones multimedia. Además especifica el formato de intercambio de

[8]Microsoft *Mediaroom*. http://www.microsoft.com/mediaroom/ Consultado: 4 de mayo de 2012
[9]*Multimedia Home Platform*. http://www.mhp.org, Consultado: 3 de Mayo de 2012
[10]*CableLabs OpenCable.* http://www.cablelabs.com/opencable/, Consultado: 4 de mayo de 2012
[11] ATSC. A/101: Advanced Common Application Platform (ACAP) http://www.atsc.org/cms/index.php/standards/published-standards/72-atsc-a101-standard, Consultado: 4 de mayo de 2012
[12]ARIB. *List of ARIB Standards in the Field of Broadcasting.*
[12]http://www.arib.or.jp/english/html/overview/sb_ej.html. Consultado: 4 de mayo de 2012
[13]Ginga. http://www.ginga.org.br, Consultado: 4 de mayo de 2012

objetos entre dispositivos. Fue creado como el *middleware* de servicios digitales en el Reino Unido. Es un lenguaje declarativo orientado a objetos con clases predefinidas y herencia [22]. Una aplicación MHEG está formada por un *script* y grupos de escenas que incluyen objetos interactivos como botones, cursores y similares; enlaces que definen los procesos producidos por las interacciones del usuario con los objetos interactivos y objetos de presentación controlados por los enlaces [47][23]. La versión MHEG-5 se especializa en decodificadores con procesamiento y memoria limitada [23].

2.2.4.2 DAVIC (Digital Audio Visual Council). Es una asociación que promueve la interoperabilidad entre equipos audiovisuales [4]. Está basado en la versión 6 de MHEG que le adiciono soporte a Java, creando una API (*Application Programming Interface*) desde la que se podían manipular objetos MHEG. DAVIC desarrollo APIs adicionales que permitían acceder a algunos sistemas de información, controlar la presentación de audio y video, y administrar los recursos del STB [2].

2.2.4.3 JavaTV. Sun desarrolló una plataforma puramente Java para DTV. Parte del trabajo consistió en unificar las APIs existentes como *Java Media Framework* (JMF) y DAVIC. También definió nuevos componentes como un nuevo modelo de aplicaciones, APIs para acceder a funcionalidad específica de DTV, y una arquitectura donde estos elementos pudieran trabajar juntos [2]. Además las aplicaciones son independientes de la tecnología de la red de transmisión [4][28][47].

Las aplicaciones Java que corren en los STB son llamados Xlets. El concepto es similar a los *applets* aunque tienen un ciclo de vida diferente y adiciona un componente, el administrador de aplicaciones, que lo controla [47].

2.2.4.4 DASE (DTV Application Software Environment). Es la especificación para la transmisión de datos para el estándar ATSC [47]. Está pensado para operar en receptores de alta gama. Consta de dos ambientes. Ambiente de Aplicaciones Procedimentales (PAE) basado en Java y Ambiente de Aplicaciones Declarativas (DAE) basado en XHTML 1.1. Ambos comparten los decodificadores de contenido y el sistema de seguridad [28]. PAE sigue el ciclo de vida definido por JavaTV [47].

2.2.4.5 MHP Multimedia Home Platform [10][13]. Es el *middleware* abierto para brindar interactividad en los diferentes estándares de transmisión de DVB (T/S/C/H) [28]. Es el resultado de experiencias pasadas y alternas en el campo de DTV, como DAVIC, JavaTV y HAVi (*Home Audio Video Interoperabilitiy*) [4]. La pila de protocolos define un sistema de transporte, un ambiente de ejecución y un conjunto de APIs para el desarrollo y transmisión de aplicaciones ITV [40][48]. Permite llegar a una amplio número de receptores de todas las gamas [48].

El sistema DVB introdujo el concepto de perfiles como apoyo a la implementación de estándares. Cada perfil hace referencia a un área de aplicación específica y define los requerimientos del STB necesarios para su soporte [2]. Existen cuatro perfiles [48]:

- **Perfil de Transmisión Mejorada:** Definido en la versión 1.0. Mezcla la transmisión de datos audiovisuales con aplicaciones [22]. Requiere de un STB de gama baja con o sin canal de retorno.
- **Perfil de TV interactiva:** Definido en la versión 1.0. Permite el uso de un canal de retorno para la implementación de aplicaciones interactivas. Este perfil soporta la descarga de aplicaciones MHP también a través del canal de retorno.
- **Perfil de Acceso a Internet** [47]: Definido en la especificación 1.1. Requiere de receptores más avanzados en términos de memoria y procesamiento, permite una completa interactividad y acceso a contenido en línea y correo electrónico entre otros.
- **Perfil IPTV**: Definido en la versión 1.2. Es el perfil usado para soportar DVB-IPTV.

DVB también definió un lenguaje declarativo llamado DVB-HTML que nunca fue usado debido a que llego un poco tarde en la especificación y los fabricantes lo encontraban complejo y difícil de implementar [4].

2.2.4.6 GEM Globally Executable MHP. Se creó para permitir el uso de MHP en redes diferentes a DVB. Define un subconjunto de MHP que remueve los elementos

relacionados a DVB pero mantiene la API de desarrollo [40][47][48][52]. más que una especificación es un *framework* que permite a nuevas plataformas de distribución especificar estándares basados en GEM [2][47].

2.2.4.7 OCAP OpenCable Application Platform. Con MHP como corazón, OCAP es el *middleware* para los sistemas de cable en los Estados Unidos. Los servicios y aplicaciones deberían ejecutarse sobre cualquier sistema de cable compatible con la norma, independiente del equipo receptor [2] pero debido a la complejidad del mercado norteamericano esto no se cumple [47].

OCAP ha definido los dos siguientes perfiles:

- OCAP 1.0: Funcionalidad Básica. Basado en MHP 1.0.x
- OCAP 2.0: Adiciona a la primera versión el soporte a HTML.

Una característica única de OCAP es la definición de una aplicación monitora, esta tiene acceso a funcionalidades que no son accesibles por aplicaciones normales. Ayuda en la administración y control del ciclo de vida de las aplicaciones, se encarga del manejo de los recursos y de aspectos de seguridad [2][47].

2.2.4.8 ACAP Advanced Common Application Platform. Fue diseñado por la ATSC para unir OCAP y DASE [41], se basa principalmente en GEM. También tiene un modelo declarativo y uno procedimental llamados ACAP-X y ACAP-J, respectivamente. El primero basado en XHTML y el segundo en Java. Las aplicaciones definidas usando ACAP-J siguen el ciclo de vida definido por JavaTV [2][47].

Una característica importante de ACAP es que es soportado por todos los sistemas de DTV de Estados Unidos, cable, satélite y redes terrestres [47].

2.2.4.9 ARIB (Association of Radio Industries and Businesses). Es la entidad encargada de la DTV en Japón. Definió dos estándares, ARIB B.23 y B.24. La norma B.23 incorpora un sistema para la ejecución de aplicaciones procedimentales basado en MHP y GEM [48]. Y la especificación B.24 que ampara la definición de aplicaciones declarativas

mediante el uso de BML (*Broadcast Markup Language*). BML, está basado en XML y se deriva de un borrador de XHTML, el cual extiende y altera. También incluye un subconjunto de CSS 1 y 2, así como ECMAScript [28].

2.2.4.10 Ginga. Es el *middleware* especificado para la norma Brasilera, SBTVD [40]. Su arquitectura se divide en dos módulos: Base común y Servicio Específico. Este último permite el desarrollo de aplicaciones Java usando la definición Ginga-J, o usando el lenguaje declarativo NCL (*Nested Context Lenguage*) con Ginga-NCL [28][40][48][49]. Esta modificación a NCL permite usar el lenguaje Lua [14] para incrustar lógica procedimental dentro de documentos NCL [48].

El módulo base o común está compuesto por decodificadores y procedimientos para obtener y preparar los contenidos transportados en los flujos de transporte MPEG-2 y en el canal de retorno [48][28]. Los servicios de decodificación atienden a ambos tipos de aplicaciones [48].

La arquitectura Ginga puede ser usada por diferentes sistemas de transporte, como satélite, cable y terrestre [28][48].

2.2.4.11 *Middleware* de HbbTV. HbbTV es un caso especial ya que no cuenta con una definición formal de *middleware* como los demás estándares. Pero si define un ambiente de ejecución donde las aplicaciones son presentadas y ejecutadas. Este ambiente está formado por un administrador de aplicaciones, que se encarga del ciclo de vida de las mismas, y un Navegador (*Browser*) que es responsable de la presentación y ejecución de las aplicaciones interactivas [50]. Las aplicaciones se definen como una colección de documentos CE-HTML[15], JavaScript, CSS, XML y archivos multimedia, que constituyen un servicio interactivo o mejorado.

[14] Lua es un lenguaje de programación imperativo, estructurado y bastante ligero que fue diseñado como lenguaje de script con una semántica extensible. Lua (*programming language*). http://en.wikipedia.org/wiki/Lua_(programming_language). Consultado: 4 de mayo de 2012.
[15] CEA-2014-B (ANSI). http://www.ce.org/Standards/Standard-Listings/R7-Home-Network-Committee/CEA-2014-B-(ANSI).aspx. Consultado: 10 de mayo de 2012

2.2.5 Soluciones *Over-The-Top* (OTT)

Con el auge de la banda ancha el negocio de la distribución de video está migrando a Internet. OTT es un término general usado para catalogar servicios de video que son distribuidos a través de la Web. Se habla de estos como servicios que "pasan-por-encima" (del inglés "*over-the-top*") por que operan sólo sobre el servicio de acceso a internet y no requieren de ninguna relación de negocios o tecnológica con los operadores de red o proveedores de servicios [52]. YouTube [16], Netflix [17] y Hulu [18] son algunos de los proveedores de este tipo de servicios que puede llegar a una gran gama de dispositivos conectados a internet como computadoras, *smartphones*, decodificadores, consolas de videojuegos, televisores inteligentes como el Smart TV de Samsung o el Google TV de Sony que también está disponible a manera de consola.

2.2.5.1 Google TV. Google TV es una plataforma de televisión inteligente. Integra el sistema operativo Android[19] y el navegador Chrome para crear una capa de interactividad sobre la televisión y sitios de WebTV adicionando una interfaz de "10-Pies"[20] [53]. La funcionalidad del sistema puede ser ampliada mediante la descarga de aplicaciones desde la tienda de aplicaciones de Google llamada Google Play el cual usa el concepto de tienda de aplicaciones, también pueden desarrollarse aplicaciones para Android o diseñando sitios web en HTML, CSS y JavaScript. Esta plataforma permite usar el navegador para acceder a sitios web y ver televisión al mismo tiempo. Los productos de Google TV cuentan con un control remoto con un teclado QWERTY completo.

2.2.5.2 Samsung Smart TV. Es la tecnología usada por Samsung en sus televisores para brindar interactividad. También usa el concepto de tienda de aplicaciones, para el caso de Samsung llamado Samsung Apps. Las aplicaciones son páginas web especiales que se ejecutan en un navegador. Soporta los estándares HTML, DOM, CSS y JavaScript además de ser compatible con la tecnología flash de Adobe [54].

---

[16] YouTube. http://www.youtube.com. Consultado: 10 de mayo de 2012
[17] Netflix. http://www.netflix.com. Consultado: 10 de mayo de 2012
[18] Hulu. http://www.hulu.com. Consultado:10 de mayo de 2012 (No disponible para Latinoamérica)
[19] Android es un sistema operativo basado en Linux creado especialmente para dispositivos moviles, como smartphones y tablets.
[20] "10-pies" se refiere al echo de que los elementos de la interfaz de usuario deben ser lo suficientemente grandes para ser fácilmente usados desde una distancia de 10 pies o 3 metros de la pantalla. http://en.wikipedia.org/wiki/10-foot_user_interface. Consultado: 12 de mayo de 2012

2.2.6 Lenguajes Multimedia

2.2.6.1 Lenguaje de Integración de Multimedia Sincronizada (SMIL). Es un estándar del Consorcio *World Wide Web* (W3C[21]) que define un lenguaje declarativo que usa XML para describir presentaciones multimedia integrando diferentes contenidos de múltiples formatos y fuentes coordinando el tiempo de presentación, el diseño y las transiciones entre otras [55].

La estructura de un documento SMIL es parecida a la encontrada en los documentos HTML con dos secciones principales <*head*> y <*body*>. La etiqueta <*head*> contiene la información de diseño y metadatos y la sección <*body*> contiene la información referente a la sincronización de los medios. Los archivos multimedia son referenciados usando URLs permitiendo que estos estén almacenados en diferentes servidores.

SMIL ha estado tras el desarrollo del Servicio de Mensajería Multimedia (MMS), ampliamente usado en los teléfonos celulares, y también sirvió como capa de interactividad en el desaparecido formato de almacenamiento de datos y de video de alta definición HD-DVD[22].

2.2.6.2 Formato de Contenido Portátil (PCF). Aunque no es un *middleware* propiamente debido a que no es un lenguaje que se compile o interprete en ningún decodificador es de gran relevancia debido a que fue concebido como una capa intermedia. Es uno de los estándares menos conocidos de los especificados por la DVB en el área de la ITV. Fue publicado por la ETSI en 2006 y hasta ahora solo ha sido usado por la BBC en un sistema de producción y en algunos trabajos de investigación [12].

Fue diseñado para describir los servicios de televisión digital interactiva. Representa un formato independiente de plataformas, su uso se pensó para el intercambio de contenido interactivo *business-to-business* y por consiguiente es un medio para incrementar la

[21] World Wide Web Consortium. http://www.w3c.org. Consultado: 13 de Mayo de 2012.
[22] SMIL. http://en.wikipedia.org/wiki/Synchronized_Multimedia_Integration_Language. Consultado: Mayo 13 de 2012

interoperabilidad de las herramientas de creación, centrales de recepción (Head-ends) y redes de transmisión [24][25][26].

Esto es logrado al representar lo que deber ser la experiencia del usuario en lugar de como implementarla [24]. Esta descripción debe ser transformada al formato específico por un conversor que usa las características propias de la plataforma para crear la experiencia descrita.

Este formato encarna un modelo declarativo de alto nivel con sintaxis XML, tipos MIME (*Multipurpose Internet Mail Extensions*) y UML (*Unified Modelling Language*). Soporta descripciones independientes para diferentes aspectos del servicio interactivo, por ejemplo, contenido, presentación, comportamiento y navegación. Además no requiere que todos los aspectos del servicio sean descritos como una única unidad física, como un archivo, así las descripciones de los servicios pueden estar diseminados arbitrariamente en archivos localizados en Internet. Esta estructura permite que el autor más apropiado tome su rol en la creación y administración de la parte del servicio interactivo que le corresponde [24].

Un buen escenario para el uso del estándar PCF sería aquel en el cual una organización de radiodifusión crea su contenido en este formato y luego lo distribuye usando operadores de cable (traduciéndolo a OpenCable), satélites (traduciéndolo una plataforma de HTML + JavaScript) y televisión terrestre (traduciéndolo a MHP).

2.2.7 Estándares de Metadatos

Los metadatos son datos que describen otros datos. En general, un grupo de metadatos se refiere a un grupo de datos, llamado *recurso*.[23]

La multimedia es una de las áreas que más ha impulsado el desarrollo de estándares de metadatos, entre los más populares están ID3[24], *Dublin Core*[25], *Framework* para la

[23]Metadato. Wikipedia. http://es.wikipedia.org/wiki/Metadato. Consultado: 4 de Mayo de 2012.

descripción de recursos (RDF)[26] y dentro del campo de la televisión los más usados son MPEG-21, MPEG-7 y TV-*Anytime* [36] [42].

2.2.7.1 TV-Anytime. No es sólo un estándar sino un conjunto de especificaciones creado por el foro TV-*Anytime* [27] para su sistema, asegurando así un alto grado de interoperabilidad entre quienes los implementan. Fue diseñado para el entorno audiovisual, permite buscar, seleccionar, consumir y usar legalmente el contenido en sistemas de almacenamiento locales o remotos tanto para servicios transmitidos o en línea [36][45].

Su elemento básico se denomina Programa. Un programa representa un recurso audiovisual y tiene la descripción tanto semántica como de bajo nivel. Además propone un proceso llamado referenciación de contenido para poder lograr la separación entre la descripción de los programas y su localización. El elemento clave en este proceso son los Identificadores de Referencias de Contenido CRIDs (Content Reference IDentifiers), que permiten vincular los metadatos de un programa con su horario y lugares de transmisión.

Para la definición de metadatos adoptó el formato XML. Estos son los principales tipos de metadatos definidos en la especificación:

- **Descripción de contenidos**: Proporcionan características de los contenidos a los que están asociados, como por ejemplo: género, idioma, título, resumen, palabras claves, créditos, clasificación, fecha y país en el que se produjeron, y demás. Este tipo de datos (junto al CRID de cada programa) son los que deben especificar los usuarios a la hora de solicitar contenidos concretos.
- **Descripción de instancias**: Describe instancias concretas de un programa, indicando, entre otros campos, la hora y el canal en el que se emite, la duración, si se transmite en directo o en diferido, si es la primera o la última vez que está disponible, etc.

---

[24] *The Audience is informed*. http://www.id3.org/ Consultado: 4 de mayo de 2012.
[25] *The Dublin Core® Metadata Initiative*. http://dublincore.org/. Consultado: 4 de mayo de 2012.
[26] Resource Description Framework. http://www.w3.org/RDF/. Consultado: 4 de mayo de 2012.
[27] *TV-Anytime Forum*. http://www.tv-anytime.org/. Consultado: 4 de mayo de 2012.

- **Metadatos sobre el consumidor**: Permiten identificar las preferencias de cada usuario y su historial.
- **Metadatos de segmentación**: Cada flujo audiovisual puede descomponerse en varios segmentos. La información relativa a cada uno de ellos se proporciona mediante estos metadatos. Este mecanismo de segmentación permite que el usuario pueda acceder tanto al contenido completo como a cada uno de los segmentos identificados en el mismo.

2.2.7.2 MPEG-7. Es un estándar creado para representar contenido multimedia. Su principal función es la de describir información de bajo nivel como textura, color, movimiento y ubicación [36][42]. Está diseñado para que sea independiente del formato de video que describe, pero encaja mejor con la codificación MPEG-4 [42].

El estándar especifica cuatro tipos de componentes: descriptores, esquemas de descripción, lenguaje de definición de descripciones y esquemas de codificación [42].

Principales funcionalidades [43]:

- **Sistemas:** incluye el formato binario de codificación, la arquitectura de transmisión y manejo de la propiedad intelectual entre otros.
- **Lenguaje de Definición de Descripciones (DDL):** Es un lenguaje para la creación de nuevos esquemas de descripción. Es basado en XML.
- **Visual**: consiste en las estructuras básicas y descriptores que cubren características visuales fundamentales: color, textura, forma, movimiento, localización y reconocimiento facial.
- **Audio:** Es conjunto de descriptores de bajo nivel para el contenido de audio. Describen características espectrales, paramétricas y temporales.
- **Esquemas de Descripción Multimedia (MDS)**: Especifica la relación entre los descriptores (D) y los esquemas de descripción (DS) con el elemento multimedia.

- ***Software* de Referencia**: Implementación de las partes más importantes del estándar con estatus de normatividad para realizar simulaciones.
- **Conformidad**: Define procedimientos y guías para realizar pruebas de las implementaciones.
- **Extracción y uso de descripciones**: Es material informativo sobre la extracción y el uso de herramientas de descripción.
- **Perfiles**: Define pautas y perfiles estándar; los actuales son sobre el DDL, los esquemas de video, de audio, y de descripciones multimedia que se basan en las versiones del espacio de nombres descrito en la definición del esquema.
- **Definición del esquema**: Especifica el esquema usando DDL y recoge todos los demás esquemas de las diferentes partes del estándar así como las correcciones y adiciones.

2.2.7.3 MPEG-21. Es un marco de trabajo que hace un uso amplio de metadatos, brinda un ambiente controlado [42] que permite a los usuarios el intercambio, acceso, consumo y manipulación de multimedia de forma eficiente, transparente e interoperable [36][44].

Se fundamenta en dos conceptos: El concepto de objeto digital (*Digital Item*, DI) como la unidad fundamental de distribución y transacción [44] y el concepto de usuarios interactuando con DI.

Su especificación es flexible y permite funcionalidad de alto nivel e interoperabilidad al poder conectar a muchas de sus partes otros esquemas de descripción. Específica en su parte 2, la declaración digital de objetos (DDI) mediante términos y conceptos abstractos para crear un modelo de definición. Las partes de este modelo son [44]:

Identificador de objetos digitales (DII): Identifica a un único DI y sus partes.
Contenedor: Estructura que permite agrupar varios objetos de forma lógica

Objeto: Es un grupo de objetos y/o componentes ligados a un descriptor. Puede contener elecciones y anotaciones. Es la representación declarativa de un DI.

Componente: Es la unión de un recurso a todos sus descriptores. Pueden ser considerados como bloques de construcción de objetos. Cada componente contiene un recurso.

Descriptor: Tiene información del objeto que lo contiene. Esta puede ser texto plano o XML.

Recurso: Es una pieza identificable individualmente como un clip de audio o video, una imagen o un texto, incluso puede ser un elemento físico.

Otros: Anclas, Condiciones, Elecciones, Selecciones, Anotaciones, Afirmaciones, Fragmentos, Declaraciones y Predicados.

2.2.8 Seguimiento de Objetos

El seguimiento de objetos, más conocido del inglés *Object Tracking*, es el proceso en el que se estima la ubicación de un objeto en un determinado momento en una fuente de video. Este puede ser un proceso lento ya que el video contiene una gran cantidad de datos y las técnicas de reconocimiento de objetos son bastante complejas ya que implican operaciones matemáticas sobre matrices que están relacionadas con el propio tamaño del video [62][63]. Estas deben tener en cuenta la semejanzas del aspecto del objeto de interés y del resto de objetos en la escena, además de la variación del aspecto del propio objeto, ya que al cambiar de posición este puede cambiar de forma y color, entre otras propiedades, como por ejemplo [62]:

- **Cambios de posición del objeto**. Cuando la posición cambia, la proyección del objeto sobre el video cambia, por ejemplo al girar.
- **Iluminación ambiente**. El color y la intensidad de la luz hace que el objeto se vea afectado modificando su aspecto
- **Ruido**. Es cuando el valor de un píxel de una imagen no corresponde con la realidad, esto hace con la imagen se distorsione.
- **Oclusiones**. Puede ser que un objeto de interés no se observe bien cuando sea parcial o totalmente tapado por otros objetos en la escena.

Para el seguimiento de un objeto es necesario definirle una representación que es algo de interés para su análisis posterior, esta representación pueden ser por su puede ser su forma y apariencia. Entre las representaciones de forma se tienen: por puntos, formas geométricas primitivas, silueta del objeto y contorno, modelos articulados de forma y modelos esqueléticos. En las representaciones de apariencia están la densidad de los objetos, plantillas, modelos activos de aspecto y modelos de aspecto e múltiples vistas.

Las características del objeto también son fundamentales para su seguimiento, entre estas se tiene el color, márgenes, flujo óptico, textura.

2.2.8.1 Detección del Objeto. Para el seguimiento de un objeto es necesario detectar o reconocer el objeto ya sea en cada cuadro, o solo en la primera coincidencia de este. Comúnmente se utiliza la información de un solo cuadro para detectar el objeto, otros métodos usan la información calculada por una secuencia de imágenes con el fin de reducir detecciones falsas, para esto se usa "*Frame difference*" que consiste en elegir un "sistema de referencia" luego este se compara con cada cuadro en el vídeo y sólo se dibujan los pixeles que son diferentes de la trama de referencia por algún límite de tolerancia definido por el usuario [62].

Entre los algoritmos mas conocidos para realizar el seguimiento de imágenes se encuentran *Template Matching*, método basado en plantillas y la relación matemática de estas con la imagen fuente y SURF (*Speeded Up Robust Features*) método que usa las propiedades gráficas de la región a buscar.

2.2.9 Arquitectura

Actualmente se encuentran diversas definiciones de arquitectura de software, haciendo de este un término muy discutido. No existe una definición aceptada universalmente [59]. En esta sección se dará una definición de arquitectura de software intentando concluir de las múltiples definiciones que se pueden encontrar pero teniendo como referencia principal la dada por Bass [57].

"La arquitectura de un programa o sistema de computación es la estructura o estructuras del sistema, que consta de elementos de software, sus propiedades externas visibles y la relación entre ellos".

Para entender adecuadamente esta definición hay que entender cada una de sus partes:

- **Elementos**: La arquitectura define los elementos de software y hardware y sus interfaces y la información de cómo se relacionan entre sí estos elementos que conforman el sistema [57][59].
- P**ropiedades visibles externamente:** Son las propiedades de un elemento que los demás elementos pueden asumir, entre estos están los servicios prestados, el manejo de errores, el uso de los recursos compartidos, las características del rendimiento entre otros [57].

En casi todos los sistemas, los elementos interactúan entre sí por medio de interfaces con elementos públicos y privados, la arquitectura se refiere a la parte pública, los detalles privados tienen que ver exclusivamente con la implementación interna los cuales no hacen parte de la arquitectura.

2.2.9.1 *Stakeholders*. Cualquier persona, grupo, organización, o sistema externo que pueda intervenir en los requerimientos o restricciones del sistema o verse afectados por estos son llamados *stakeholders*. Los *stakeholders* están definidos por sus interacciones con los sistemas de software. Cada uno requerirá documentación en la que pueda entender las estructuras relevantes que conciernen a sus interacciones. Los *stakeholders* también se interesan en la distribución de la funcionalidad dentro del sistema, que puede ser descrita en términos de estructuras arquitectónicas. Las cualidades de las propiedades de estas estructuras y sus relaciones pueden ser asignadas directamente a los requerimientos no funcionales del sistema [56][57][60].

2.2.9.2 Atributos de calidad. Los atributos de calidad se enfocan en los requisitos no funcionales del sistema, siendo estos los aspectos que no afectan directamente a la funcionalidad requerida. Los atributos de calidad definen las cualidades y características que el sistema debe soportar.

La arquitectura posibilita o restringe los atributos de calidad, aunque no todos los atributos son responsabilidad de la arquitectura. La Estos atributos de calidad son los mencionados por Bass[57]

- **Disponibilidad:** Es la capacidad de un sistema o componente para mantener la prestación de sus servicios. Esta relacionado con las fallas del sistema y las consecuencias asociadas. Por ejemplo un sistema transaccional debe estar disponible 7x24, y en caso de falla debe recuperarse de esta en un intervalo X de tiempo.
- **Modificabilidad:** Tiene que ver con el costo del cambio en el sistema. Los aspectos que se tienen en cuenta son, que hay que cambiar, cuando y quien. Un caso de modificabilidad puede ser que se quiera tener una nueva funcionalidad en el sistema y que el análisis, diseño, implementación, pruebas y despliegue no tome más de x días contados desde su aprobación.
- **Rendimiento:** Está relacionado con el desempeño, es decir la velocidad de respuesta y precisión u otras restricciones temporales en las que el sistema lleva a cabo una funcionalidad específica. Un ejemplo de rendimiento es el tiempo establecido en el que un sistema en condiciones normales procesaría una cantidad X de transacciones por unidad de tiempo.
- **Seguridad:** Medida sobre la capacidad del sistema para resistir usos no autorizados mientras sigue proveyendo sus servicios a los usuarios autorizados, por ejemplo si en un administrador de seguridad elimina o adiciona permisos sobre un componente del sistema, este debe reflejar inmediatamente las restricciones.
- **Verificabilidad:** Es la facilidad que tiene un sistema para ser probado.
- **Usabilidad:** es el grado de facilidad con que un usuario hace uso de un sistema. En este atributo se contemplan casos como el de proveer la posibilidad de cancelar operaciones o el de deshacer una acción ejecutada.

2.2.9.3 Escenario de Calidad. La definición de estos atributos siempre ha sido tema de análisis y discusión. Una forma de expresar estos atributos de una forma concreta y medible es usar la técnica de escenarios de calidad propuesta por Felix Bachmann y Mark

Klein [57] en el que un atributo se representa mediante un escenario, como se ve en la siguiente figura.

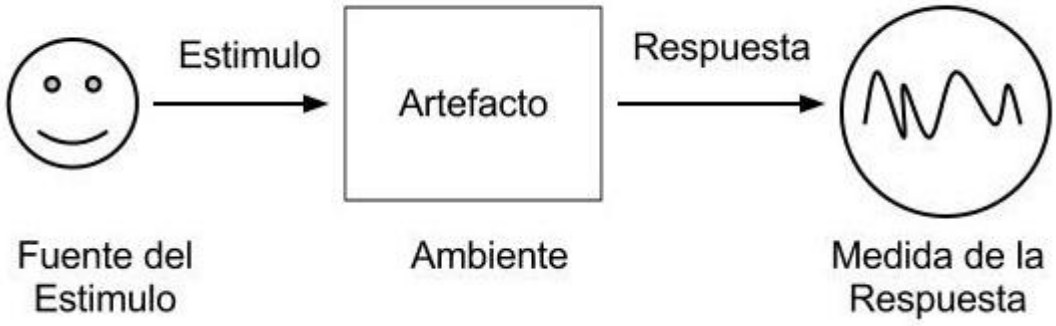

Figura 6 Partes de un escenario de atributos de calidad (Adaptada de la figura 4.1 *Quality attribute parts* de [57]).

- Fuente de estímulo: Es una entidad, ser humano, sistema o cualquier otro actor que generó el estímulo.
- Estímulo. Es una condición que debe ser considerada cuando llega a un sistema.
- Ambiente: Condiciones en la cual se encuentra el sistema en el momento que se recibe el estímulo afectándolo.
- Artefacto: Componentes del sistema que son afectados.
- Respuesta: es la actividad que debe realizar el sistema después de recibir el estímulo.
- Medida de la respuesta: Es un tipo de medida con el cual debe cumplir la respuesta para que el requerimiento pueda ser probado.

2.2.10 Cumpliendo los Atributos de Calidad

Luego de exponer los atributos que un sistema debe satisfacer lo que continua es tomar las decisiones de diseño que ayudarán a cumplir dichos atributos.

2.2.10.1 Táctica. Una táctica es una decisión de diseño que influye en el control de la respuesta de un atributo de calidad. Por ejemplo, una táctica para lograr disponibilidad

puede ser optar la redundancia de componentes de tal forma que se puedan intercambiar cuando se presente una falla [57].

Es importante tener en cuenta que las tácticas que se usen para darle solución a uno de los atributos de calidad impactan en menor o mayor grado otros atributos.

2.2.10.2 Estilo. Un estilo arquitectónico es una especialización de tipos de elementos y tipos de relaciones junto con un grupo de restricciones sobre cómo esos tipos de elementos y relaciones pueden ser usados. Los estilos no especifican información del contexto o del problema, estos sirven para categorizar arquitecturas y definir sus características comunes, encapsulan decisiones importantes sobre los elementos arquitectónicos y enfatizan las restricciones importantes de los elementos y sus relaciones [56][57][59].

Un estilo arquitectónico está determinado por:

- Un conjunto de tipos de componentes que cumplan con alguna especificación.
- La distribución lógica o física de los componentes indicando sus interacciones durante el tiempo de ejecución.
- Un conjunto de restricciones.
- Y un conjunto de conectores que median la comunicación y coordinación entre los componentes.

La siguiente figura resume algunos estilos arquitectónicos.

Figura 7 Ejemplos de estilos (Adaptada de la figura 4.1 *A partial representation of the space of C&C styles* de [56])

2.2.10.3 Patrones. Un patrón es similar a un estilo en cuanto a sus componentes, la gran diferencia es que los patrones existen para un contexto en particular para el problema a resolver [56][57][59]. Se puede decir que un estilo es una solución generalizada y un patrón es una especialización de una solución.

Ambos, estilos y patrones, proveen vocabularios que facilitan la explicación y la comunicación de la arquitectura.

2.2.10.4 Diseño Basado En Atributos (ADD). ADD [57] es un método de diseño arquitectónico de la SEI en el que la forma de definir una arquitectura de software basa su proceso en los atributos de calidad que se deben satisfacer. Este método es recursivo, toma como entrada una lista de elementos llamados *drivers* arquitectónicos y da como resultado la base de la arquitectura de una manera conceptual.

Las etapas que conforman este método son [56][57]:

1 Seleccionar el sistema o módulo a diseñar.
2 Identificar los *drivers* arquitectónicos, es decir el conjunto de escenarios de calidad, requerimientos funcionales y restricciones, que le darán forma a la arquitectura o módulo.
3 Escoger el estilo o patrón de acuerdo a las tácticas que se puedan usar para satisfacer los *drivers.*
4 Identificar los submódulos que implementan las tácticas y asignar funcionalidades.
5 Definir las interfaces de los submódulos.
6 Verificar y refinar los requerimientos y convertirlos en restricciones para los submódulos.
7 Repetir los pasos para cada elemento que requiera mayor análisis.

2.2.11 Documentación

El diseño arquitectónico de software desde una perspectiva técnica según [57] es importante porque:

- Estandariza la comunicación con las personas interesadas o *stakeholders*.
- Manifiesta el primer conjunto de decisiones de diseño importantes con respecto a la organización de un sistema software.
- Y sirve como modelo para transferir y reusar.

Por estos motivos la documentación de una arquitectura es fundamental a la par de la misma toma de decisiones de diseño. Una arquitectura debe ir acompañada de la documentación apropiada de tal forma que los *stakeholders* puedan conocer y usar lo que les interesa de la arquitectura.

2.2.11.1 Vista. Debido a la alta complejidad de los sistemas y de los diferentes *stakeholders*, el diseño se debe dividir en diferentes perspectivas o dimensiones, enfocando la atención en un momento dado a un pequeño número de estructuras del sistema. La representación de estas estructuras es a lo que llamamos vista [56][57][59][60].

Existen varios modelos que ayudan a definir las diferentes dimensiones organizacionales, de negocios y tecnológicas de una arquitectura. Dos de estos son el modelo “4+1” vistas de Kruchten [61] y el modelo “Vistas y más allá” (V&B, Views and Beyond) propuesto por el Instituto de Ingeniería de Software, SEI [56]. El modelo “4+1” vistas propone cinco vistas: vista lógica, vista de procesos, vista de desarrollo, vista física y vista de escenarios. Y V&B por su parte define tres “estilos de vistas”: estilos de módulos, estilos de componentes y conectores y estilos de asignación. Ambos modelos se centran en describir las estructuras de software y no en definir las notaciones de estas estructuras.

2.2.12 Evaluación De Una Arquitectura

Es la arquitectura la que determina el éxito o fracaso de un sistema, minimizando los riesgos que puedan presentarse en su implementación. Debido a su gran importancia se hace necesario que esta sea evaluada para garantizar que la arquitectura seleccionada sea apropiada, verificando que los requerimientos no funcionales estén presentes y determinando en qué grado se satisfacen, y así cumplir con las necesidades especificadas de los *stakeholders*.

Según la literatura [57][65][66] la evaluación de la arquitectura puede realizarse en cualquier momento, pero se distingue principalmente entre evaluación temprana y evaluación tardía.

- **Temprana**: No es necesario que la arquitectura esté completamente especificada para efectuar la evaluación y abarca desde las fases tempranas de diseño y hasta las de desarrollo.
- **Tarde**: Esta evaluación se realiza cuando existe por lo menos un diseño detallado en el que ya se pueden recoger métricas concretas.

Para realizar estas evaluaciones existen diversas técnicas que se pueden clasificar en cualitativas y cuantitativas. Las técnicas cualitativas son utilizadas durante la evaluación temprana, mientras que las técnicas de cuantitativas, se usan en la evaluación tardía. Las técnicas más principales son [66]:

Cualitativas:

- Cuestionarios
- Listas de verificación
- Escenarios

Cuantitativas:

- Métricas arquitectónicas
- Prototipos [67][68][69] y/o simulaciones
- Experimentos

2.2.12.1 Cuestionarios y Listas de verificación. Los métodos basados cuestionarios y listas de verificación usan preguntas para averiguar si la arquitectura cumple con los requisitos funcionales y no funcionales. Son analizados estadísticamente a partir de las respuestas dadas por los *stakeholders* consultados [57].

2.2.12.2 Escenarios. Los métodos basados en escenarios [66] evalúan la arquitectura desde un nivel de abstracción más alto y todos siguen a grandes rasgos los mismo pasos:

1 Describir la arquitectura a evaluar.
2 Desarrollar los escenarios a partir de los atributos de calidad.
3 Priorizar los escenarios.
4 Evaluar la arquitectura desde el punto de vista de los escenarios de mayor prioridad.
5 Exposición de resultados

En este grupo algunos de los métodos más usados son:

- **ATAM (Architecture Trade-off Analysis Method):** este método se encarga de definir los estilos arquitectónicos que se usarán para satisfacer los escenarios de calidad y como esta definición afecta a otros atributos mediante la identificación de riesgos [57].
- **ADR (Active Design Review):** Evalúa diseños detallados como los componentes o módulos. Las preguntas giran en torno a la calidad y completitud de la documentación y la suficiencia, el ajuste y la conveniencia de los servicios que provee el diseño propuesto [56].
- **ARID (Active Reviews for Intermediate Design):** es un híbrido entre ATAM y ADR, es conveniente para realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo. Consiste en ensamblar el diseño de los *stakeholders* con los escenarios de usos más importantes.

También existen técnicas para evaluar atributos de calidad específicos basadas en escenarios, como es el caso de SALUTA, que se concentra en evaluar la usabilidad [70].

2.2.12.3 Métricas Arquitectónicas. Estas métricas permiten medir el comportamiento de atributos de la arquitectura, en especial del código, y sobre estas mediciones inferir acerca de la calidad de la arquitectura. Algunas de estas medidas son: Complejidad, grado de herencia, Acoplamiento, que mide la dependencia entre componentes;

Cohesión, que mide la dependencia entre partes de un mismo componente; entre otros [66].

2.2.12.4 Prototipos. Según Bardram [69] un prototipo arquitectónico es una técnica que:

> "... consta de en un conjunto de ejecutables creados para investigar las cualidades arquitectónicas relacionadas con los intereses planteados por los *stakeholders* de un sistema en desarrollo."

Y los clasifica en:

- Exploratorios: Usados principalmente para aclarar los requerimientos de un sistema y para discutir posibles arquitecturas.
- Experimentales: Usados para evaluar la idoneidad de la arquitectura propuesta, o para obtener detalles antes de realizar grandes inversiones en implementación.

Los prototipos experimentales son los que se usan normalmente para evaluación. En estos se construyen unos componentes ejecutables que implementan la funcionalidad necesaria para ejecutar los escenarios propuestos sobre los atributos de calidad. Los resultados que se obtengan se usan para las siguientes etapas de desarrollo o para corregir la arquitectura de ser necesario [66].

CAPÍTULO 3 DISEÑO DE LA ARQUITECTURA

3.1 INTRODUCCIÓN

El propósito de este numeral es dar a conocer el diseño de la arquitectura propuesta y del porqué de las decisiones tomadas para la solución del problema planteado describiendo el sistema y sus componentes. Cabe anotar que el diseño propuesto es el resultado de varias iteraciones y del cual se desprende el desarrollo de un prototipo funcional que será explicado en el siguiente capítulo.

Para el diseño de la arquitectura se siguió un esquema fundamentado en el método de Diseño Basado en Atributos [57] tal como se explicó en el marco teórico. Los resultados de aplicar este método son expuestos en una serie de vistas arquitectónicas y de interfaces que sirven para entender las diferentes dimensiones del diseño. Para documentar estos elementos se siguió una notación semi-formal basada en el trabajo propuesto por Clements y otros en [56] para el SEI.

3.2 DESCRIPCIÓN DEL DOCUMENTO

3.2.1 Audiencia

Esta documentación se creó para los participantes del proyecto, entre quienes se encuentran, el director del proyecto, el investigador principal y los auxiliares de la universidad EAFIT en el área de DTV. Además para empresas de telecomunicaciones, empresas de producción de video y empresas desarrolladoras de aplicaciones para ITV que necesitan portar programas relacionados con sus videos a diferentes plataformas. También se tuvieron en cuenta los posibles evaluadores de este proyecto.

3.2.2 Organización

Este capítulo consta de cinco secciones distribuidas así:

- Sección 1: Resume la descripción del documento mismo.
- Sección 2: Describe la visión general del proyecto incluyendo el contexto del sistema, y los participantes principales.
- Sección 3: Expone los *drivers* arquitectónicos (restricciones, casos de uso más representativos y los atributos de calidad)
- Sección 4: Lista los estilos y tácticas arquitectónicas usadas con el fin de cumplir con los *drivers* arquitectónicos.
- Sección 5: Describe el diseño arquitectónico del proyecto a través de vistas.

3.2.3 Documentación Del Proyecto Relacionada

Para una mejor comprensión de la documentación presentada se sugiere ver: ANEXO A. Documento de Requisitos y Casos de Uso

3.3 VISIÓN GENERAL DEL PROYECTO

3.3.1 Contexto Del Negocio

El contexto del negocio, tal como se expuso en las secciones iniciales de este documento, está definido por la interacción de los diferentes participantes de la cadena de producción de televisión, las plataformas de transmisión y las aplicaciones interactivas.

El propósito principal de esta arquitectura es ofrecer a las diferentes entidades involucradas en el proceso de producción de televisión interactiva herramientas que les permitan crear aplicaciones a partir de la selección de objetos o elementos presentes en un video, permitiéndoles llegar a múltiples plataformas de ITV con mejores tiempos de desarrollo y con aplicaciones asociadas a su contenido.

La realización de este proyecto se da gracias a la infraestructura proporcionada por el laboratorio de televisión digital encargado de la investigación, el desarrollo y la innovación de ésta área en la Universidad EAFIT.

3.3.2 Contexto Del Sistema

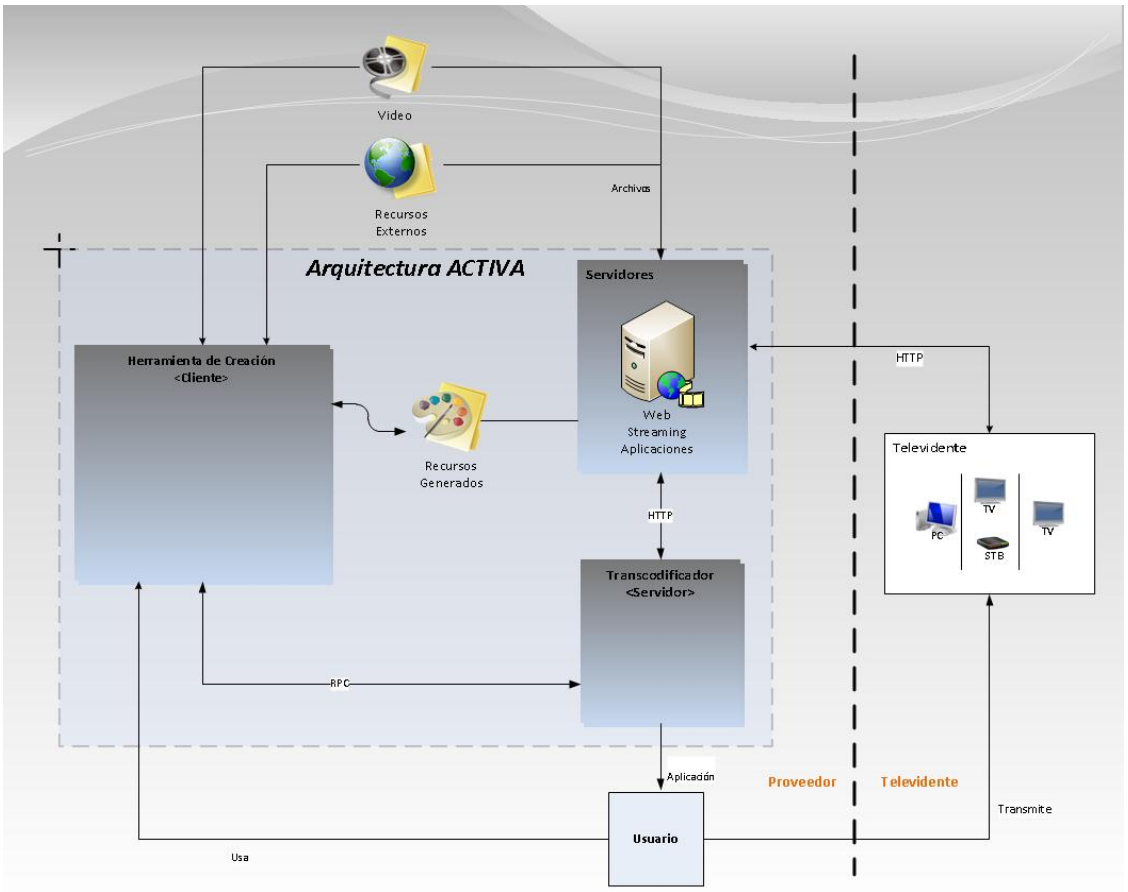

Figura 8 Arquitectura General

A partir del análisis de requisitos y casos de uso se diferencian dos componentes principales de la arquitectura, la Herramienta de Creación y el Transcodificador, tal y como se presenta en la Figura 8. Estos dos componentes definen las dos grandes funciones de la arquitectura, que son el diseño y la transformación de las aplicaciones respectivamente.

El estilo básico de la arquitectura es Cliente-Servidor mediante el uso de la Invocación a Procedimientos Remotos o RPC (del inglés, Remote Procedure Call). Siendo la herramienta de creación quien inicia el proceso solicitando al Transcodificador servicios de conversión.

El proceso de desarrollo de una aplicación para ITV comienza desde la concepción del video. Los productores deben decidir qué elementos del video (personajes, productos, entre otros) harán parte de la aplicación, como y cuando se van a ver y qué información se distribuirá con ellos. Por ejemplo: si se decide que un personaje de un video hará parte de la aplicación, los encargados de esta deberán suministrar en que tiempo del video aparece, con que color se verá enmarcado y los datos biográficos que se desplegaran.

La siguiente parte del proceso es el diseño de la aplicación. Durante el diseño los usuarios deben suministrar a la Herramienta de Creación el video y a medida que lo reproducen lo deben enriquecer seleccionando los elementos que definieron en la concepción de la aplicación y adicionando los datos y recursos suministrados. Para esto usan herramientas que les permiten dibujar y seleccionar objetos sobre el video en los momentos en los que los elementos se presentan.

Al terminar el diseño de la aplicación el proceso continúa con su transformación. El usuario solicita al Transcodificador que convierta el diseño a una o varias plataformas de televisión dependiendo de a cuales va a transmitir.

En la última parte del proceso, una vez la aplicación está terminada los operadores son los encargadas de distribuirlas a sus usuarios finales, es decir a sus televidentes, usando las plataformas de televisión que tenga a su disposición y para las cuales haya creado sus aplicaciones.

Como resultado el televidente podrá acceder a la información adicionada o ejecutar acciones asignadas a diferentes objetos del video al momento de su reproducción o transmisión haciendo uso de las aplicaciones de ITV generadas.

Debido a que uno de los posibles escenarios de uso de la arquitectura se puede dar de forma distribuida debido a la naturaleza de los participantes, el Transcodificador se definió como un componente prestador de servicios, dándole la posibilidad de atender a diferentes clientes, para este caso, diferentes Herramientas de Creación instaladas en diferentes empresas. Así, por ejemplo el diseño de las aplicaciones puede estar en manos de quienes crean el video y la transformación en manos de las empresas de telecomunicaciones que son las encargadas de distribuirlo.

3.3.3 Participantes Claves

- **Director de Proyecto**. Es el asesor de esta tesis. También es el encargado de dar los lineamientos generales del proyecto y de la asignación de recursos. Así mismo de la revisión y aprobación del diseño y el prototipo.
- **Arquitecto de Software**. Ingeniero de Sistemas y principal investigador del proyecto. Encargado del desarrollo de la arquitectura en todos sus niveles, diseño detallado, construcción y pruebas del prototipo. Adicionalmente de distribuir los módulos a desarrollar y a documentar.
- **Desarrolladores del proyecto**. Grupo de personas pertenecientes al laboratorio de televisión digital responsables de implementar los diferentes módulos de software. En este caso el Arquitecto también hace parte de este grupo.
- **Documentadores**. Grupo formado por un estudiante de ingeniería de sistemas y el arquitecto, encargado de mantener la documentación relacionada a la arquitectura.
- **Otros *Stakeholders* Involucrados**. Empresas de Telecomunicaciones, de producción de video y empresas de desarrollo: encargados de administrar contenidos y de crear aplicaciones para ITV que necesitan desarrollar aplicaciones para diferentes plataformas.

3.4 *DRIVERS* ARQUITECTÓNICOS

3.4.1 Restricciones De Negocio

Leyenda

RN = Restricción de Negocio

Tabla 1 Restricciones de Negocio

RN1	La arquitectura sólo es responsable del desarrollo de aplicaciones, en ningún momento realizará las funciones de transmisión.
RN2	La herramienta de creación debe ejecutarse en múltiples plataformas.

3.4.2 Restricciones Técnicas

Leyenda

RT = Restricción Técnica

Tabla 2 Restricciones Técnicas

RT1	Los *middlewares* a usar deberán soportar realizar gráficos programáticamente.
RT2	Los *middlewares* a usar deberán tener la capacidad de entregar el tiempo de reproducción de un video en milisegundos.
RT3	El ambiente de ejecución de las aplicaciones finales deben tener la capacidad de refrescar un mínimo de 20 cuadros por segundo.

3.4.3 Requisitos Funcionales Claves

El sistema para la creación de aplicaciones de ITV deberá soportar las siguientes funcionalidades presentadas en la Figura 9.

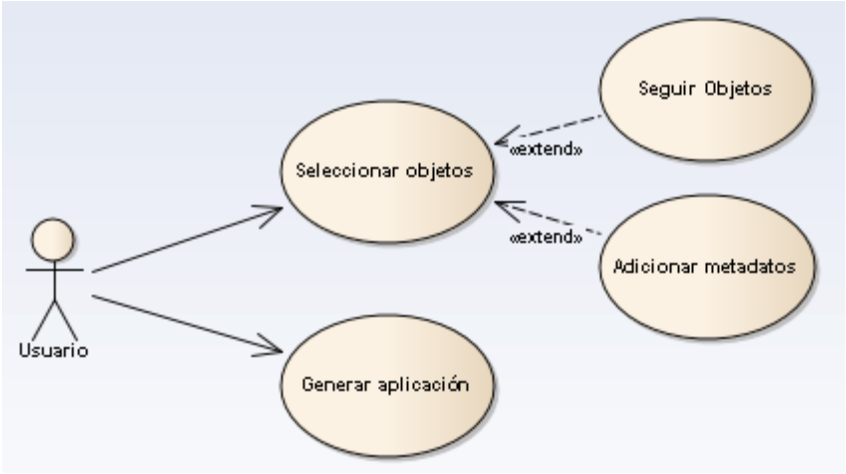

Figura 9. Casos de uso más representativos del proyecto de investigación

Leyenda

CU = Caso de Uso

Tabla 3 Casos de Uso representativos

ID	Nombre	Descripción	Atributos de Calidad
CU3	Seleccionar objetos	El sistema debe permitir dibujar sobre el video que esta siendo reproducido para seleccionar objetos.	Rendimiento Usabilidad
CU3.1	Adicionar Metadatos	El sistema debe permitir enriquecer los objetos seleccionados mediante la adición de metadatos.	Interoperabilidad
CU3.2	Seguir objetos	El sistema debe permitir hacer seguimiento automático de los objetos seleccionados.	Rendimiento
CU4	Generar Aplicación	El sistema debe permitir exportar las aplicaciones diseñadas a diferentes plataformas de televisión.	Portabilidad

3.4.4 Atributos De Calidad

A continuación se presentan los atributos de calidad más representativos en términos de los escenarios de calidad [57] para la arquitectura:

3.4.4.1 Escenarios de Usabilidad

Fuente del Estímulo: Usuario desarrollador de las aplicaciones para ITV.

Estímulo: Los usuarios desean seleccionar y enriquecer objetos de un video en un instante de la reproducción dado.

Artefacto: Herramienta de creación.

Ambiente: Tiempo de ejecución durante el diseño de una aplicación.

Respuesta: Interfaz con barras de herramientas para dibujo como trazar líneas, seleccionar colores, seleccionar objetos dibujados, entre otras. (Similar a una herramienta de edición de imágenes.). Adicionalmente se debe dar la posibilidad de buscar y reusar información previamente suministrada.

Medida:

- El grado de satisfacción de los usuarios debe superar el 80%.
- El tiempo de selección de un objeto y la selección de un algoritmo de seguimiento no debe superar 1 minuto. Al tiempo de enriquecimiento no se le dará una medida porque esta depende de la cantidad de datos que se agreguen.
- El usuario no debe equivocarse en más del 10% de las tareas.

3.4.4.2 Escenarios de Rendimiento

Fuente del Estímulo: Usuario desarrollador de las aplicaciones para ITV.

Estímulo: Solicitud para ejecutar el seguimiento de objetos.

Artefacto: Herramienta de creación.

Ambiente: Tiempo de ejecución durante el diseño de una aplicación.

Respuesta: El seguimiento de objetos es ejecutado.

Medida: El tiempo de procesamiento por cuadro de video no puede ser superior a 500 milisegundos.

3.4.4.3 Escenarios de Extensibilidad

Fuente del Estímulo: Desarrollador de la arquitectura

Estímulo: Adición de nueva forma de seleccionar objetos o de un nuevo algoritmo de seguimiento de objetos.

Artefacto: Herramienta de creación.

Ambiente: Tiempo de desarrollo.

Respuesta: Se adiciona la nueva funcionalidad sin modificar el código existente.

Medida: El tiempo de adicionar la nueva funcionalidad no debe tomar más de un día y no debe incluir el tiempo de desarrollo de la funcionalidad.

3.4.4.4 Escenarios de Extensibilidad. La necesidad planteada al inicio de la investigación sobre la portabilidad de las aplicaciones de ITV se transforma en un escenario de extensibilidad para la arquitectura. Esta es la forma de darle solución al despliegue de una aplicación de ITV en múltiples plataformas.

Fuente del Estímulo: Desarrollador de la arquitectura

Estímulo: Adición de nueva plataforma de televisión de interactiva.

Artefacto: Transcodificador.

Ambiente: Tiempo de desarrollo.

Respuesta: Se adiciona la nueva plataforma de televisión.

Medida: El tiempo de adicionar la nueva plataforma no debe tomar más de una semana y no debe incluir el tiempo de desarrollo de la nueva plataforma.

3.5 TÁCTICAS Y ESTILOS ARQUITECTÓNICOS

3.5.1 Tácticas Arquitectónicas

3.5.1.1 Disponibilidad

- **Detección de fallas - Excepciones**: En caso de presentarse una situación excepcional durante el funcionamiento de los diferentes componentes de la

arquitectura el componente afectado registrará en archivos de *logs* dicha excepción.

- **Recuperación de fallas - Transacciones:** Esta táctica será usada para prevenir que los metadatos creados usando la herramienta de creación se pierdan si ocurre alguna falla en este componente.

3.5.1.2 Modificabilidad

- **Localización de modificaciones - Mantener la coherencia semántica:** Se hará una separación de funciones y se deberá asegurar que cada módulo de software se mantenga acorde a esta definición. Esta separación se especifica en la vista de módulos en el numeral 3.6.1.
- **Localización de modificaciones - Anticipación de cambios esperados:** Se supone que las solicitudes de los cambios se harán en las funciones de selección de objetos, seguimiento de objetos y plataformas de televisión. Aparte de la complejidad que pueda tener la implementación de cada uno, su integración a la arquitectura debe ser posible adicionando puntos de extensión donde solamente se registren funciones nuevas sin necesidad de modificar el código existente. Ver más detalles en la vista de módulos en el numeral 3.6.1.
- **Prevenir efectos de propagación - Separación de interfaz e implementación:** Los diferentes servicios de acceso a datos y lógica de negocio serán implementados a partir de la definición de una interfaz con el fin de mitigar el efecto que pueden tener los cambios o modificaciones en dichos servicios en otras partes de la arquitectura.
- **Aplazar tiempo de vinculación - Registro dinámico de componentes:** Al aplicar la táctica anterior de separación de interfaz e implementación se obtiene el beneficio de permitir múltiples implementaciones, por lo que en tiempo de ejecución habrá un módulo encargado de registrar para cada interfaz cuales son las implementaciones que se pueden usar y entregar la adecuada en el momento de ser solicitada. Ver más detalles en la justificación sobre PicoContainer en la vista de componentes y conectores, numeral 3.6.4.
- **Aplazar tiempo de vinculación - Archivos de Configuración:** La herramienta de creación contará con un archivo de configuración en que se registrará la ruta de acceso al Transcodificador y otros componentes externos (servidores web) a usar

de tal forma que estos puedan ser cambiados en tiempo de ejecución. De igual manera el Transcodificador también dependerá de un archivo de configuración para anotar la rutas de: acceso a los recursos que usa para la conversión de aplicaciones y al servidor donde alojará los resultados.

3.5.1.3 Rendimiento

- **Demanda de Recursos - Incrementar eficiencia computacional:** Al tratarse con un recurso que requiere tanta capacidad de cómputo como lo es el video, las funciones de reproducción y de seguimiento de objetos se delegaran a librerías nativas[28] especializadas en éstas funciones que puedan explotar propiedades específicas de la plataforma computacional, en especial de la arquitectura de los procesadores de las máquinas donde se ejecuten las herramientas de creación. Ver más detalles en las justificaciones de las vistas de módulos, numeral 3.6.1.4 y vistas de componentes y conectores numeral 3.6.4.

3.5.1.4 Usabilidad

- **Táctica en tiempo de ejecución - Modelo de la tarea:** La interfaz de usuario deberá mantener un modelo de las acciones realizadas por el usuario, en este caso como las tareas tienen relación con dibujar y seleccionar objetos las herramientas visuales como iconos y barras de herramientas deben de estar acorde con éstas tareas, por ejemplo seguir el diseño de otras aplicaciones que sirvan para este propósito, como editores de imágenes y de video, y así brindar un ambiente familiar los usuarios de este tipo de herramientas en el que visualice fácilmente las acciones para dibujar y para pintar. La otra propiedad sobre el modelo de la tarea que en este caso se debe tener en cuenta es la reducción de dibujos que el usuario deba realizar permitiendo repetir selecciones de un cuadro a otro sin la intervención del usuario y en algunos casos hacerlo sobre cuadros sucesivos automáticamente mediante las funciones de seguimiento de objetos.
- **Táctica en tiempo de diseño - Separación de la interfaz de usuario:** Esta táctica es a fin con la táctica sobre el mantenimiento de la coherencia semántica.

[28] Una librería nativa es una librería escrita para un *hardware* y un sistema operativo específico, usualmente escritas en C, C++ o lenguaje ensamblador.

La interfaz de usuario de la herramienta de creación se deberá mantener separada de la lógica de negocio así se facilitan futuros cambios reduciendo el impacto sobre los otros componentes. Para más detalles ver la justificación de la Herramienta de Creación en la vista de módulos, numeral 3.6.1.

3.5.1.5 Interoperabilidad

- **Utilización de estándares:** Para garantizar en gran medida la interoperabilidad del Transcodificador y sus posibles clientes se deberá preferir usar un estándar ya conocido en el ámbito de la televisión tanto para el transporte de los metadatos como para la especificación de aplicaciones interactivas.

3.5.2 Estilos Arquitectónicos

- **Invocación Remota de Procedimientos (RPC)**: Se seguirá éste estilo para habilitar los servicios del Transcodificador a cualquier cliente que quiera solicitar la generación de aplicaciones de televisión interactiva más allá de la herramienta de creación de la arquitectura, siempre y cuando estos clientes implementen las interfaces y los estándares que se definirán para transformar las aplicaciones. Ir a la justificación de la vista de componentes y conectores, numeral 3.6.4.
- **Datos compartidos:** El sistema implementará este estilo para compartir datos entre la herramienta de creación y el Transcodificador. En este implementación la herramienta de creación y el Transcodificador subirán la aplicación diseñada y la aplicación generada respectivamente a un espacio compartido en un servidor web.
- **Máquina virtual:** La herramienta de creación de aplicaciones se construirá para que funcione sobre la máquina virtual de Java. Aunque este estilo impacta directamente en el desempeño de la arquitecta se escogió porque permite darle solución a la restricción de negocio planteada sobre la portabilidad de la herramienta de creación.
- **Arquitectura en capas**: En la herramienta de creación se adoptará este estilo. Desde ahora se prevén las capas a) de presentación: dedicada principalmente al reproductor de video, a la zona para seleccionar objetos y a los formularios para el enriquecimiento de los objetos. b) De servicios o lógica de negocios: además de prestar servicios sobre el modelo de datos del negocio y del ciclo de vida de los

proyectos de desarrollo tendrá una división adicional que corresponderá a la capa de librerías nativas usadas para ciertos procesamientos de video. c) De acceso a datos y unas capas transversales que tienen que ver con el d) modelo de objetos y con e) utilidades generales como logs e internacionalización entre otros. Tal y como se aprecia en la Figura 10 Diagrama de Uso de Módulos. Para ver más detalles ir a la justificación de la vista de módulos.

3.5.3 Patrones Arquitectónicos

- **Inyección de dependencias**: Se usará este patrón para liberar la arquitectura de la complejidad de esta implementación. Las capas de acceso a datos y de servicios registrarán sus implementaciones del modelo del negocio para que los demás servicios y o capas superiores hagan uso de ellos indirectamente.
- **Métodos modelo:** En los algoritmos de seguimiento y en la generación de la aplicación final se prevé usar algoritmos genéricos con algunos pasos que deberán rellenar las diferentes implementaciones que se creen.

3.6 VISTAS DE LA ARQUITECTURA E INTERFACES

3.6.1 Vista de Usos, Capas y Aspectos

A continuación se presentan los principales módulos que conforman la arquitectura. Corresponde a una separación lógica que muestra las partes de cada subsistema y sus relaciones de uso. También se presentan las diferentes entidades externas, en este caso las librerías a utilizar.

3.6.1.1 Presentación Primaria

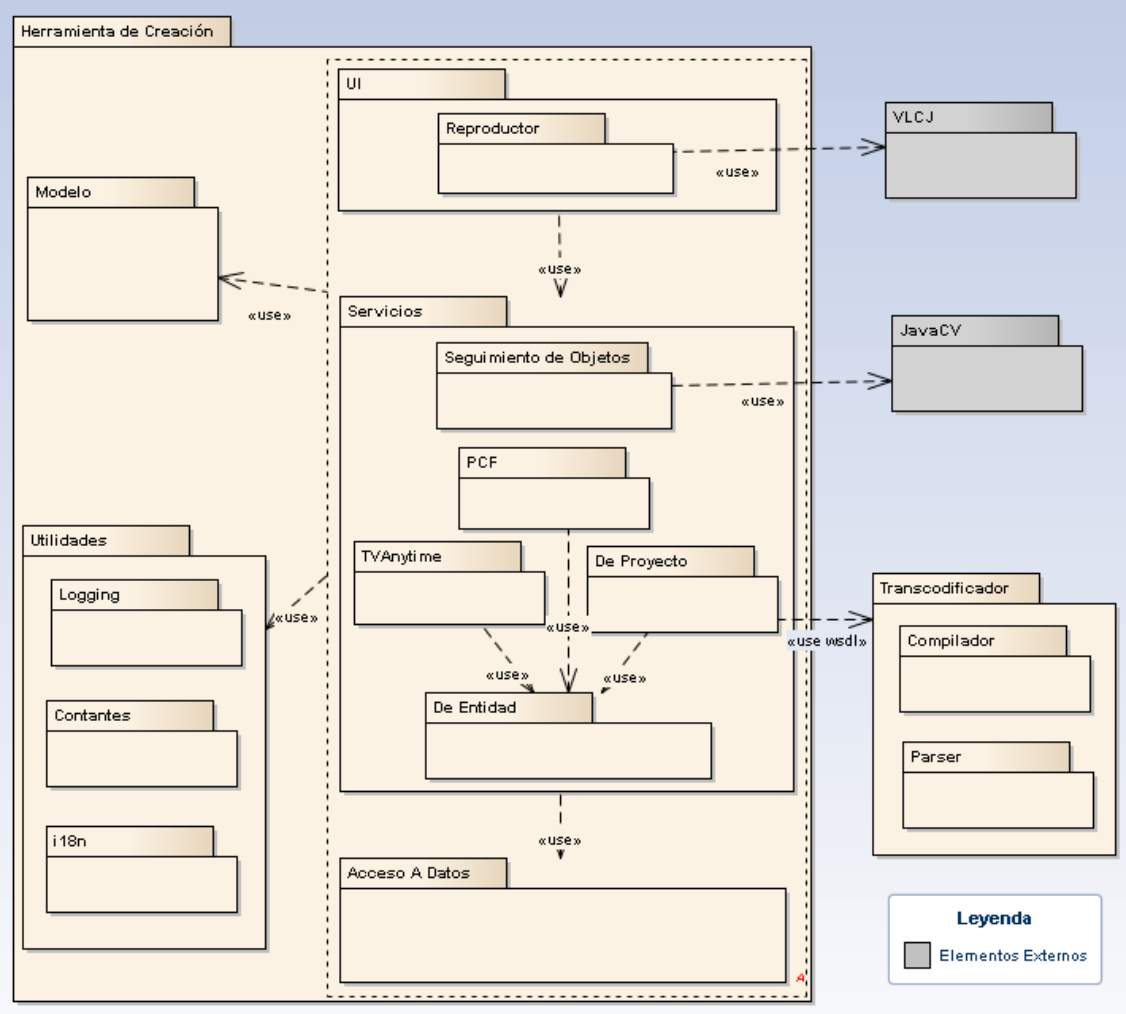

Figura 10 Diagrama de Uso de Módulos

3.6.1.2 Catálogo de Elementos

- **Herramienta de creación**: Herramienta de autor encargada del diseño de las aplicaciones. Estará dividida en cinco capas y será construida usando la plataforma de clientes enriquecidos o RCP[29] del proyecto Eclipse:

[29] Rich Client Platform. http://www.eclipse.org/home/categories/rcp.php Consultado: 24 de Abril de 2012

- **Modelo:** Esta capa es quizás conceptualmente la más importante. Representa una capa transversal en la herramienta de creación. Define las interfaces que deben implementar la capa de acceso a datos, los servicios de entidad, proyecto y de metadatos. Para cada uno de los objetos en el modelo existirá un POJO, una clase de acceso a datos y una clase que implemente la lógica específica de cada objeto.
- **UI:** Esta capa corresponde a la interfaz de usuario, es la puerta de entrada a todas las funcionalidades expuestas por la capa de servicios. Permitirá crear, abrir, guardar y cerrar proyectos. En esta capa se implementarán las herramientas para dibujar objetos y los formularios para ingresar los metadatos que enriquecerán los objetos seleccionados.
    - **Reproductor:** Este módulo servirá para reproducir el video. Tendrá las opciones de reproducir, pausar, parar**,** adelantar y retroceder rápidamente, adelantar y retroceder cuadro a cuadro, bajar y subir el volumen y enmudecido. Servirá como lienzo sobre el cual el usuario dibuje los objetos. Cada vez que se dibuje un objeto se habilitará un formulario para ingresar los metadatos correspondientes.
- **Servicios:** Esta capa es la encargada de implementar la lógica del negocio y de proveer servicios a la capa de UI definidos por el modelo.
    - **Seguimiento de Objetos:** Este módulo contiene los diferentes algoritmos para realizar el seguimiento de objetos en un flujo de video.
    - **TVAnytime:** Este módulo es responsable de almacenar y leer los metadatos en el estándar TV-Anytime. El formato de los archivos es TVA.
    - **PCF:** Este módulo es responsable de almacenar el diseño de la aplicación en un archivo con formato PCF. El resultado es un “pseudo-programa”, una descripción del cómo y cuándo deberían verse los objetos que se han enriquecido.
    - **De Proyecto:** Este módulo presta servicios relacionados con el modelo de datos del proyecto y además es el encargado de llamar el servicio web expuesto por el Transcodificador para transformar la aplicación que está siendo diseñada.

- **De Entidad:** Este módulo presta servicios relacionados con el modelo de las entidades. Para cada una de las entidades existe un clase de servicios la cual interactúa con la capa de acceso a datos.

✓ **Acceso A Datos:** Esta capa es la encargada de implementar el acceso y la persistencia de datos. Es usada por los servicios de entidades del modelo. También es responsable de la transaccionalidad.

✓ **Utilidades:** Está capa contiene utilidades usadas por las diferentes capas de la Herramienta de Creación.

- ***Logging*:** Este módulo contiene utilidades para depuración y registro errores.
- **Constantes:** Este módulo contiene las constantes generales a todo el sistema.
- **i18n:** Este módulo contiene utilidades para el manejo de mensajes para la internacionalización de la herramienta.

- **Transcodificador:** Este módulo es el encargado de generar la aplicación final que será transmitida al televidente de acuerdo a la plataforma seleccionada. Se aprovecharán partes de una herramienta construida en C# en el laboratorio de televisión digital que sirve para editar y transformar archivos PCF. Esta compuesto por dos submódulos:
  - ***Parser*:** Este módulo se encargará de analizar los archivos PCF y TVA, para extraer las propiedades del video y de las entidades y dejarlas disponibles para el compilador.
  - **Compilador:** Usará la información de salida del módulo anterior para generar el código final en una plataforma dada. Las aplicaciones cliente incluirán la mayoría de la información adicionada a las entidades esto con el fin de no depender en tiempo de ejecución de servidores para obtenerla. La única información que no será embebida serán los archivos externos que se relacionen con las entidades, estos serán referenciados mediante URLs.

    El compilador usará un mecanismo de plantillas. Las plantillas son archivos escritos en el lenguaje de una plataforma específica que definen el cómo se comportarán y como se verán los objetos en dicha plataforma.

- **VLCJ[30]:** Es una interfaz que permite usar el *framework* VLC[31] desde aplicaciones Java.
- **JavaCV[32]:** Es una interfaz para usar desde un programa escrito en Java algunas de las librerías más comúnmente utilizadas por investigadores en el campo de visión artificial.

3.6.1.3 Guía de Variabilidad

- **Internacionalización**: La herramienta de creación soporta múltiples lenguajes. Para cada componente los mensajes estarán almacenados en un archivo llamado plugin<localización>.properties.
- **Compilador – Plantillas**: El Transcodificador tendrá una preferencia para la localización de las plantillas. Adicional a esto para definir varias plantillas para una misma plataforma se deberá seguir el siguiente esquema para su localización:
  /ruta de localización de las plantillas/Templates/<plataforma><consecutivo>/

3.6.1.4 Justificaciones y decisiones de diseño

- **Herramienta de Creación y RCP**: Esta decisión particularmente conlleva una serie de ventajas muy visibles para el diseño de la arquitectura [58][64] si la comparamos con una implementación desde cero realizada solo con Java:
  - ✓ Favorece la portabilidad de la herramienta a múltiples plataformas por estar basada en Java y por proveer internacionalización por defecto.
  - ✓ Permite extender la funcionalidad de la aplicación por su arquitectura modular y flexible basada en la implementación de componentes débilmente acoplados llamados *plug-ins*.

[30] VLCJ. Java Framework for the VLC Media Player. http://code.google.com/p/vlcj/ . Consultado: Marzo 10 de 2012.

[31] VLC es un reproductor multimedia y un *framework* libre y de código abierto que reproduce la mayoría de archivos multimedia así como DVDs, CDs de Audio, VCDs y varios protocolos de *streaming*. http://www.videolan.org/vlc/index.html. Consultado: Marzo 10 de 2012.

[32] JavaCV. Java interface to OpenCV and more. http://code.google.com/p/javacv/ . Consultado: Marzo 10 de 2012.

[32]

- ✓ Los componentes son ensamblados en tiempo de despliegue y relacionados en tiempo de ejecución favoreciendo la modificabilidad.
- ✓ Con respecto a la implementación de la interfaz gráfica permite utilizar y extender una gran de componentes ya existentes con solo editar archivos de configuración liberando a los desarrolladores de su implementación.

NetBeans también ofrece una plataforma muy similar a la de Eclipse para el desarrollo de clientes ricos. La elección en este caso es solo cuestión de gusto y experiencia con la herramienta.

- **Arquitectura por capas:** Aunque este estilo puede perjudicar el rendimiento y hacer un poco difícil la separación de funciones de las capas favorece la localización de las modificaciones, permitiendo modificar o implementar los algoritmos de seguimiento de objetos de tal forma que los cambios afecten solamente a sus capas vecinas.

- **Modelo**: El modelo es un conjunto de interfaces que define todo en la herramienta de creación esto con el fin de separar la implementación del uso de las mismas facilitando el cambio de funcionalidades en un futuro. Tratando de afectar los menos módulos posibles y permitiendo reemplazar cualquier parte por implementaciones externas.

- **Seguimiento de objetos y JavaCV**: Una de las tareas más complicadas en la edición de video es la extracción y el seguimiento de objetos. La primer posibilidad que tiene el usuario es dibujar cuadro a cuadro el área donde se encuentra la entidad a seleccionar. Pero de esta forma la interacción es muy intensa y por lo tanto pesada y absorbente. Acá es donde los algoritmos de seguimiento de objetos pueden ser de gran ayuda para reducir el esfuerzo del usuario. Los algoritmos usados para este propósito son bastante complejos y necesitan de un gran poder de cómputo. Por esta razón se decidió usar JavaCV que da la oportunidad de usar librerías que están escritas fundamentalmente pensando en el rendimiento, incluso haciendo uso de propiedades específicas de los

procesadores, lo que las hace ideales para cumplir con los requisitos de rendimiento de la aplicación.

- **VLCJ y JavaCV**: Ambas librerías tienen un papel fundamental en la arquitectura. Son interfaces de otras librerías soportadas en múltiples plataformas dándole consistencia a la idea de la portabilidad de la herramienta de creación. Y son librerías escritas en lenguajes con mayor velocidad de procesamiento que el de Java, favoreciendo otros atributos de calidad como los son el rendimiento y la usabilidad.

- **TV-Anytime**: Después de analizar los requerimientos que el modelo de metadatos debía cumplir (ver Anexo A.1.3 Requisito Funcional RF3.2), se decidió que la mejor solución no correspondía a un único estándar sino a la unión de propiedades de diferentes especificaciones de metadatos debido a que cada una responde a necesidades determinadas y uno solo no podría cumplir todos los requisitos. En este sentido un enfoque híbrido utilizando los estándares TV-Anytime y MPEG-7 es el que mejor se adapta.

  Por su calidad, ricos vocabularios, esquemas elaborados y por ser altamente aceptado y reconocido internacionalmente en el dominio de la televisión, se decidió usar TV-Anytime como el estándar de almacenamiento de metadatos. Esto facilitará la integración de los componentes y aumentará la interoperabilidad con otras posibles sistemas externos. TV-Anytime brinda una forma de describir el contenido además de su segmentación temporal, cuadros, tomas, escenas, etc. mientras que MPEG-7 describe las propiedades de bajo nivel presentes en contenidos multimedia como histogramas, nivel de volumen, etc. Incluso la especificación de TV-Anytime se complementa mediante un subconjunto de la norma MPEG-7 que ayuda a describir un programa de televisión con un gran nivel de detalle.

  Otro estándar de metadatos que se estudió fue el estándar MPEG-21. Escoger esta especificación implicaba una solución más compleja y más genérica debido a

que abarca mucho más que solo la televisión, lo que implicaba un mayor esfuerzo de desarrollo y una subutilización de funcionalidades.

- **Portable Content Format**: Para seleccionar el lenguaje en el cual se almacenaría el diseño de las aplicaciones y que serviría para la comunicación entre la herramienta de creación y el Transcodificador, se evaluaron los lenguajes Ginga-NCL, SMIL y PCF. Estos fueron preseleccionados dentro de los diferentes lenguajes expuestos en el marco teórico debido a que son estándares abiertos basados en XML y adicionalmente son lenguajes declarativos. La utilidad de que sean XML radica en que es fácil crear hojas de transformación XSL y/o analizadores de este tipo de documentos que permiten hacer conversiones a otros lenguajes, por ejemplo XML-a-Java o XML-a-HTML.

  La siguiente tabla contiene los criterios de evaluación[33]

  Independencia de la Plataforma: El lenguaje debe ser lo más genérico posible. La plataforma para la que fue desarrollado no debe limitar su funcionalidad.
  Extensibilidad: Capacidad para adicionar funcionalidad.
  - ✓ Personalización: Capacidad para modificar el estándar.
  - ✓ Gráficas: Capacidad para dibujar formas programáticamente.
  - ✓ Ejecución: Posibilidad de ejecutar las aplicaciones definidas.
  - ✓ Sincronización de Medios: Capacidad para definir eventos que involucren recursos multimedia a nivel de milisegundos.

[33] Los criterios son una adaptación de los usados en [35] para realizar una comparación entre lenguajes de metadatos.

Tabla 4 Evaluación de formatos para la portabilidad

Lenguaje / Criterio	Ginga-NCL	SMIL	PCF
Independencia de la Plataforma	Creado para la plataforma brasilera SBTVD.	Creado para describir presentaciones multimedia. Depende de un reproductor.	Creado para transferir contenido interactivo independiente de la plataforma.
Extensibilidad	Permite separar diseño y contenido en diferentes archivos.	Permite separar diseño y contenido en partes del mismo archivo.	Permite definir por medio de URLs cualquiera de sus partes. Proporcionando una gran separación de funciones.
Personalización	No.	No.	Provee esquemas de XML que permiten definir nuevas etiquetas personalizadas.
Gráficas	No permite crear figuras geométricas, se deben usar imágenes.	No permite crear figuras geométricas, se deben usar imágenes.	Permite crear rectángulos, polígonos y elipses.
Ejecución	Es posible ejecutarlo usando un simulador34.	Es posible ejecutarlo usando un reproductor 35 o alguna extensión para navegadores.	No es posible ejecutarlo, este lenguaje es de descripción y para el intercambio de datos.
Sincronización de medios	Si tiene sincronización pero no a nivel de milisegundos.	Si tiene sincronización pero no a nivel de milisegundos.	Si. Simulada. Es posible usando Action Language36.

[34] Exhibitions tools. GINGA-NCL VIRTUAL STB. http://www.gingancl.org.br/en/ferramentas. Consultado: 3 de Abril de 2012

[35] Ambulant Open SMIL Player. http://www.ambulantplayer.org. Consultado: 3 de Abril de 2012.

[36]Action Language es el lenguaje definido en el estándar PCF para especificar una secuencia de acciones en respuesta a un evento o una transición de un estado.

El lenguaje seleccionado finalmente fue PCF debido a que es mejor en la mayoría de los criterios, siendo el más importante la independencia de la plataforma, afectando positivamente la portabilidad de las aplicaciones diseñadas. Otros criterios significativos son la capacidad de representar figuras geométricas y la representación del tiempo en milisegundos.

- **Compilador y Plantillas:** Como se mencionó anteriormente se decidió incluir en la aplicación generada toda la información de metadatos contenida de los archivos PCF y TVA en la misma aplicación evitando la dependencia de archivos externos y quizás en servidores externos.

  Otra opción que se evaluó era la de transformar la aplicación diseñada tiempo de ejecución dándole a las plantillas la capacidad de interpretar el contenido de los archivos PCF y TVA en el dispositivo del cliente, pero este tipo de implementaciones introduciría un sobrecosto en tiempo de ejecución que no es viable teniendo en cuenta las limitaciones en recursos de los decodificadores.

  Las plantillas son una forma de implementar el patrón arquitectónico llamado “Métodos modelo”. Una plantillas contiene secciones que son reemplazadas en tiempo de ejecución manteniendo el resto de la plantilla invariante. Este mecanismo permitirá adicionar diferentes aplicaciones para una misma plataforma, así por ejemplo un desarrollador podría optar por señalar las entidades con un diseño diferente al dibujado usando alguna especie de marca. Otro podría crear algún tipo de funcionalidad relacionada con redes sociales, como compartir con amigos una entidad que un televidente seleccione. Y otro podría crear una tienda virtual y adicionar la entidad al carrito de compras del televidente.

3.6.2 Modelo de datos

La siguiente figura representa la información que se debe almacenar. El triángulo azul representa un elemento presente en un video y cada cuadro representa un cuadro de video. Como se aprecia en la figura el objeto va cambiando de posición respecto del

cuadro anterior. El usuario debe dibujar la forma correspondiente al objeto y seleccionar un algoritmo de seguimiento de objetos para encontrar las posiciones del objeto en los cuadros siguientes y posteriormente asignarle la información que corresponda al triángulo azul.

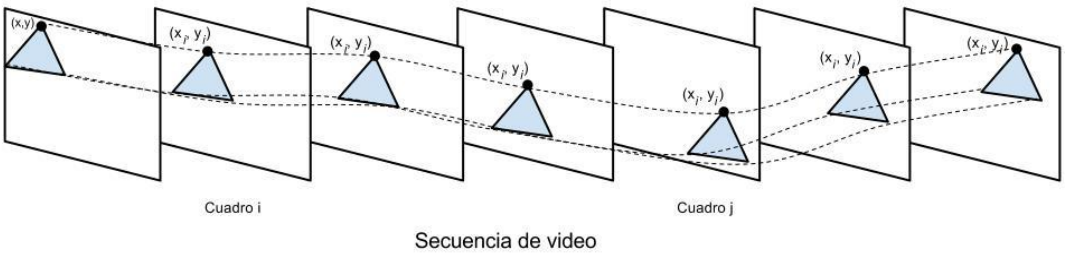

Figura 11 Ubicación espacio-temporal de un objeto

El siguiente es el modelo de datos sobre el que se basan los proyectos de diseño de aplicaciones en la herramienta de creación. Contiene información exclusiva de cada proyecto necesaria para llevar a cabo las funciones de gestión de cada aplicación que se diseñe, particularmente el tiempo y la localización de objetos seleccionados además de los metadatos de cada objeto enriquecido.

3.6.2.1 Presentación Primaria

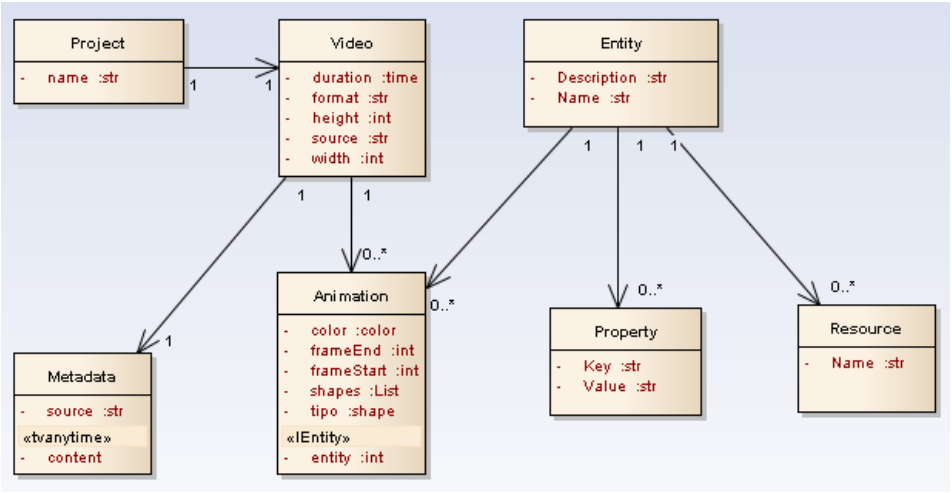

Figura 12 Modelo de datos

3.6.2.2 Catálogo de Elementos. Para cada uno de los siguientes objetos existe una clase tipo servicio encargada de implementar la lógica asociada al objeto y una clase de acceso a datos encargada de la persistencia del mismo.

- ***Project*:** Objeto que representa un proyecto de diseño de una aplicación para televisión interactiva. Se identifica con un nombre que será solicitado al crear un proyecto nuevo mediante la selección de un video.
- ***Metadata*:** Objeto que contiene la referencia a los metadatos en formato TVA.
- **Video:** Objeto que representa el video sobre el que se diseña una aplicación. Almacena: formato, duración, tamaño en pixeles, ubicación del video.
- ***Animation*:** Esta entidad representa uno de los objetos dibujados en un video. Recoge la posición en la pantalla del objeto usando el atributo *shape* y el tiempo usando los atributos de *frameStart* y *frameEnd*, es decir, sirve para almacenar la posición del objeto en cada cuadro y el tiempo durante el que está presente. También permite especificar el color del dibujo y el tipo de dibujo.
- ***Entity*:** Este es el objeto base de la arquitectura. Contiene la información general del objeto seleccionado. Tiene un nombre, un tipo y una descripción.
- ***Property*:** Representa un metadato. Se define mediante una dupla atributo-valor.
- ***Resource*:** Objeto que representa la relación con archivos asociados a los objetos.

3.6.2.3 Guía de Variabilidad

- **Tipo de dibujo**: El atributo tipo del objeto *animation* depende de la herramienta de dibujo implementada. Cada herramienta retornará una cadena de caracteres con un nombre que identificará el tipo de dibujo usado.

3.6.2.4 Decisiones de Diseño

- ***Entity***: La clase *entity* no posee información sobre la relación de un elemento seleccionado con un video en particular. Contiene información general del elemento que puede ser referenciada desde otras partes del mismo video o incluso en otros videos. El vínculo entre una entidad y un video específico se da a través del atributo *entity* de la clase *Animation*.

3.6.3 Vistas De Comportamiento

3.6.3.1 Presentación Primaria

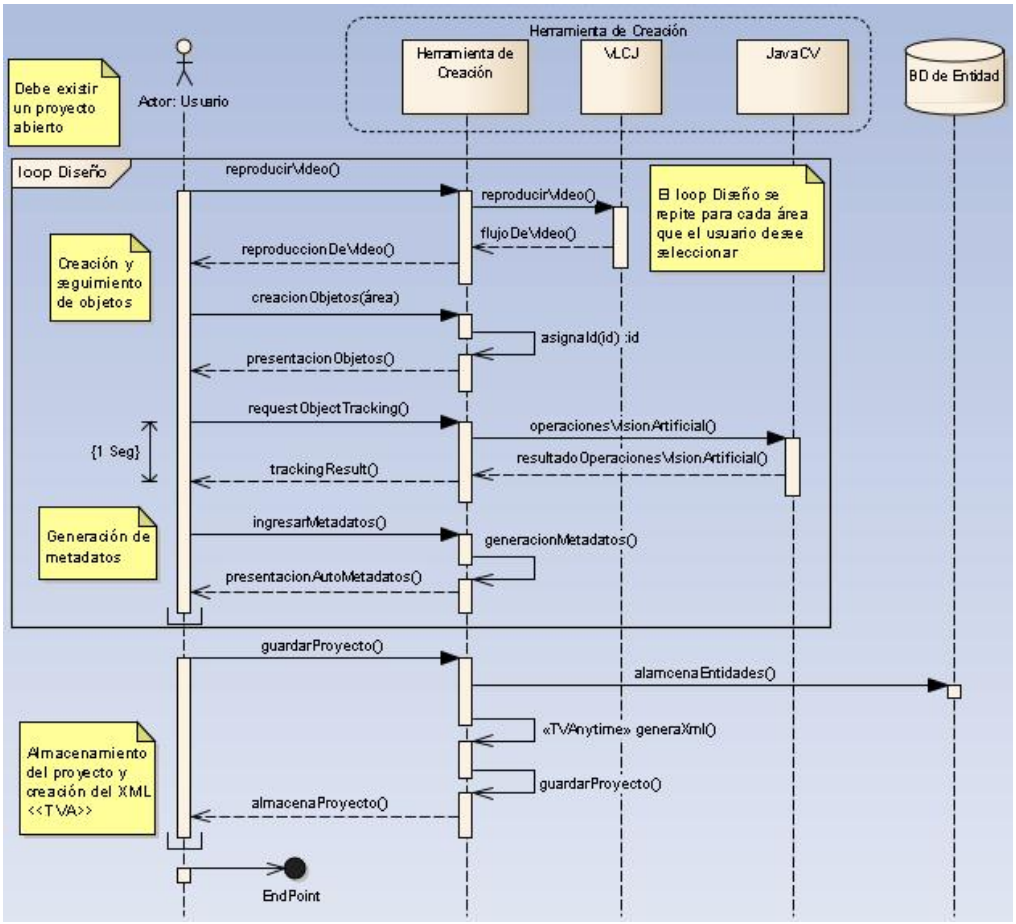

Figura 13 Diagrama de secuencia básico para el diseño de aplicaciones

La Figura 13 y la Figura 14 representan las secuencias de interacciones más importantes que ocurren durante el diseño y la generación de una aplicación interactiva. Sirven para entender lo que ocurre entre las partes definidas en la vista de uso de módulos con el fin de cumplir con los casos de uso seleccionar objetos, adicionar Metadatos, seguir objetos y generar aplicación.

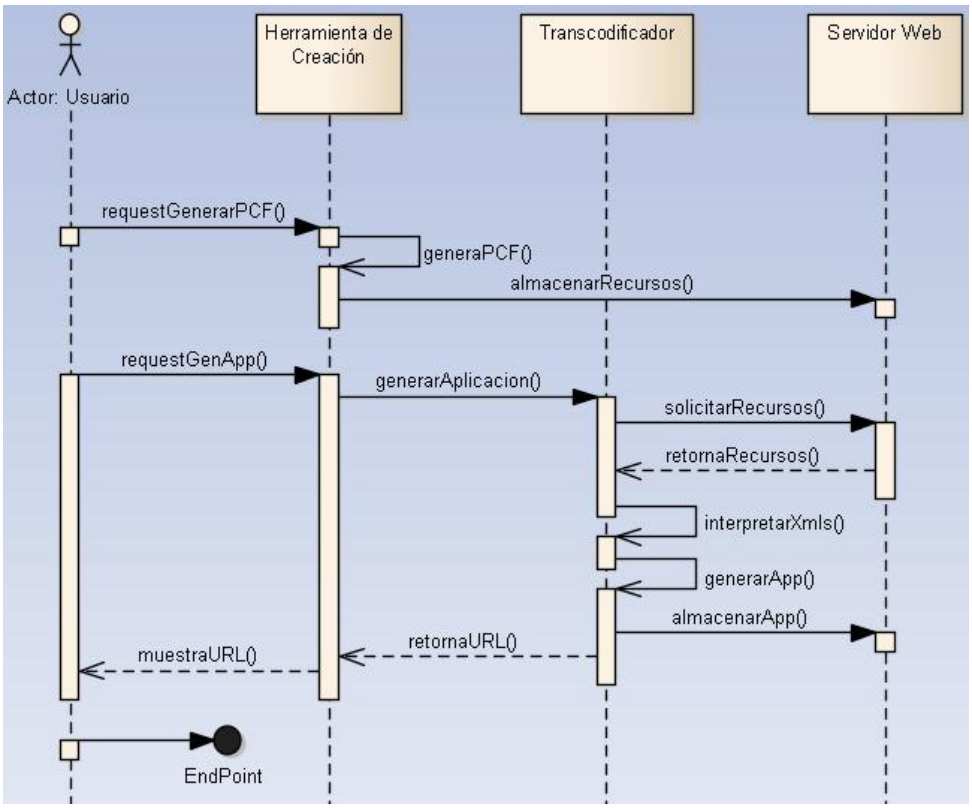

Figura 14 Diagrama de secuencia básico para generar aplicaciones

3.6.3.2 Catálogo de Elementos

- **Usuario:** Hace referencia al usuario encargado de crear las aplicaciones.
- ***Loop* Diseño:** Agrupa las tareas que debe repetir el usuario para cada objeto que se desee enriquecer.
- **BD de Entidad:** Representa el repositorio donde se almacenará el modelo de datos expuesto en la vista anterior.
- **Servidor Web:** Este componente servirá para alojar los archivos que representan los proyectos de diseño y a la vez para compartir datos entre la herramienta de creación y el Transcodificador.

Para una descripción de los elementos no listados en esta sección consulte el catálogo de elementos de las vistas anteriores.

3.6.3.3 Guía de Variabilidad. Consulte la guía de variabilidad de la vista siguiente de componentes y conectores.

3.6.4 Vistas De Componentes Y Conectores

3.6.4.1 Presentación Primaria

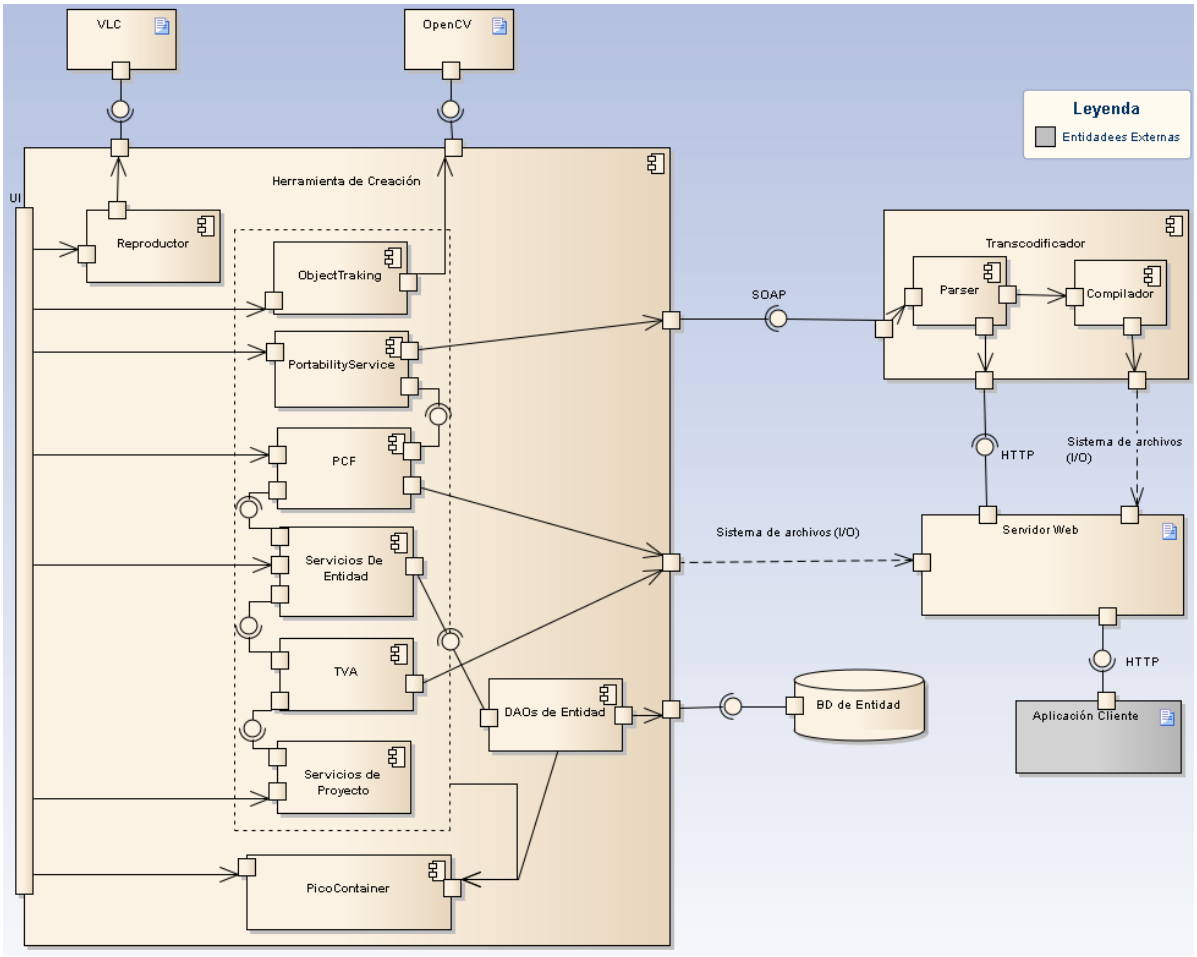

Figura 15 Diagrama de Componentes

3.6.4.2 Catálogo de Elementos

- **VLC:** Este componente presta servicios para el manejo de archivos de vídeo de forma nativa. El componente reproductor hace uso extensivo de éste.

- **OpenCV[37]:** Este componente presta servicios de visión artificial a las diferentes implementaciones de seguimiento de objetos.
- **ObjectTracking:** Representa una instancia de un algoritmo de seguimiento de objetos.
- **PortabilityService:** Este servicio define una clase abstracta encargada de solicitar la transformación de aplicaciones al servicio web del Transcodificador. Ver la guía de uso de la interfaz TranscoderWebService para mayor información numeral 3.6.6.2. Antes de hacer la invocación al servicio web hace uso del servicio de PCF para generar el diseño de la aplicación y garantizar que la aplicación exista en dicho archivo.
- **PicoContainer[38]:** Es un contenedor que soporta el patrón de Inversión de Control. Administra las dependencias en tiempo de ejecución.
- **Transcodificador:** Este componente es el encargado de la generación de aplicaciones. Su funcionalidad se expondrá mediante un servicio web llamado TranscoderWebService y esta descrito en la sección 3.6.6.2.
- **Aplicación Cliente:** Este componente representa una aplicación ya finalizada y transmitida a un televidente o accedida por este, realizando peticiones para acceder a los recursos asociados a los objetos de la aplicación.

Para una descripción de los elementos no listados en esta sección consulte el catálogo de elementos de las vistas anteriores.

3.6.4.3 Guía de Variabilidad

- **Seguimiento de Objetos**: El sistema permite la adición de diferentes algoritmos de seguimiento de objetos. Para facilitar dicha adición cada algoritmo deberá implementar la interfaz ObjectTracking. Interfaz que está documentada en la sección 3.6.6.1.

- **Acceso a Datos**: Se podrá usar cualquier repositorio de datos, se deja libre a la implementación del modelo, por ejemplo se podría almacenar en archivos XML o

[37] OpenCV. http://opencv.willowgarage.com/wiki/. Consultado: 10 Mayo 2012

[38] PicoContainer. http://picocontainer.codehaus.org/index.html. Consultado: Marzo 12 de 2012

en bases de datos relacionales. Lo importante es mantener el patrón DAO propuesto para que los clientes no accedan a los datos directamente.

- **PortabilityService**: Las implementaciones concretas de esta clase corresponderán una a una con las plataformas de televisión implementadas en el Transcodificador. Cada plataforma en el Transcodificador deberá tener un código que será usado por los clientes para transformar sus diseños a esa plataforma.

- **Transcodificador**: Existirán dos parámetros configurables en la herramienta de creación relacionados con el Transcodificador:
  - ✓ El URL del Transcodificador a usar.
  - ✓ Y el URL base del servidor web donde se alojarán los archivos PCF, TVA y demás recursos del proyecto.

- **Servidor Web**: La arquitectura no requiere de una implementación en particular de un servidor web. Este se deja a elección al momento de despliegue.

- **Número de Instancias**: El número de instancias de la herramienta de creación, del Transcodificador y del servidor web es completamente variable. La estructura se puede acomodar a cualquier distribución dependiendo de las necesidades de uso.

3.6.4.4 Justificación

- **Inyección de Dependencias y PicoContainer:** Como se definió en la vista de módulos la interacción entre las diferentes capas se da usando las interfaces del modelo y no implementaciones concretas. Por esta razón se hace necesario un mecanismo que permita en tiempo de ejecución gestionar las dependencias sin hacer invocaciones a implementaciones específicas. Dicho mecanismo se logra implementando el patrón de Inyección de Dependencias que es una forma de Inversión de Control (IoC).

PicoContainer es el contenedor de IoC escogido para gestionar las dependencias. Su selección se debe a que está libre de dependencias externas, es muy fácil de configurar y es muy liviano en comparación con otros contenedores como Spring[39] y Google Guice[40], que aunque más robustos no son indicados debido a que adicionarían un gran carga adicional para el escaso número de clases de la herramienta de creación.

- **OpenCV**: Para mejorar el rendimiento y la usabilidad se buscaron métodos para el seguimiento de objetos debido a que permiten acelerar la selección de los mismos. En este referente se encontraron varias librerías de procesamiento matemático y de visión artificial, como Matlab[41], SimpleCV[42], VXL[43], LTI[44] y OpenCV entre otras.

  OpenCV es la que ofrece mayores ventajas según comparaciones realizados por Bradski [62] y los desarrolladores de SimpleCV[45].

  OpenCV es una librería de visión artificial que contiene cientos de funciones para una amplia gama de áreas en el proceso de visión, como reconocimiento de objetos, calibración de cámaras, visión estereoscópica y visión robótica [62][63].

  Se escogió esta librería por tres razones principales:

  1 Es multiplataforma, contando con versiones para GNU/Linux, Mac y Windows.

[39] Spring Framework. http://www.springsource.org/spring-framework Consultado: Marzo 12 de 2012

[40] Google-guice. http://code.google.com/p/google-guice/ Consultado: Marzo 12 de 2012

[41] Matlab. http://www.mathworks.com/products/matlab/ Consultado: Marzo 12 de 2012

[42] SimpleCV. http://simplecv.org/ Consultado: Marzo 12 de 2012

[43] VXL. http://vxl.sourceforge.net/ Consultado: Marzo 12 de 2012

[44] LTI. http://ltilib.sourceforge.net/doc/homepage/index.shtml Consultado: Marzo 12 de 2012

[45] OpenCV vs. Matlab vs. SimpleCV. http://simplecv.tumblr.com/post/19307835766/opencv-vs-matlab-vs-simplecv. Consultado: 10 Mayo de 2012.

45

2 Es libre, cuenta con una gran comunidad que constantemente hace aportes para su evolución.
3 Rendimiento, saca provecho de diferentes arquitecturas de procesamiento.

- **Invocación a Procedimientos Remotos:** Se escogió el estilo RPC porque brinda un alto grado de interoperabilidad entre el Transcodificador y cualquier entidad interna o externa a la arquitectura. También da flexibilidad en términos de las plataformas de implementación de esas entidades.

- **Servidor Web**: Se escogió utilizar un servidor web como medio para compartir datos o pasar parámetros entre la herramienta de creación y el Transcodificador debido a su facilidad de instalación y de puesta en marcha que no le agregan mayor complejidad a la arquitectura.

- **Número de Instancias**: El estilo de cliente-servidor seleccionado combinado con otros factores mencionados anteriormente como el uso de RPC, aunque impacta desfavorablemente el rendimiento, lo que se obtiene en escalabilidad y la flexibilidad del sistema es de mayor valor para los objetivos de la arquitectura. Se pueden tener múltiples instancias de la herramienta de creación haciendo uso de un solo servidor web y un solo Transcodificador, o se pueden tener varios servidores web atendiendo a varios grupos de herramientas de creación y de Transcodificadores, entre otras variaciones.

3.6.5 Vista De Despliegue

La siguiente figura muestra la asignación de los componentes de la arquitectura en términos físicos, es decir de los equipos en los que se instalarán y ejecutarán.

3.6.5.1 Presentación Primaria

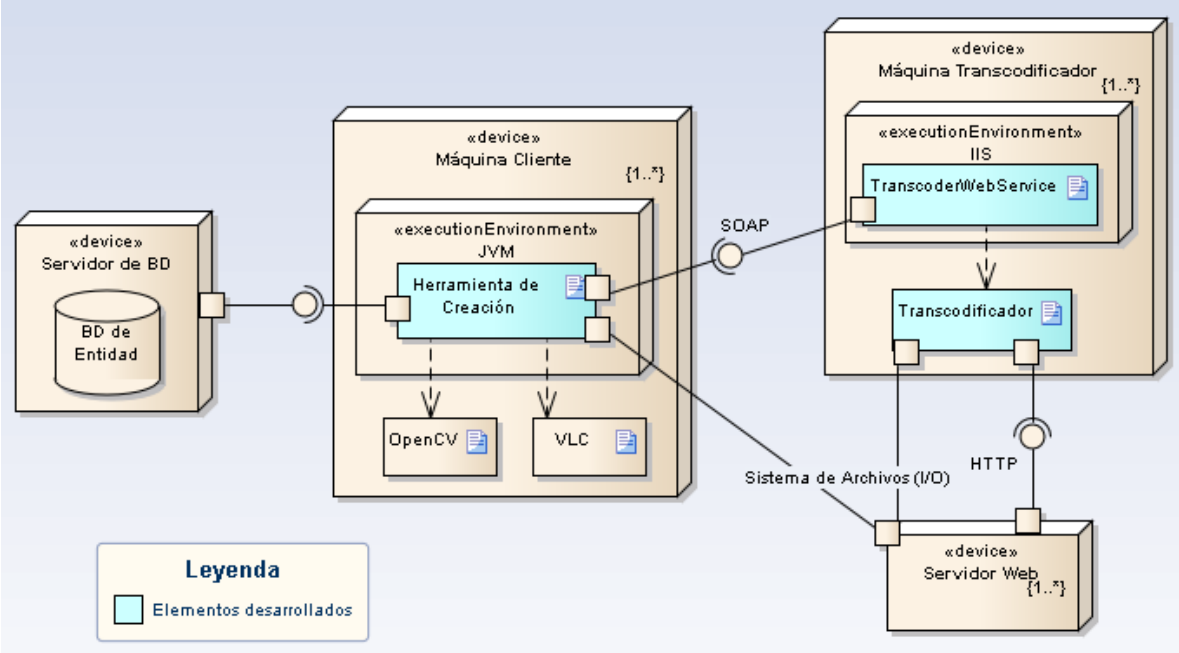

Figura 16 Diagrama de despliegue

3.6.5.2 Catálogo de Elementos

- **Servidor de BD:** Nodo que representa la máquina en la que se almacenará el modelo de datos.
- **Máquina Cliente:** Nodo que representa la máquina del usuario que diseña las aplicaciones. Puede tener instalado uno de los siguiente sistemas operativos: GNU/Linux, Mac o Windows.
- **JVM:** Máquina virtual de Java sobre la que corre la herramienta de creación. Debe ser la versión 1.6 o superior.
- **Máquina Transcodificador:** Nodo que representa la máquina donde se ejecuta el Transcodificador. Debe correr sobre un sistema operativo basado en *windows* y tener instaladas las herramientas de .NET 4.0.
- **IIS:** Servidor web de Microsoft donde se ejecutará el servicio web TranscoderWebService.

3.6.5.3 Guía de Variabilidad

- **Servidor de BD:** Como el acceso de datos y la persistencia de los mismos no se ha especificado en ninguna tecnología en particular la especificación de los requerimientos de este nodo queda abierta para las posibles implementaciones.

- **Máquina Cliente**: Aunque la herramienta de creación fue diseñada para correr en equipos de escritorio, la configuración será considerada variable, pues el rendimiento mencionado dependerá de la calidad del video que se esté manipulando.

3.6.6 Diseño De Interfaces

3.6.6.1 Interfaz para el Seguimiento de Objetos. La siguiente es la documentación de la interfaz definida en la vista de módulos llamada ObjectTracking que los algoritmos para el seguimiento de objetos deben implementar. Esta es una de las interfaces más importantes de la arquitectura ya que se espera que existan trabajos futuros implementando y mejorando algoritmos ya existentes.

Tabla 5 Interfaz ObjectTracking

Identificación de la Interfaz	ObjectTracking
Caso de uso relacionado	CU3.2 Seguir Objetos
Propósito de la interfaz	Definir cómo será la interacción de la herramienta de creación y los algoritmos de seguimiento de objetos.
Operación	*track(long currentTime, Shape template)*

Propósito de la Operación	Encontrar las sucesivas apariciones de objeto en un video a partir de un tiempo de dado.
Parámetros de entrada	● currentTime: Tiempo en milisegundos que especifica el punto del video desde el cual se empezará a buscar el objeto. ● template: Imagen del objeto a buscar.
Parámetros de salida	● Lista de posiciones del objeto. Cada posición de la lista representa un cuadro.
Precondiciones	● El tiempo debe ser menor a la duración del video. ● La imagen no puede ser nula
Poscondiciones	● El usuario recibirá un mensaje en caso de no encontrar ninguna coincidencia. ● En caso de encontrar coincidencias el algoritmo retornará una lista con las posiciones del objeto en los cuadros siguientes.
Manejo de Errores	Los errores serán manejados internamente por cada implementación. Se escribirá en el log de la aplicación el error y se retornará una lista vacía.
Tipos de Datos	Shape: Es el conjunto de coordenadas que definen el área donde se encuentra un objeto.
Guía de Uso	*Shape template = AnimationServices.getSelected();* *animation.getShape(activaPlayer.getCurrentFrame());* *long currentTime = player.getCurrentTime();* *List matches = track(currentTime, template);* *video.addMatches(matches);*

3.6.6.2 Interfaz del Transcodificador. La siguiente es la documentación de la interfaz del servicio web del Transcodificador llamada TranscoderWebService. Esta interfaz es la más importante ya que es la puerta de entrada a la generación de aplicaciones y a que puede

ser usada por entidades externas al entorno natural de la arquitectura, por eso su definición y documentación es crucial para que las diferentes entidades sepan cómo elaborar sus peticiones y que esperar como respuesta.

Tabla 6 Interfaz TranscoderWebService

Identificación de la Interfaz	Este es un servicio web tipo SOAP[46] llamado TranscoderWebService
Caso de uso relacionado	CU4 Generar Aplicación
Propósito de la interfaz	Permitir generar una aplicación para alguna de las plataformas de televisión interactiva a partir de un archivo PCF.
Operación	*doWork(Integer platform, String projectURL, String resourcesURL, Integer template)*
Propósito de la Operación	Genera el código fuente para una aplicación de ITV. Esta operación retorna una cadena de texto o *String* con el URL desde la cual los archivos generados quedan disponibles
Parámetros de entrada	● platform: identificador de la plataforma de ITV destino. ● projectURL: URL del proyecto donde se encuentra el archivo PCF a transformar. ● resourcesURL: URL que será usado para ligar los recursos a la aplicación. ● template: identificador de las plantillas a usar para la aplicación.
Parámetros de salida	● String con la URL de la aplicación generada
Precondiciones	● El identificador de la plataforma objetivo debe estar definido en

[46] SOAP (S*imple Object Access Protocol*). Protocolo estándar para el intercambio de mensajes sobre HTTP, permitiendo que dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de mensajes XML.

	las constantes. • La URL del proyecto debe existir y contener un archivo con extensión PCF válido. • La URL de los recursos puede no estar disponible al momento de generar la aplicación pero cuando la aplicación sea usada por un televidente ésta deberá estar disponible en caso de que la aplicación haga uso de recursos externos para enriquecer los objetos. • Debe existir un conjunto de plantillas para la plataforma selecciona con el id especificado en este parámetro.
Poscondiciones	Una ejecución exitosa retorna una URL para cada aplicación.
Invariantes	El archivo fuente de la aplicación se mantiene sin modificaciones.
Manejo de Errores	Los errores serán manejados internamente por el Transcodificador. El cliente recibirá un mensaje de texto explicando la posible causa del error en el parámetro de salida.
Constantes	En el capítulo siguiente sobre la implementación del prototipo se definirán las primeras constantes correspondientes a las plataformas desarrolladas para el prototipo.
Variabilidad	Se pueden adicionar nuevas plataformas y nuevas plantillas al servicio. Cuando estas se adicionen se deben agregar los valores correspondientes en la sección de constantes.
Atributos de Calidad	Interoperabilidad: Al usar un estándar, en este caso como PCF, los usuarios no ven limitada su capacidad de elección a un determinado proveedor, sino a todos aquellos que cumplen con él, por tanto, se crean productos que son interoperables con mayor posibilidad de interconexión e integración con otros sistemas.
Justificación y decisiones de diseño	**C#:** El servicio de transformación dependerá de desarrollos ya existentes en esta materia en el laboratorio y los cuales se implementaron en tecnologías de Microsoft.

	Web Service: Al ofrecer esta funcionalidad como servicio web SOAP se provee la interoperabilidad requerida garantizando el acceso a una gran variedad de usuarios potenciando la interacción entre pares de diferentes plataformas. Se escogió usar SOAP por sobre otros métodos para comunicar componentes distribuidos como CORBA[47] o IIOP[48] por su facilidad de implementación y a que la naturaleza de la interacción no supone un uso excesivo de la interfaz. **Parámetros de entrada:** El envío de un URL en lugar del propio archivo PCF como parámetro también se debe a la restricción de negocio debido a que la herramienta existente ya soportaba el manejo de estos archivos usando URIs para su localización así que era un paso más natural pasar el URL de donde se encuentran los archivos que cambiar el código para que soportara el paso de archivos XML.
Guía de Uso	*TranscoderWebServiceProxy tws = new TranscoderWebServiceProxy("http://x.x.x.x/TranscoderWebService");* *String resourcesURL = "http://y.y.y.y/aplicationx/resources";* *String projectURL = "http://z.z.z.z/aplicationx/aplicationx.pcf";* *String result = tws.doWork(1, projectURL, resourcesURL, 1);*

[47] CORBA (Common Object Request Broker Architecture).

[48] IIOP (Internet Inter-ORB Protocol)

CAPÍTULO 4 DESARROLLO DEL PROTOTIPO

4.1 INTRODUCCIÓN

En este capítulo se describen en grandes rasgos las tareas realizadas durante la implementación de un prototipo de la arquitectura presentada en el capítulo anterior, así como las herramientas utilizadas en su construcción. También se darán detalles de la implementación de los diferentes componentes de la arquitectura.

Es importante resaltar que el principal objetivo de implementar este prototipo es servir como vehículo para probar como los atributos de calidad son logrados a través de las decisiones de diseño involucradas en la arquitectura y que por lo tanto no constituye una versión final de la arquitectura.

4.2 DETALLES DE IMPLEMENTACIÓN

Al igual que en algunos de los proyectos explorados relacionados con la selección de objetos en un video [7][9][11][19], se optó por una estrategia de simulación mediante la representación de los objetos en una capa superior al video donde se marcan los personajes, los productos u otros objetos que se puedan ver en un momento dado de la transmisión y que sea de interés resaltar e enriquecer, todo esto sin modificar directamente el video original. La interactividad de estos objetos dependerá de lo que se desarrolle en cada plantilla. De esto se hablará en la sección sobre la implementación del Transcodificador.

La implementación se hizo usando una estrategia incremental y consto de 4 etapas descritas a continuación:

Etapa 1: Creación del proyecto

- Instalación de la base de datos y creación del modelo de datos
- Implementación de la Herramienta de Creación inicialmente con el reproductor de video.

Etapa 2: Manual con rectángulos

- Implementación de la selección de objetos dibujando rectángulos cuadro a cuadro (solo era permitido uno al tiempo).
- Se mostraban metadatos que habían sido adicionados directamente a los objetos en la base de datos.
- Creación de la primera versión de una aplicación usando el formato PCF.
- Implementación de la primera versión de un convertidor de PCF a MHP a manera de aplicación de escritorio.

Etapa 3: Manual con polígonos

- Implementación de la selección de objetos dibujando polígonos.
- Adición de metadatos usando el formato TVA.
- Implementación de la transformación de PCF a Google TV

Etapa 4: Semi-Automática

- Validación de los archivos PCF y TVA desde la creación de los mismos
- Seguimiento de objetos.
- Extracción automática de metadatos.
- Implementación de la transformación de PCF a HbbTV
- Implementación del servicio web para el Transcodificador

4.3 HERRAMIENTA DE CREACIÓN

4.3.1 Herramientas Utilizadas

Para el desarrollo de este componente se utilizaron las siguientes herramientas (algunas fueron explicadas con mayor detalle en el capítulo anterior):

- MacBook pro con procesador 2.6GHz Intel Core 2 Duo con 4GB de RAM y sistema operativo Snow Leopard 10.6.8.
- JDK 1.6 64 bits: Conjunto de Herramientas de programación para Java.
- Eclipse Indigo 3.7: Como ambiente de desarrollo integrado. Con los *plug-ins* de RCP y el paquete adicional Babel para internacionalización.
- Postgres 9.1: Como motor de base de datos.
- Apache 2.2: Servidor Web para alojar los proyectos creados y sus recursos
- OpenCV 2.3.1: Librería para realizar el seguimiento de objetos.
- VLC 2.1.0: Librerías para la reproducción del video
- Hibernate 3: Librería para facilitar la persistencia del modelo de datos.
- JAXB: Arquitectura para mapear clases de Java a representaciones XML y viceversa. Es usado para mapear la parte del modelo de datos correspondiente al manejo de proyectos de diseño de aplicaciones, Es decir las entidades: Project, Video y *Animation* y Metadata. También es usado para generar los archivos PCF y TVA.
- VLCJ 2.1.0: Librería que permite usar el *framework* VLC desde aplicaciones Java.
- JavaCV 20120118: Librería que permite usar OpenCV desde aplicaciones Java.
- PicoContainer 2.14.1: Librería que implementa un contenedor para IoC.

Eclipse

La herramienta de creación, consta de varios *plug-ins*, cada uno es un proyecto en eclipse. Cada uno de estos *plug-ins* se mapea con algún módulo o componente de la arquitectura.

Módulo o Componente	***Plug-in***
Modelo	edu.eafit.maestria.activa.model
Utilities	edu.eafit.maestria.activa.utilities
Acceso A Datos	edu.eafit.maestria.activa.dao.hibernate
UI	edu.eafit.maestria.activa.ui
Servicios - Seguimiento de Objetos	edu.eafit.maestria.activa.objectrtracking
Servicios - PCF	edu.eafit.maestria.activa.pcf edu.eafit.maestria.activa.pcf.ui
Servicios - De Entidad	edu.eafit.maestria.activa.services
Servicios - TVA	edu.eafit.maestria.activa.tva
Componente - Contenedor de IoC	edu.eafit.maestria.activa.container

4.3.2 Interfaz Gráfica

Figura 17 Interfaz Gráfica de la Herramienta de Creación

En la Figura 17 puede verse la pantalla de la herramienta de creación, esta consta de 3 capas:

1 La capa de aplicación contiene toda la funcionalidad de la herramienta de creación, la cual esta dividida en 4 partes:

   a Barra de herramientas. Allí se encuentra el acceso a las funcionalidades relacionadas con la administración de un proyecto, a los algoritmos para el seguimiento de objetos, y a la generación o transformación de la aplicación.

b Barra de dibujo. en esta barra se puede seleccionar la forma del dibujo para la selección de un objeto en el video, las formas implementadas fueron rectángulo y polígono. También se encuentra la opción para seleccionar el color que tendrá el área seleccionada.

c Controles de reproducción. Con estos botones se podrá reproducir, parar, pausar, adelantar y devolver el video rápidamente y cuadro a cuadro y así facilitar la selección de los objetos.

d Formulario de metadatos. A través de este formulario se ingresa la información del objeto seleccionado como es el nombre, descripción, el tipo de objeto, y las propiedades o metadatos del objeto.

2 La capa del reproductor de video.

3 La capa de gráficos contiene una ventana invisible que se sobrepone al video y que sirve de lienzo para dibujar sobre esta los objetos.

4.3.3 Seguimiento de Objetos

Para realizar el seguimiento de objetos se implementaron y ajustaron los algoritmos de *Template Matching* y de SURF (*Speeded Up Robust Features*) usando las librerías JavaCV y OpenCV.

4.3.3.1 *Template Matching*. Es una técnica para el procesamiento de imágenes digitales que cuantifica la similitud entre dos imágenes o parte de estas, con el fin de establecer si son la misma o no [62]. Para este proyecto se ha adaptado para trabajar con video.

La técnica consiste en seleccionar una imagen inicial (plantilla), ver Figura 18, que luego se desplaza desde la esquina superior izquierda en búsqueda de los objetos que se asemejen a la plantilla hasta encontrar el mayor número de coincidencias del objeto seleccionado.

Figura 18 Plantilla a buscar

Para reducir el tiempo de procesamiento del algoritmo se usaron tres técnicas:

1 Se transformaron las imágenes a blanco y negro para homogeneizar las imágenes y reducir el dominio del color.
2 Se escala cada imagen, plantilla y cuadro, al 50% de su tamaño original y
3 Se busca el objeto solamente sobre una región de interés (ROI) de la imagen donde se supone está el objeto. Corresponde al cuadro rojo en la Figura 19.

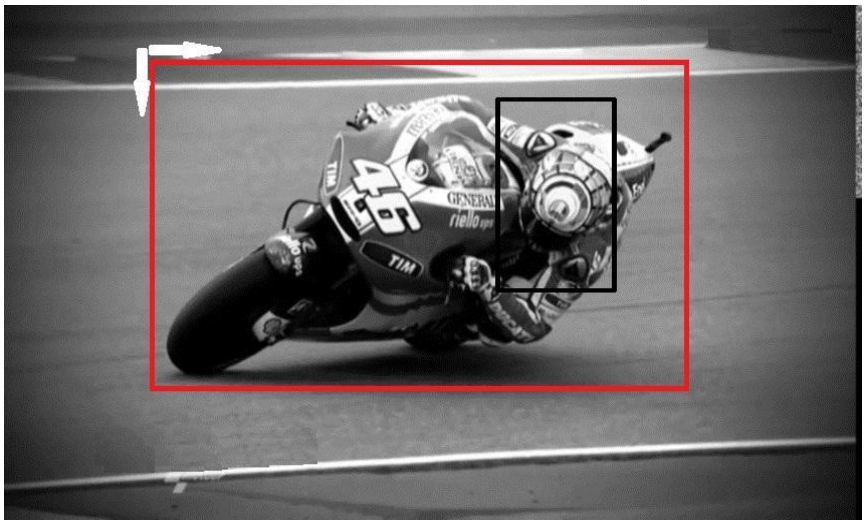

Figura 19 Cuadro de video con la Región de Interés (ROI) y la coincidencia encontrada.

Para tratar de encontrar mejores coincidencias a medida que se iban pasando cuadros de video, se hizo una variación al algoritmo que consistía en que a la plantilla original se le iba mezclando con las imágenes coincidentes (en este caso corresponde al cuadro negro sobre la Figura 19) y así ir adaptando la imagen a los cambios sucesivos.

El código fuente de este algoritmo se encuentra en el Anexo C.1.

4.3.3.2 SURF (*Speeded Up Robust Features*). Esta técnica consiste en replicar una imagen y buscar puntos que coincidan en todas las réplicas. Esta técnica es llamada multi-resolución [63]. La forma de obtener las réplicas es reduciendo el ancho de banda de la imagen original cada vez más obteniendo imágenes cada vez más borrosas, esta técnica se llama *Scale–Space*.

Luego de replicar la imagen se procede a seleccionar los puntos característicos y asignarles un descriptor para hacerlos únicos, para luego establecer la correspondencia entre dos imágenes buscando los puntos característicos de una imagen y en la otra y así para cada réplica, al final se comparan los descriptores resultantes de cada imagen en los cuales se refleja la información del entorno del punto característico, la información del descriptor sin la orientación del punto e información de la vecindad.

Entre más resolución tenga una imagen la comparación entre los descriptores encontrará más pares de puntos similares.

En este algoritmo también se aplicaron las técnicas usadas en *Template Matching* para mejorar la velocidad de procesamiento.

El código fuente de este algoritmo se encuentra en el Anexo C.2.

4.4 APLICACIÓN RESULTANTE

Como aporte de este proyecto de investigación para complementar la unión entre los estándares TV-Anytime y MPEG-7, se hizo una modificación a los esquemas XML de TV-Anytime. La modificación comprendió la extensión de las definiciones del tipo de descripción del programa (`tva:ProgramDescriptionType`) y del tipo de información de segmentación (`tva:SegmentInformationType`) para incluir los metadatos de las entidades y la información espacio-temporal de los objetos respectivamente. La extensión se debió a que la información de segmentación o de divisiones de un video en TV-

Anytime esta hecha en unidades de tiempo, como tomas, escenas, capítulos y/o temporadas, pero no incluía información espacial, es decir, sobre áreas o regiones presentes en un momento dado en un video.

La modificación principal fue la inclusión de los elementos de MPEG-7 responsables de definir una región y la creación de elementos nuevos en TVA que permitieran vincular los objetos del video con los nuevos elementos de MPEG-7. En el Anexo B.1 se muestra el esquema extendido de TV-Anytime y en el Anexo B.2 se lista el esquema extendido de MPEG-7 para TV-Anytime.

A continuación se describen los archivos TVA y PCF que representan una aplicación diseñada en la herramienta de creación. Solo se hace referencia a las partes usadas de los estándares, los ejemplos no son exhaustivos en cuanto a todo el contenido que deben tener.

4.4.1 Archivo TV-Anytime

La primera parte del archivo corresponde al encabezado, como se ve en el siguiente extracto de XML incluye, a demás de los espacios de nombres de TVA, los dos esquemas creados `activa` y `activampeg7` para poder hacer uso de los nuevos elementos adicionados.

```
1. <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2. <tva:TVAMain xmlns:tva2="urn:tva:metadata:extended:2011"
3.   xmlns:tva="urn:tva:metadata:2011"
4.   xmlns:mpeg7="urn:tva:mpeg7:2008"
5.   xmlns:interstitial="urn:tva:metadata:interstitial:2011"
6.   xmlns:rmpi="urn:tva:rmpi:2011" xmlns:mpeg21="urn:tva:mpeg21:2011"
7.   xmlns:activa="http://maestria.eafit.edu/activa/metadata"
8.   xmlns:activampeg7="http://maestria.eafit.edu/activa/metadata/mpeg7">
```

El formato TVA define que todos sus elementos son opcionales para así lograr un alto grado de interoperabilidad, sin embargo es recomendable por lo menos incluir la descripción del programa de televisión o video (`ProgramDescription`). En este caso se usa

la extensión creada, `ExtendedProgramDescriptionType`, para poder hacer uso de los nuevos elementos.

```
9.      <ProgramDescription xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
10.       xsi:type="activa:ExtendedProgramDescriptionType">
11.         <ProgramInformationTable/>
```

La descripción del programa contiene, entre otra información, las divisiones o segmentos del video, para lo que incluye el elemento `SegmentInformationTable` y su hijo `SegmentList`. En este último es donde se definen los tipos de segmentos del video, en este caso la primera división se llena con información espacio-temporal usando el tipo `ExtendedSegmentInformationType` cuyo atributo `segmentId` hace referencia a un cuadro de video (información temporal) y el elemento `EntityRegion` define mediante su contenido las coordenadas que delimitan la región (información espacial) asociada a un elemento. Pueden existir varios `EntityRegion` por cada cuadro o `SegmentInformation`. En el siguiente extracto de un archivo TVA se ven dos segmentos correspondientes a los cuadros 6568 y 6569, cada uno con una región.

```
12.         <SegmentInformationTable>
13.            <SegmentList>
14.              <SegmentInformation xsi:type="activa:ExtendedSegmentInformationType"
15.                 segmentId="6568">
16.                     <activa:EntityRegion entityRef="entity_634">
17.                         <activa:Region>
18.                            <activampeg7:Box activampeg7:dim="2 4">
19.                              303 162 380 162 303 232 380 232
20.                            </activampeg7:Box>
21.                         </activa:Region>
22.                     </activa:EntityRegion>
23.              </SegmentInformation>
24.              <SegmentInformation xsi:type="activa:ExtendedSegmentInformationType"
25.                 segmentId="6569">
26.                     <activa:EntityRegion entityRef="entity_634">
27.                         <activa:Region>
28.                            <activampeg7:Box activampeg7:dim="2 4">
29.                              303 162 380 162 303 232 380 232
30.                            </activampeg7:Box>
31.                         </activa:Region>
32.                     </activa:EntityRegion>
33.                 </SegmentInformation>
34.             </SegmentList>
35.         </SegmentInformationTable>
```

En la última parte de la descripción del video se incluye la información de las entidades relacionadas con las regiones anteriormente descritas. Para esto se crea una lista de elementos `Entity` dentro del elemento `EntityInformationTable`.

```
36.         <activa:EntityInformationTable>
37.             <activa:Entity entityId="entity_634" type="tipo 3">
38.                 <activa:Name>Casco de Casey Stoner</activa:Name>
39.                 <activa:Description> Se trata del casco desarrollado por Jorge Lor
40.                   enzo y Casey Stoner en el campeonato del Mundo de MotoGP.
41.                   Las 3 tallas de calota ultra
42.                   resistente, hacen que se adapte en peso y volumen al usuario...
43.                 </activa:Description>
44.                 <activa:Image>resource_633.jpg</activa:Image>
45.                 <activa:Properties>
46.                     <activa:Property>
47.                         <activa:Key>Precio   </activa:Key>
48.                         <activa:Value>371</activa:Value>
49.                     </activa:Property>
50.                     <activa:Property>
51.                         <activa:Key>Referencia</activa:Key>
52.                         <activa:Value>Ref. M-01906610</activa:Value>
53.                     </activa:Property>
54.                     <activa:Property>
55.                         <activa:Key>URL</activa:Key>
56.                         <activa:Value></activa:Value>
57.                     </activa:Property>
58.                 </activa:Properties>
59.                 <activa:Resources>
60.                     <activa:Resource>
61.                         <activa:Name>
62.                           00000006146-entity3115341367355911196.jpg
63.                         </activa:Name>
64.                         <activa:HREF>resource_633.jpg</activa:HREF>
65.                     </activa:Resource>
66.                 </activa:Resources>
67.             </activa:Entity>
68.         </activa:EntityInformationTable>
69.     </ProgramDescription>
70. </tva:TVAMain>
```

La relación entre las dos secciones se da por el atributo `entityId` del elemento `Entity` y el atributo `entityRef` del elemento `EntityRegion`, por ejemplo, la información relacionada con la siguiente región:

```
26.                     <activa:EntityRegion entityRef="entity_634">
```

esta contenida en:

```
37.             <activa:Entity entityId="entity_634" type="tipo 3">
```

4.4.2 Archivo PCF

Este archivo contiene la descripción de la aplicación. Esta divido en cuatro partes. En la primera se guarda información general como el nombre de la aplicación o `Service` como es llamado en PCF, el tamaño del video (`Size`), número de cuadros por segundo y la ubicación del video para la cual se creo la aplicación, entre otros.

```
1.  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2.  <pcf:PCF xmlns:pcf="http://www.dvb.org/pcf/pcf"
3.    xmlns:pcftypes="http://www.dvb.org/pcf/pcf-types"
4.    xmlns:xdvbpcf="http://www.dvb.org/pcf/x-dvb-pcf">
5.    <pcf:Service name="AplicacionPrueba">
6.      <pcf:String value="1.0" name="pcfSpecVersion"/>
7.      <pcf:Size value="720 576" name="referenceScreen"/>
8.      <pcf:String value="display-anamorphic" name="referenceScreenMapping"/>
9.      <pcf:Integer value="25" name="fps"/>
10.     <pcf:Stream name="video">
11.       <pcf:ExternalBody content-type="application/octet-stream"
12.         uri="Best_of_2011_Slow_Motion.avi"/>
13.       <pcf:Video name="default_fullscreen_video"/>
14.       <pcf:Audio name="default_audio"/>
15.     </pcf:Stream>
16.     <pcf:URI value="#main" name="firstScene"/>
```

La segunda parte corresponde a un arreglo creado con el elemento `Collection` que permite agrupar otros elementos bajo una misma estructura. Cada uno de los elementos es en si mismo otro arreglo, uno para cada cuadro de video, identificado por un atributo `name` que contiene el número del cuadro al cual hace referencia y dentro de este hay otro contenedor para referenciar una figura o región y su posición. En el siguiente código se puede distinguir un rectángulo que será posicionado en las coordenadas (48,91) para el cuadro 6568 y en (48,90) para el cuadro 6569, quiere decir que el rectángulo se desplaza un pixel en sentido vertical.

La información específica de cada región se presenta en la tercera parte del archivo y es referenciada mediante el atributo `href` que contiene el nombre de la región.

```
17.     <pcf:Collection>
18.       <pcf:Collection name="timestamp_6568">
19.         <pcf:Integer value="6568" name="frames"/>
20.         <pcf:Collection>
21.           <pcf:Position value="48 91" name="origin"/>
22.           <pcf:Rectangle href="#../../../animation_1350779725819"/>
23.         </pcf:Collection>
24.       </pcf:Collection>
25.       <pcf:Collection name="timestamp_6569">
26.         <pcf:Integer value="6569" name="frames"/>
27.         <pcf:Collection>
28.           <pcf:Position value="48 90" name="origin"/>
29.           <pcf:Rectangle href="#../../../animation_1350779725819"/>
30.         </pcf:Collection>
31.       </pcf:Collection>
32.     </pcf:Collection>
```

La tercera parte contiene el detalle de cada una de las regiones que se especificaron en la sección anterior. Cada región es identificada con un atributo `name` y contiene el color que la representa, el evento asociado que se deberá usar cuando se seleccione y la identificación del elemento en el archivo TVA con el que se relaciona.

```
33.     <pcf:Rectangle name="animation_1350779725819">
34.       <pcf:Position context="original" href="#../origin"/>
35.       <pcf:Color value="#0000ffff" name="bordercolor"/>
36.       <pcf:OnEvent name="select">
37.         <pcf:Trigger eventtype="KeyEvent">
38.           <pcf:UserKey value="VK_COLORED_KEY_3" name="key"/>
39.         </pcf:Trigger>
40.         <pcf:ActionLanguage name="show">
41.             entitySelected.content = entity_634.content;
42.             SceneNavigate(
43.              <uri>AplicacionPruebaentityDetailTemplate.pcf.xml#entityDetail
44.              </uri>,<enum>forget</enum>,nil);
45.           </pcf:ActionLanguage>
46.       </pcf:OnEvent>
47.       <pcf:Size value="77 70" name="size"/>
48.     </pcf:Rectangle>
```

En la última parte del PCF se hace referencia a la información de las entidades contenida en el archivo TVA. Cada región creada en la parte anterior del archivo PCF se relaciona con una entidad en el archivo TVA mediante el uso del atributo `name` del elemento `MarkedUpText` y del atributo `uri` del elemento `ExternalBody`. Así se obtiene la ruta completa para obtener los metadatos asociados a la región.

```
49.     <pcf:MarkedUpText name="entity_634">
50.       <pcf:ExternalBody content-type="text/xml" uri="tvaAplicacionPrueba.xml"/>
51.     </pcf:MarkedUpText>
52.     <pcf:Scene name="main">
53.       <pcf:URI value="#../entityDetail" name="next_scene"/>
54.       <pcf:BooleanVar name="running">
55.         <pcf:Boolean value="true" name="value"/>
56.       </pcf:BooleanVar>
57.     </pcf:Scene>
58.   </pcf:Service>
59. </pcf:PCF>
```

Aunque la información del PCF y del TVA puede resultar redundante, su duplicación se debe a que en los dos archivos significa cosas diferentes, en TVA es la representación de la información de un región de interés y los metadatos asociados a ésta y en PCF representa una zona de interacción. Esta duplicación permite que la información de TVA siga estando contenida en una única fuente y así facilitar la comunicación con entidades externas.

4.5 TRANSCODIFICADOR

4.5.1 Herramientas Utilizadas

- Visual Studio 2010: Ambiente de desarrollo que incluye el *framework* .Net 4.0 de Microsoft.
- AStyled.dll: Librería que permite darle estilo al código generado de Java, está incluido en la aplicación.
- LINQ (*Language Integrated Query*): Es un componente del *framework* .Net 4.0 de Microsoft que agrega una sintaxis especial para ejecutar consultas semejantes a las de SQL, define operadores de consulta estándar que permite filtrar, enumerar y crear proyecciones de varios tipos de colecciones usando la misma sintaxis, tales colecciones pueden ser vectores, arreglos, XML, bases de datos relacionales y orígenes de datos de terceros.
- *Internet Information Server*
- Apache 2.2

4.5.2 Parser

La implementación del módulo *Parser* se hace usando el componente LINQ. Este es usado para analizar los archivos PCF y TVA. Se usa para obtener la información de los entidades y sus respectivas ubicaciones en el espacio y en el tiempo, además de los metadatos asociados a éstas y dejarlas disponibles en estructuras de datos que puedan ser usadas por las diferentes plataformas de ITV.

4.5.3 Plataformas de ITV

Antes de incluir una plataforma de ITV en el Transcodificador primero se implementó una aplicación de prueba, por fuera de la arquitectura, que sirvió para probar el concepto de la interfaz con los elementos del video. Dicha aplicación serviría para crear eventualmente las plantillas respectivas para esa plataforma.

Una vez definida la aplicación se crearon las clases generadoras, estas clases toman la información analizada y la estructuran de acuerdo a las necesidades específicas de cada plataforma. Así por ejemplo la clase generadora de MHP crea objetos tipo java.awt.Rectangle mientras que la clase que genera HbbTV crea etiquetas div.

4.5.3.1 Google TV (HTML 5)

Ambiente de desarrollo

- Google TV STB. Sony Internet TV 3d Blu-ray player. Modelo: NSZ-GT1

Clase: Transcoder.Controllers.HTML5Controller

La aplicación que se hizo para Google TV es también compatible con cualquier navegador que soporte HTML5.

En esta solución se aprovechó el potencial de las etiquetas video y *canvas* de HTML5. La etiqueta video agrega al navegador una forma de reproducir video nativamente sin recurrir a objetos embebidos, tiene una restricción y es que soporta una pequeña cantidad de formatos de video. Y la etiqueta canvas es un elemento que soporta la generación de gráficos dinámicamente usando JavaScript.

La etiqueta *canvas* es superpuesta, usando posicionamiento absoluto, sobre el reproductor de video en una página web. En esta página, adicionalmente, se crean, aunque ocultas, todas las representaciones de los objetos que se pueden mostrar. Estas representaciones se crean usando etiquetas normales de HTML (*table*, *div*, entre otras).

En un archivo de JavaScript se encuentra asociada la información de las figuras geométricas que representan los objetos y sus respectivas posiciones. Cada cierto tiempo (dado por los cuadros por segundo del video de la aplicación) se ejecuta un método que extrae las estructuras almacenadas en el código JavaScript que se mostrarán. Estas estructuras son dibujadas luego por el objeto *canvas*.

4.5.3.2 MHP

Ambiente de desarrollo

- Set Top Box Telesystem 7900HD – *Digital Terrestrial Receiver*. Procesador 400MHz y 256MB RAM.
- Servidor de aplicaciones Open Caster 2.0 Avalpa

Clase: Transcoder.Controllers.MHPController

En esta plataforma no hay que hacer uso de ninguna etiqueta para desplegar el video debido a la naturaleza de la misma plataforma, donde el video es una capa que siempre esta presente.

La solución incluye una clase que convierte los datos analizados en el *parser* a componentes gráficos de MHP (HComponent). Estos componentes usan las librerías de gráficas nativas de Java haciendo uso de los objetos java.awt.Rectangle y java.awt.Polygon para dibujar las áreas seleccionadas.

4.5.3.3 HbbTV

Ambiente de desarrollo

- Televisor Philips compatible con DVB-T2
- Servidor de aplicaciones Open Caster 2.0 Avalpa

Clase: Transcoder.Controllers.HBBTVController

Esta solución es muy similar a la propuesta para Google TV, las dos diferencias principales son: primero que no se cuenta con una etiqueta video, por lo que se debe usar un objeto embebido que esta disponible en la API del televisor y segundo, que no existe un objeto con el que se pueda dibujar por lo que se recurre a crear rectángulos usando etiquetas div y animar dichas etiquetas usando JavaScript.

4.5.3.4 Samsung TV

Ambientes de desarrollo

- Samsung TV Apps Editor versión 3.1.1 para Televisores versión 2012
- Samsung TV Apps Editor versión 2.5.1 para Televisores versión 2010-2011
- Smart TV versión 2011

A pesar de los intentos por desarrollar una plantilla para esta plataforma no fue posible por múltiples razones:

- En la versión 3.1.1 del ambiente de desarrollo no es posible obtener el tiempo transcurrido en un video en milisegundos debido a que el reproductor no es compatible con la especificación de la etiqueta video de HTML5, esto implica un problema para desarrollar aplicaciones como las propuestas en este proyecto pues es necesaria una precisión del orden de milisegundos para poder sincronizar la aplicación y el video a la hora de señalar los objetos.
- Para solucionar este problema de sincronización, probamos lanzar un cronómetro a la par con la reproducción del vídeo pero a pesar de que era mucho más preciso, sufrimos de un pequeño retardo con el dibujado de las objetos.
- También para esta versión se probó el reproductor incluido por Samsung en su API pero no fue posible cargar correctamente varios videos de pruebas. Lo más común era que se reproducía el sonido pero no se podía ver la imagen.
- Otro problema fue la visibilidad de los objetos en HTML5 usando el objeto *canvas*, en la versión 2.5.1 con el mismo código fuente, era posible dibujar los objetos sin este inconveniente. Para resolver esto, en lugar de dibujar usando HTML5,

probamos a dibujar los objetos como instancias de la etiqueta *div*; después de lograr dibujar sobre el video, al probarlo en el televisor nos dimos cuenta de que había un problema de rendimiento, dichos etiquetas se dibujan en tiempos muy diferentes a los especificados, y la precisión es crítica para este tipo de aplicaciones.

- Luego, haciendo más pruebas en el televisor, observamos que el *layout* del video no se podía controlar, no se permite poner márgenes, por lo cual la relación del aspecto del video nos dejaba un problema con las posiciones de los recuadros que íbamos a dibujar.

4.5.4 Plantillas

Las plantillas pueden desplegar metadatos del objeto seleccionado, desplazar al televidente a otras partes del video o de otros videos, realizar búsquedas y/o compras, entre otros ejemplos. Para esta implementación las plantillas describen aplicaciones que muestran las entidades de la misma forma en que fueron dibujadas en la Herramienta de Creación, así si sobre un objeto se dibujó un rectángulo de color rojo en la aplicación final el televidente verá una figura similar y los metadatos recolectados serán mostrados en un área específica de la pantalla, que puede o no sobreponerse al video.

La Figura 20 muestra el listado de plantillas para cada una de las plataformas implementadas.

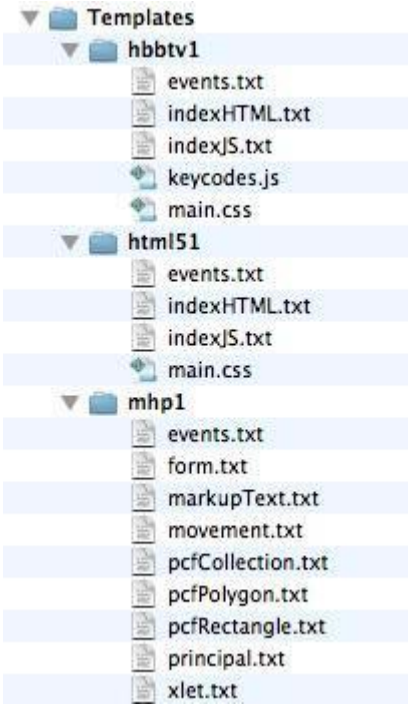

Figura 20 Estructura y listado de plantillas para cada plataforma

Las siguientes propiedades quedan disponibles para ser utilizadas en las plantillas luego de analizar los archivos PCF y TVA. Para incluirlas en una plantilla nueva deben escribirse con la estructura *<propiedad>*:

SCENENAME: Nombre del proyecto como fue concebido en la Herramienta de Autor
SIZE.WIDTH: Alto del video en píxeles
SIZE.HEIGHT: Ancho del video en píxeles
URLVIDEOPATH: URL donde se encuentra el video
FRAMESPERSECOND: Número de cuadros por segundo
GETVALUESHAPE: Arreglo de las posiciones de las figuras para dibujar
SHAPEJS: Arreglo en JavaScript de las figuras para dibujar
HREFS: Arreglo de recursos asociados a las entidades

4.5.5 Interfaz TranscoderWebService

La siguiente tabla contiene la modificación realizada a Interfaz TranscoderWebService definida en la sección N del capítulo de la arquitectura

Constantes	En el parámetro de entrada *platform* los valores disponibles en la primera versión implementada de la arquitectura serán: GoogleTV = 1; HBBTV = 2; MHP = 3; En el parámetro de entrada *template* el valor disponible en la primera versión implementada de la arquitectura será 1;

CAPÍTULO 5 EVALUACIÓN DE LA ARQUITECTURA

5.1 INTRODUCCIÓN

Para la evaluación de la arquitectura se llevaron a cabo dos pruebas, una temprana correspondiente a la evaluación empírica basada en la experiencia y las observaciones de un experto[49]. Evaluación que sirvió para refinar el diseño y cuyo resultado se ve materializado a lo largo del Capítulo Documento de la Arquitectura. Y una segunda evaluación tardía que se detallará en el presente capítulo basada en prototipos donde cada uno de los escenarios de calidad fueron puestos a prueba.

La forma de evaluación por prototipos se escogió por:

- Al tratarse éste de un proyecto con fines académicos y realizarse en el laboratorio de DTV, crear un prototipo y evaluar la arquitectura mediante esta técnica resulta conveniente ya que este sirve como punto de partida para una implementación completa.
- Además este tipo de evaluación posibilita aprender sobre el dominio del problema según Bardram [69], en este caso permite la experimentación con las diferentes plataformas de ITV presentes en el laboratorio.
- Y porque debido a los atributos de calidad que se desean evaluar para este diseño en particular resulta significativo realizar un ejercicio práctico.

El primer paso para realizar cualquier evaluación es conocer qué es lo que se quiere evaluar. En este caso lo que interesa evaluar es el cumplimiento de los escenarios de

[49] Especialista Lenin David Lozano. Ingeniero de Sistemas de la Universidad de Antioquia y Especialista en Desarrollo de Software de la Universidad EAFIT. Actualmente consultor en Arquitectura de Aplicaciones. ATAM Evaluator Certified.
http://www.sei.cmu.edu/training/certificates/holders/.

calidad sobre rendimiento, extensibilidad y usabilidad propuestos en la sección de *drivers* arquitectónicos del capítulo sobre el diseño de la arquitectura.

5.2 EXPERIMENTO

Para evaluar los atributos de rendimiento y usabilidad se elaboró un experimento a partir de la definición de un caso de uso sobre una aplicación en la que se debían señalar y enriquecer tres objetos de un video para una plataforma web, usando HTML y JavaScript.

El experimento estuvo dividido en dos partes, la primera era estimar el tiempo de desarrollo bajo un ambiente sin la arquitectura y la segunda consistía en desarrollar la misma aplicación haciendo uso de la arquitectura.

Antes de que los participantes se enfrentarán a la segunda parte del experimento fueron capacitados para diseñar una aplicación usando la arquitectura. Para este desarrollo se definieron 11 tareas. Para cada una se midió el tiempo que tardaba el usuario en completarla (sin contar el tiempo que tardaba el sistema en completar sus funciones internas) y el número de equivocaciones que tuvo.

Las 11 tareas fueron:

- Dibujar o seleccionar 3 objetos
- Enriquecer las 3 selecciones anteriores
- Ejecutar para las mismas 3 selecciones alguno de los algoritmos de seguimiento
- Generar la aplicación
- Probar la aplicación

Luego de participar en el experimento, las personas diligenciaron un cuestionario en el que se recogió información general además de la información de las dos partes del experimento. En la primera parte el participante debía ingresar el cálculo de su estimación en horas y en la segunda parte debía responder una serie de preguntas cualitativas sobre la arquitectura.

El cuestionario estaba dividido en 4 partes:

- Parte 1. Información General
- Parte 2. Especificación de la aplicación a desarrollar: Contenía la especificación del caso de uso.
- Parte 3. Desarrollo de una aplicación interactiva para video en Internet usando métodos tradicionales: Preguntas sobre los tiempos estimados.
- Parte 4. Desarrollo de una aplicación interactiva para video en Internet usando acTiVa: Preguntas para obtener una calificación subjetiva de la arquitectura.

El caso de uso y el cuestionario usados se encuentran en el ANEXO D. Cuestionario para la evaluación de la Arquitectura.

Las pruebas se realizaron en un MacBook Pro con procesador 2.6GHz Intel Core 2 Duo y 4GB de RAM tipo SDRAM a 667MHz. con Java 1.6 instalado para ejecutar la herramienta de creación. El Transcodificador se ejecutó sobre un Windows 7 Ultimate ejecutándose sobre una máquina virtual VMWare Fusion 4.1.1 en el mismo equipo. Además hay dos servidores web Apache instalados, uno en cada sistema operativo.

El análisis estadístico de los datos arrojados por el experimento fueron realizados con la herramienta SPSS Statistics Desktop versión 19.0.0 y Statgraphics.

5.2.1 Sobre la muestra

En el experimento participaron seis personas. Debido a circunstancias contextuales la medición de más individuos se vio limitada. Todos los participantes tenían experiencia como desarrolladores de software. Tres de ellos con amplia experiencia en desarrollo de aplicaciones relacionadas con video.

Si nos apegamos a una estadística formal para calificar el tamaño de muestra para esta evaluación se podría decir que no es el apropiado. También se podría decir, que los análisis y resultados estadísticos obtenidos con los datos recogidos, especialmente para el atributo de usabilidad, son preliminares y no son 100% confiables debido al tamaño de muestra tan pequeño.

Sin embargo, para las mediciones de usabilidad algunos autores, aunque reconocen de la importancia de un tamaño de muestra mayor, sugieren que con una muestra de entre 6 y 12 participantes se pueden encontrar alrededor del 80% de problemas de usabilidad [8], un número que brinda balance entre la confiabilidad y los costos de realizar este tipo de pruebas, es una consideración importante sabiendo que se trata de un prototipo y no de una versión final en este caso.

5.3 PRUEBA DE USABILIDAD

La usabilidad se refiere a la medida con que un usuario puede hacer uso de una herramienta de una manera fácil, intuitiva y cómoda, de tal forma que pueda lograr los objetivos específicos de la herramienta con efectividad, eficiencia y satisfacción en un contexto de uso específico [8][14].

En esta prueba el contexto de uso esta dado por el experimento descrito previamente. Donde se describen el usuario o participante, el objetivo (que es la aplicación a desarrollar), las tareas a realizar, el entorno de trabajo y los equipos a usar (representados en los componentes de la arquitectura).

De acuerdo a la definición de usabilidad, esta se mide comúnmente por las métricas [8][51]:

- Efectividad
- Eficiencia
- Satisfacción

5.3.1 Efectividad

La efectividad se refiere a la precisión con la que un usuario logra los objetivos, generalmente se mide por la tasa de tareas completadas con éxito y fracaso [8].

Para esta medida se definió en el escenario de usabilidad que:

El usuario no debe equivocarse en más del 10% de las tareas.

Para este experimento el 10% de las tareas corresponde al 10% de 1, es decir, el usuario no puede equivocarse en más de 1 tarea.

Para evaluar la efectividad de la arquitectura se observó de cerca la realización del experimento para contabilizar las veces que los usuarios solicitaron ayuda para realizar las tareas o que sin preguntar no usaron las herramientas adecuadas.

Tabla 7 Errores en el uso de la herramienta

Participante	Error-Dibujo	Error-Enriquecer	Error- seguimiento
1	1	0	1
2	3	1	0
3	1	0	0
4	0	0	1
5	3	0	0
6	1	3	0

En la tabla anterior se puede observar que los participantes 2 y 5 obtuvieron el mayor puntaje de error relacionado al dibujo lo cual indica que no comprendieron la actividad. Así mismo, el individuo 6 tampoco comprendió la tarea de enriquecimiento. Estos valores (Color amarillo) no fueron tenidos en cuenta porque fueron errores por no comprender la actividad y no errores asociados al desarrollo de la actividad misma.

El número total de errores es 6, y el umbral propuesto para este caso también es 6, que equivale al 10% del número total de tareas que es 66.

Total de tareas = número de participantes * por el número de tareas

= 6 * 11

= 66

Máximo número de tareas permito = 10% de 66

= 0.1 * 66

= 6.6

Tasa de error resultante <= Tasa de error esperada

1 error <= 1 error

5.3.2 Eficiencia

Por eficiencia se entiende como la rapidez y facilidad con la que los usuarios de un sistema hacen uso de él [14], generalmente se mide por el tiempo que le toma a un usuario realizar una tarea [8].

Para esta medida se definió en el escenario de usabilidad que:

El tiempo de selección de un objeto y la selección de un algoritmo de seguimiento no debe superar 1 minuto.

Para evaluar la eficiencia cada una de las tareas propuestas en el experimento fue cronometrada para cada participante, (ver el ANEXO F para ver la tabulación completa de los datos). La siguiente tabla agrupa el tiempo en segundos que tomó cada usuario en realizar las tareas de dibujo y seguimiento. En total correspondían a 36 tareas de las 66 planteadas.

Tabla 8 Tiempo Promedio de las tareas de dibujo y seguimiento

Participante	Dibujo + Seguimiento 1	Dibujo + Seguimiento 2	Dibujo + Seguimiento 3	Promedio Participante
1	29	28	8	21.67
2	88	48	28	54.67
3	34	29	78	47.00
4	29	46	40	38.33
5	12	59	19	30.00
6	35	20	15	23.33
			Promedio Total	35.83

Como se observa en la tabla el promedio total para la agrupación de la tareas dibujo y seguimiento es de casi 36 segundos. Mejorando el tiempo propuesto para la eficiencia propuesta.

Tiempo resultante <= Tiempo esperado

35.83 Seg <= 60 Seg

5.3.3 Satisfacción

Por último, la satisfacción hace referencia a la sensación de agrado que tiene un usuario por el uso de la herramienta, generalmente es una medida subjetiva que se obtiene de preguntar específicamente por las impresiones y los juicios de valor que tienen los usuarios sobre la herramienta.

Para esta medida se definió en el escenario de usabilidad que:

El grado de satisfacción de los usuarios debe superar el 80%.

Después de una revisión detallada de la cuarta parte del cuestionario se tomaron los siguientes cuestionamientos como los más relevantes :

- En general, estoy satisfecho con la facilidad para completar las tareas
- En general, estoy satisfecho con el tiempo que me tomo completar las tareas
- Me siento satisfecho con la aplicación resultante de usar acTiVa

Para cada una de las preguntas de esta sección se les pidió a los usuarios seleccionar qué tan de acuerdo estaban con en el enunciado, siendo el valor 1 completamente en desacuerdo y 5 completamente de acuerdo. Luego se realizó una unión de estas respuestas para formar la variable llamada “satisfacción” (correspondiente a la 5ta columna de la siguiente tabla) debido a que estos enunciados tratan de recoger información sobre dicha propiedad. La siguiente tabla muestra el resultado de este proceso.

Tabla 9 Nivel de Satisfacción

Participante	Satisfacción Facilidad Tareas	Satisfacción Tiempo Tareas	Satisfacción Resultado	Satisfacción
1	5	5	4	4
2	5	5	5	5
3	5	5	4	4
4	4	5	5	4
5	5	5	5	5
6	5	5	5	5

Por tratarse en este caso de variables categóricas no se puede sacar un promedio para asignarlo como valor de la variable satisfacción, lo que debe hacerse es recodificar el grupo de variables de acuerdo a las necesidades. Para este análisis se recodificaron las variables tomando el menor valor para cada participante, y a si darle visibilidad a las falencias.

De la tabla anterior se obtuvo que el 50% de los participantes tienen un nivel de satisfacción de 4 y el otro 50% restante de 5.

Para poder hacer una comparación entre la medida esperada y la variable satisfacción en los mismos términos, el porcentaje dado como medida se transformó a la escala de satisfacción usando la siguiente tabla.

Tabla 10 Equivalencias entre la escala de cinco niveles y el porcentaje de satisfacción. (Adaptada de la tabla 6.33 de [8])

Porcentaje (%)	0	25	50	75	100
Equivalencia en la escala de 1 a 5	1	2	3	4	5

De esto se obtiene que un porcentaje de satisfacción del 80% corresponde al nivel 4 en la escala usada en el cuestionario.

Por lo tanto podemos concluir que la medida de satisfacción fue alcanzada. Debido a que la mitad de los participantes tienen un grado de satisfacción igual al esperado y la otra mitad mayor a la esperaba.

Satisfacción Resultante > Satisfacción Esperada

50% Nivel 4, 50% Nivel 5 > Nivel 4

5.3.4 Opiniones

El siguiente es un compendio de los comentarios más significativos escritos por los participantes en el cuestionario. Estas opiniones también sirven como un criterio de evaluación de usabilidad, aunque no se puedan medir fácilmente expresan las propiedades o características del sistema susceptibles de mejorar.

- *“La selección de objetos es sencilla e intuitiva... me reduce completamente el tiempo de construcción... al contrario de cuando uno está programando que para ver como va la cosa toca dar muchas vueltas para poder ensayar”*
- *“Se debe mejorar el seguimiento de objetos, a veces es muy corto”*
- *“No esta mal pero podría ser mejor el ingreso de metadatos.”*
- *“Incómodo que no se puedan usar los comandos de copiado y pegado, cancelar o deshacer”*
- *“...lo mejor es la transformación a diferentes plataformas, ya que reduce el tiempo de migración...”*

5.4 PRUEBA DE RENDIMIENTO

El rendimiento está relacionado con el desempeño de un sistema, es decir con la velocidad de respuesta y precisión u otras restricciones temporales en las que el sistema lleva a cabo una funcionalidad específica [57].

El tiempo de procesamiento por cuadro de video no puede ser superior a 500 milisegundos.

En este caso la prueba debe superar la medida establecida en el escenario de rendimiento. La decisión de diseño que se definió para tratar de darle atención a este atributo fue aumentar la eficiencia computacional mediante el uso de librerías nativas, particularmente mediante el uso de OpenCV.

Durante la ejecución del experimento se recolectaron datos sobre el rendimiento de los algoritmos para el seguimiento de objetos. Cada vez que los participantes seleccionaron uno de los dos algoritmos, *Template Matching* o SURF, para realizar el seguimiento de los objetos se imprimió en un archivo de *logs* el tiempos que tomaba procesar cada cuadro de video exitosamente. No se midió la tasa de éxito y/o fracaso, es decir, el porcentaje de las

veces que el algoritmo debería encontrar el objeto y no lo hizo, porque no se trataba de evaluar la validez del algoritmo sino la velocidad de procesamiento.

En total fueron tomados 830 datos para ambos algoritmos para un video con una dimensión de 854x480 pixeles, los objetos evaluados eran de 100x100 pixeles de aproximadamente.

A continuación se presenta la estadística descriptiva del tiempo de ejecución asociado a ambos algoritmos:

Tabla 11 Estadística descriptiva asociada a los algoritmos de seguimiento de objetos.

Medida	***Template Matching*** **Tiempo1**	**SURF** **Tiempo2**
Promedio	**16,1361**	**77,1518**
Mediana	14,0	64,0
Moda	10,0	36,0
Desviación Estándar	**13,9546**	**46,3094**
Coeficiente de Variación	**86,4803%**	**60,0237%**
Mínimo	7,0	8,0
Máximo	217,0	328,0

Esta tabla incluye medidas de tendencia central y medidas de variabilidad. De particular interés aquí son el promedio, la desviación y en especial el coeficiente de variación el cual hace referencia a la relación entre el tamaño de la media y la variabilidad de la variable (desviación estándar) [71].

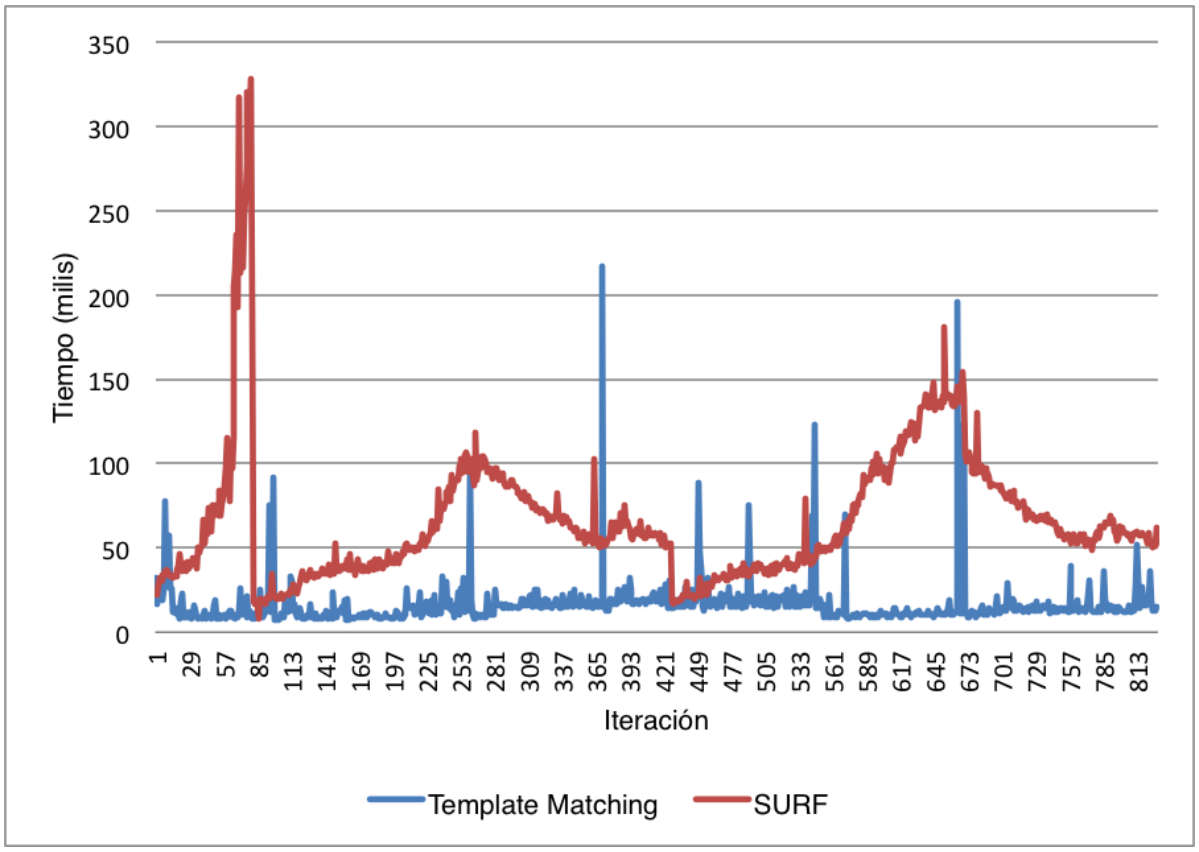

Figura 21 Gráfica de tiempos de *Template Matching* y SURF

Observaciones:

- Se puede concluir que los datos asociados a *Template Matching* tienen mayor variabilidad con respecto a la media que los asociados a SURF, ya que a más elevado el coeficiente de variación más dispersión o variabilidad tienen los datos. (86,48% > 60,023%).
- También se puede concluir de las medidas de tendencia central, en especial del promedio, que el algoritmo de *Template Matching* es más rápido que el algoritmo SURF. (16.13 < 77.15) Esto se visualiza más fácilmente en la gráfica de tiempos donde la línea que representa *Template Match*ing aparece por debajo de SURF la mayor parte del gráfico.
- En general se puede concluir que ambos algoritmos superan ampliamente el umbral propuesto de 500 milisegundos para el atributo de rendimiento propuesto.

- Podemos predecir que en un equipo con características de *hardware* más actualizadas el rendimiento debe ser mejor que este resultado superando aún más la medida propuesta de 500 milisegundos por cuadro de video.

(Template Matching y SURF) < Rendimiento Esperado
(16.13 y 77.15) milis < 500 milis

Para una mayor comprensión de los datos acá presentados se sugiere leer el ANEXO E. Categorización de variables.

5.5 PRUEBA DE EXTENSIBILIDAD

La extensibilidad se refiere a la facilidad con la que una parte del sistema puede ser extendida sin afectar otras partes del sistema. Es una propiedad que no es visible en tiempo de ejecución sino en tiempo de desarrollo.

Existen algunas métricas relacionados directamente con el código fuente que sirven para determinar el grado de extensibilidad. Una de estas propiedades es el acoplamiento, medida que tiene que ver con las dependencias entre diferentes componentes. A una menor medida de acoplamiento más independientes son los componentes entre sí y más fácil es realizar cambios [66].

Para esta medida se definió en el escenario de extensibilidad que:

El tiempo de adicionar la nueva funcionalidad (herramienta de dibujo, algoritmo de seguimiento de objetos) no debe tomar más de un día y no debe incluir el tiempo de desarrollo de la funcionalidad.

Las funcionalidades de las que se habla en el cuadro anterior, es decir, las herramientas de dibujo y los algoritmos de seguimiento de objetos, eran responsabilidad de la herramienta de creación. Esta herramienta, como definido desde el diseño, fue construida usando la plataforma RCP de Eclipse.

RCP usa un mecanismo de *plug-ins* que es en esencia un mecanismo de *callback,* que es comúnmente usado para alcanzar extensibilidad [64]. Un *plug-in* extiende funcionalidad implementando los "puntos de extensión" expuestos por otros *plug-ins* o define los suyos para que otros los extiendan. La configuración de estos puntos de extensión se hace en archivos XML en lugar del código lo que hace que el acoplamiento sea muy bajo en la mayoría de los casos.

A continuación se presenta la métrica de acoplamiento obtenida usando el *plug-in* CodePro AnalitiX 7.1.0 para Eclipse 3.7 [50] para los paquetes relacionados con las funcionalidades de dibujo y seguimiento de objetos en la herramienta de creación.

Tabla 12 Acoplamiento de la Herramienta de creación

Paquete	**Acoplamiento Aferente[51]**	**Acoplamiento Eferente[52]**
edu.eafit.maestria.activa.ui.handlers.track	0	3
edu.eafit.maestria.activa.objecttracking	4	1
edu.eafit.maestria.activa.objecttracking.services	2	4
edu.eafit.maestria.activa.objecttracking.utils	3	1
edu.eafit.maestria.activa.ui.player	13	7

De la experiencia con la implementación del prototipo y de los métricas expuestas se concluyo que: si se hizo una correcta utilización del mecanismo de *plug-ins* de Eclipse

[50] CodePro AnalitiX, https://developers.google.com/java-dev-tools/codepro/doc/. Consultado: 4 agosto de 2012.

[51] Acoplamiento Aferente es el número de tipos que dependen de clases del paquete analizado. Este es un buen indicador de cómo los cambios en este paquete podrían influenciar otras partes del sistema.

[52] Acoplamiento Eferente es el número de tipos de los cuales dependen clases del paquete analizado. Este valor indica que tan sensitivo es un paquete a los cambios en otros paquetes.

para cumplir con el atributo de extensibilidad ya que se pudieron adicionar los algoritmos de seguimiento en cuestión de horas y no de días, además de que solo se afectó un paquete adicional.

Tiempo que toma adicionar funcionalidad nueva	**<**	***Tiempo Esperado***
n horas	**<**	***1 día***

En el caso del Transcodificador el escenario planteaba la siguiente medida:

> ***El tiempo de adicionar la nueva plataforma no debe tomar más de una semana y no debe incluir el tiempo de desarrollo de la nueva plataforma.***

Para este componente los valores de acoplamiento fueron bastante altos. Debido principalmente a que traía ya vicios desde su construcción inicial, también a la falta de experiencia de las personas a cargo y a que C# no cuenta con un mecanismo como del RPC desde su concepción.

Esto se vio reflejado en el tiempo que tomó adicionar de cada una de las plataformas durante la implementación del prototipo. Este tiempo fue de alrededor de 2 semanas, solo para la parte de integración con el Transcodificador.

La siguiente tabla resume los datos de acoplamiento obtenidos usando la herramienta Visual Studio Code Metrics PowerTool 10.0[53]

[53] Visual Studio Code Metrics PowerTool. http://www.microsoft.com/en-us/download/details.aspx?id=9422. Consultado: 4 de Agosto de 2012.

Tabla 13 Acoplamiento del Transcodificador

Paquete	Acoplamiento
Transcoder.Controllers.HTML5Controller	35
Transcoder.Controllers.HbbTVController	34
Transcoder.Controllers.MHPController	48

Tiempo que toma adicionar una plataforma nueva	**<**	***Tiempo Esperado***
más de una semana	***no es <***	***1 semana***

5.6 TIEMPO DE DESARROLLO

Aunque se pretendía comparar estadísticamente el tiempo de desarrollo de una aplicación con y sin la arquitectura. Dicha comparación resulta innecesaria ya que con la sola observación del experimento y a las opiniones de los participantes se puede establecer que en efecto el tiempo de desarrollo disminuye.

Los datos suministrados por los participantes en la primera parte del experimento, correspondientes a estimaciones basadas en su experiencia, eran claramente mayores a las mediciones obtenidas durante el experimento con la arquitectura. Según los tiempos estimados, en promedio un desarrollador gastaría algo más de 70 horas entre implementación y pruebas, y de acuerdo a las mediciones realizadas durante el experimento con la arquitectura se obtuvo que en promedio un desarrollador se tardaría 31 minutos, siendo claramente menor. Dichos datos se pueden ver mejor la siguiente tabla.

Tabla 14 Tabulación de tiempos de desarrollo medidos y estimados

Participante	Tiempo Medido (min)	Análisis y Diseño	Implementación	Pruebas	Tiempo I+P (horas)	Otras Tareas	Migración
1	27	13.5	54	9	63	13.5	76.5
2	30	4.5	16	10	26		8
3	43	5	14	8	22		20
4	41	10	56	3	59		56
5	25	81	90	36	126	63	27
6	22	54	90	36	126	90	27
Promedio	**31.33**				**70.33**		

Aunque para ajustar la medida hay que tener en cuenta que también hubo un tiempo de desarrollo que se dedicó durante la implementación del prototipo a cada una de las plataformas y que sería muy similar a los tiempos estimados por los participantes para una implementación sin la arquitectura. La ventaja radica en que en la arquitectura, una vez incluida una plataforma de ITV, ésta queda disponible para futuros desarrollos, así el tiempo que tomó implementarla solo cuenta la primera vez y en las implementaciones futuras sólo se cuenta el tiempo que tarda en desarrollarse la aplicación. Adicionalmente, el tiempo de migración a una plataforma diferente de ITV es despreciable usando la arquitectura (si dicha plataforma ya está implementada), mientras que sin la arquitectura siempre habría que rescribir la aplicación resultando en más costos y tiempo.

5.7 CONCLUSIONES

Tras realizar la evaluación de la arquitectura usando prototipos se puede decir que está técnica ayuda a entender como funcionan las diferentes tácticas, estilos y patrones arquitectónicos al implementarlas en un caso real. También ayuda a reducir los riesgos de una mala decisión debido a su cercanía con la implementación.

Adicionalmente los prototipos resultan de utilidad para probar los atributos de calidad que son visibles en tiempo de ejecución, como lo son el rendimiento y la usabilidad.

La decisión de diseño de aumentar la eficiencia computacional mediante el uso de librerías nativas para darle cumplimiento al atributo de rendimiento fue correcta, la

velocidad de procesamiento superó con gran margen la medida propuesta en el escenario.

Aunque la arquitectura supera la prueba de usabilidad, queda claro que hay que trabajar aún más en este criterio, especialmente hay que involucrar desde el diseño arquitectónico las tareas que los usuarios encuentran con más naturalidad en cualquier herramienta, como borrar, cancelar, deshacer, copiar y pegar.

Otra conclusión importante acerca del uso de prototipos es que es recomendable entrenar mejor a los participantes de las pruebas no solo en lo que se pretende evaluar sino en la técnica, en este experimento algunos de ellos esperaban una herramienta completa, es decir, en una versión final y este no es el caso con los prototipos.

CAPÍTULO 6 CONCLUSIONES Y TRABAJO FUTURO

6.1 CONCLUSIONES

En esta tesis, se implementó una arquitectura que permite generar aplicaciones multiplataforma para televisión interactiva que usan los objetos del video como medio de interacción y para la cual se implementó un prototipo que genera aplicaciones para las plataformas: MHP, Google TV (HTML5) y HbbTV.

Para lograr esto, se recurrió al Diseño Basado en Atributos (ADD). Este método permitió concentrarse en los atributos más importantes que se debían cumplir, usabilidad, rendimiento y extensibilidad, y para los cuales se definieron los escenarios de calidad, y entorno a estos ir creando y satisfaciendo el resto de requerimientos tanto funcionales como no funcionales.

Se definieron dos componentes principales. Una herramienta de escritorio llamada Herramienta de Creación, encargada del diseño de las aplicaciones y un servicio web, llamado Transcodificador, encargado de la conversión a múltiples formatos.

La manipulación de los objetos contenidos en el video se da en la herramienta de creación y se hace por fuera del video, es decir, el video no es modificado en ningún momento, lo que se hace es guardar referencias de cuando y donde tienen presencia estos objetos. Para facilitar este trabajo se usaron algoritmos de seguimiento de objetos.

Los datos que se intercambian entre estos dos componentes contienen la información de los objetos de los videos, sus metadatos y el código de las aplicaciones. Esto fue representado usando los estándares TV-Anytime y PCF.

TV-Anytime es usado como medio para transportar los metadatos de un video asociados a una aplicación. En él también se almacena información espacio-temporal de los objetos. Para esto fue necesario extender la definición de algunas funcionalidades del estándar al crear un conjunto nuevo de etiquetas para la marcación de regiones basadas en MPEG-7. Al ser TV-Anytime una especificación altamente usada brinda cierto nivel de interoperabilidad entre las aplicaciones diseñadas con posibles aplicaciones y plataformas con las que se requiera intercambiar información.

PCF es el lenguaje que describe las aplicaciones, se usa para definir la representación del cómo se deberían ver los objetos y en que momento. Este estándar se seleccionó principalmente porque es independiente de cualquier plataforma brindando un alto grado de portabilidad.

Para valorar la viabilidad y efectividad de la arquitectura, está fue sometida a una evaluación usando prototipos. La evaluación mostró que la arquitectura propuesta cumplía con los escenarios de calidad planteados.

La evaluación por prototipos resultó una técnica adecuada. Para arquitectos en formación es de gran ayuda ya que permite aprender sobre nuevos patrones y estilos de una forma práctica llevándolos a una implementación concreta.

Para terminar, vale la pena resaltar las dificultades que se tuvieron en la realización de este proyecto, la principal fue la de adaptar componentes ya desarrollados y sobre los cuales no se tenía mucho conocimiento, caso particular el del Transcodificador, limitando en gran medida el diseño de la arquitectura.

Otro dificultad que vale la pena resaltar se presentó durante la evaluación de la arquitectura, especialmente con la consecución de los participantes, debido a que no existe una población considerable de desarrolladores en esta área, de la cual se pudiera tomar una muestra idónea. La mayoría de estos desarrolladores pertenece a grupos aislados en empresas y universidades y no hay una interacción que permita aunar esfuerzos y conocimientos.

6.1.1 Recomendaciones

Aunque este proyecto de investigación logró llevar a buen término los objetivos propuestos, aún existen algunos puntos sobre los que conviene realizar mejoras. En este apartado se recomienda:

- Aplicar la evaluación por prototipos luego de que se está seguro acerca del diseño de la arquitectura y asegurarse de transmitir la diferencia entre prototipo y versión final a los evaluadores.
- Usar la técnica de evaluación por prototipos en los casos en los que se pretende evaluar los atributos de calidad que son más visibles durante la ejecución (aunque todos los diferentes atributos se pueden evaluar), como el rendimiento y la usabilidad
- Incluir en el diseño de la arquitectura soluciones a lo los problemas de usabilidad que se resaltaron en la evaluación de la arquitectura, como lo son las acciones de copiar, pegar, deshacer y cancelar
- Relacionado con el punto anterior también se recomienda adicionar un módulo de extracción automático de metadatos que permita reducir el tiempo de enriquecimiento de los objetos buscando metadatos en fuentes externas (p. e. Wikipedia).
- Restructurar parte del Transcodificador para mejorar su extensibilidad.
- Desarrollar una versión completa de la arquitectura tomando como base el prototipo ya diseñado.
- Adicionar nuevas funcionalidades a la arquitectura, sin que esto involucre una modificación al diseño, tal como estaba planteado en los escenarios de extensibilidad. Por ejemplo, adicionar a la herramienta de creación nuevas herramientas de dibujo y de selección de objetos por reconocimiento de bordes que permitan seleccionar áreas automáticamente.
- También se recomienda sumar nuevas plataformas de televisión al Transcodificador, en especial aquellas que puedan ser susceptibles de usar en nuestro entorno, como por ejemplo IPTV.

6.2 TRABAJOS FUTUROS

A pesar del trabajo realizado y del logro de los objetivos, esta arquitectura aún puede seguir evolucionando para acomodarse a nuevas necesidades. Adicionalmente pueden surgir nuevas ideas que complementen conceptos aquí tratados. Los siguientes temas pueden considerarse candidatos para trabajos futuros y para delinear nuevas rutas de investigación:

- **Extender la arquitectura**. Explorar desde el punto de vista arquitectónico como incluir los sistemas de difusión de las plataformas de televisión, al igual que las aplicaciones finales para los televidentes.
- **Definir una metodología**. Explorar el uso de la arquitectura en un ambiente real, donde se lleve a cabo la creación de una aplicación para un programa de televisión y su posterior transmisión, dando la posibilidad de definir una metodología de trabajo que integre la arquitectura como parte de su proceso.
- **Estudiar la usabilidad de aplicaciones que usen los objetos del video.** Independientemente de la arquitectura planteada, sería un gran aporte evaluar la usabilidad de este tipo de aplicaciones y poder medir su aceptación por parte de los televidentes.
- **Detección de Objetos y Escenas.** Una línea de investigación complementaria a los algoritmos de seguimientos y que permitiría mejorar el diseño de aplicaciones interactivas de video es la detección de objetos y de escenas. Esto podría servir para reconocer divisiones semánticas y/u objetos de un video, sin señalarlos manualmente.
- **Tiempo Real**. Adaptar el diseño de la arquitectura para crear aplicaciones para transmisiones en vivo.

BIBLIOGRAFÍA

1. DANE. CNTV. Colombia. Anuario Estadístico de la Televisión en Colombia 2009. (http://www.cntv.org.co). Disponible en: http://www.cntv.org.co/cntv_bop/estudios/anuario_estadistico_09a.pdf. Consultado: 2 mayo 2011.
2. Morris, Steven; Smith-Chaigneau, Anthony. 2005. *Interactive TV Standards*. Focal Press. Estados Unidos. p21, 32. pp 585
3. Cesar, Pablo; Chorianopoulos Konstantinos. The Evolution of TV Systems, Content, and Users Toward Interactivity. En: *Foundations and Trends® in Human-Computer Interaction*. 2009; Vol. 2: No 4, pp 373-95.
4. The MHP Knowledge Project. *The MHP-Guide*. Version 1.0; 2006: pp. 19-29, 170, 201. Disponible en: http://www.mhp-knowledgebase.org/publ/mhp-guide.pdf
5. Tolva, John. MediaLoom: An Interactive Authoring Tool for Hypervideo. 1997:1-18. Disponible en: http://www.ascentstage.com/medialoom/paper.html
6. Chorianopoulos, Konstantinos; Spinellis, Diomidis. 2004. Affective usability evaluation for an interactive music television channel. En: ACM Computers in Entertainment. Vol. 2: No 3.
7. Goldman, Dan; Gonterman, Chris; Curless, Brian; Salesin, David; Seitz, Steven. 2008. *Video object annotation, navigation, and composition*. *Proceedings of the 21st annual ACM symposium on User interface software and technology - UIST '08*. ACM. pp 3-12.
8. Kunert, Tibor. 2009. *User-Centered Interaction Design Patterns for Interactive Digital Television Applications*. Springer, London; p.16-17, 33-40.
9. Cardoso, Bernardo; De Carvalho, Fausto; Fernàndez, Gabriel; Huet, Benoit; Otros. Agosto 2004. Personalization of Interactive Objects in the GMF4iTV project. En: *Proceedings of the* TV'04: the 4th Workshop on Personalization in Future TV, Eindhoven, Holanda.

10. European Telecommunications Standards Institute, ETSI TS 102 727 V1.1.1 (2010-01). Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.2.2.
11. Livelink System. TeleClix. http://www.teleclix.com/Products_&_Services/TeleClix_LiveLink_system/ENG, Consultado: 30 Junio 2011.
12. Rodriguez-Alsina, Aitor; Moreno-Berengue, Marc; Carrabina, Jordi. Octubre 2010, PCF Dynamic Data Transfers for Web-Based ITV Services. En: Cumbre NEM 2010. Disponible en: http://nem-summit.eu/wp-content/plugins/alcyonis-event-agenda//files/PCF-Dynamic-Data-Transfers-for-Web-Based-iTV-Services.pdf (Consultado: 12 julio 2011)
13. Multimedia Home Platform. MHP. 2008. http://www.mhp.org/. Consultado: 21 Diciembre 2011.
14. ISO 9241–11. 1998. Ergonomics requirements for office work with visual display terminals (VDTs)—part 11: guidance on usability. International Standard.
15. Adivi ProductionKit. http://www.adivi.net/en/productionkit.html Consultado: 19 Julio 2011.
16. Quick.TV. http://www.quick.tv/, Consultado: 30 Mayo 2011.
17. Overlay.TV. http://www.overlay.tv/ Consultado: 5 Junio 2011.
18. wireWAX. http://www.wirewax.com/. Consultado: 16 Junio 2011
19. VideoClix. http://www.videoclix.tv/, Consultado: 29 Mayo 2011.
20. Hsin-Ta Chiao, Kuo-Shu Hsu, Yu-Kai Chen, Shyan-Ming Yuan. 2006. *A Template-Based MHP Authoring Tool*. En: *The Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*. Seoul, Korea del Sur. p: 138.
21. European Broadcasting Union. Multimedia. Noviembre 2004. http://www.ebu.ch/en/multimedia/index.php?display=EN#0, Consultado: 9 Junio 2011
22. Tsekleves, Emmanuel; Cosmas, John; Aggoun, Amar; Loo, Jonathan. 2009. *Converged digital TV services: The role of middleware and future directions of interactive television. International Journal of Digital Multimedia Broadcasting. Hindawi Publishing Corporation*. pp: 1-20.
23. Benoit, Hervé. 2008. *Digital Television: Satellite, Cable, Terrestrial, IPTV, Mobile TV in the DVB Framework*. 3ra Ed. Focal Press. pp 289

24. DVB. Resumen: *Portable Content Format*. 2010. http://www.dvb.org/technology/fact_sheets/DVB-PCF_Factsheet.pdf Consultado: 12 Julio 2011.
25. *European Telecommunications Standards Institute*, ETSI TS 102 523. Digital Video Broadcasting; Portable Content Format Specification 1.0.
26. R. Bradbury, R. Cartwright, J. Hunter, S. Perrott, J.E. Rosser . 2006. BBC R&D White Paper #134. Portable Content Format: a standard for describing an interactive digital television service. http://downloads.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP134.pdf Consultado: 12 Agosto 2011.
27. Arnold, John; Frater, Michael R.;Pickering, Mark. 2007. *Digital Television: technology and standards*. Hoboken, NJ. Wiley-Blackwell. pp: 644.
28. Alencar, Marcelo S. 2009. *Digital Television System*. Cambridge University Press. pp: 285
29. Ordóñez, César; Herrera, José. 2009. Evaluación de la televisión interactiva desde una perspectiva de usabilidad. Ciencia e Ingeniería Neogranadina. Vol 19-1. Bogotá. pp: 99-106.
30. Lu, Karyn Y. 2005. Tesis: *Interaction Design Principles for Interactive Television*. *Computing*. Georgia Institute of Technology. pp: 202
31. Felipe A. de Almeida, Reinaldo Silva Fortes, Baccan, Davi D'Andrea. 2004, *The Cossack System: A Platform for Interactive TV*. En: *2nd European Conference on Interactive Television: Enhancing the Experience*. Brighton. p 133- 138.
32. *Interoperability. EBU Position Papers - Telecommunications Package*. 2004. http://www.ebu.ch/en/legal/position/tp/index.php Consultado: 1 Octubre 2011
33. Brown, Adam Watson. 2005. *Interoperability, standards and sustainable receiver markets in the European Union. Ebu Technical Review*. pp 16.
34. Matteucci, Nicola. 2008. *Open standards and interoperability in EU digital TV: economics and policy issues*. En: *Review of Economic Research on Copyright Issues*. Volumen 5. Número 2. pp. 45-70.
35. Tsekleves, Emmanuel, Cosmas, John. 2007. *Semi-Automated Creation of Converged ITV Services: From Macromedia Director Simulations to Services Ready for Broadcast*. En: *Journal of Virtual Reality and Broadcasting*. Volumen:4. Número: 17. pp 14.

36. Jouve, Wilfried. 2004. *Enriching Multimedia Content Description for Broadcast Environments : From A Unified Metadata Model to A New Generation of Authoring Tool*. Tesis de Maestría. INRIA Rennes. pp 51.
37. Chun, Sooduck; Shin, Junghoon; Lee, Sangjun; Lee, Soowon; Oh, Kyoung-Su.2009. *Design and Implementation of Digital Contents Authoring Tools with Metadata*. En: *International Conference on Computer Engineering and Technology. IEEE computer society*. pp 534-537.
38. Emili, Prado; Rosa, Franquet; Teresa, María; Ribes, Xavier; Quijada, David Fernández. 2008. Tipología funcional de la televisión interactiva y de las aplicaciones de interacción con el televisor. ZER. Vol.13. No 25. pp.11-35.
39. Bordignon, Alexandro; Varella, Fernando; et. al. 2009. *Mechanisms for interoperable content production among Web, Digital TV and Mobiles*. *Informática Na Educação: teoria & prática*. Volumen: 12. Número: 1. pp. 110-118.
40. Andreadis, Alessandro; Baldo, David; Benelli, Giuliano; Luca Daino, Giovanni. 2007. *Towards ITV applications' portability across digital terrestrial television frameworks.* En: *15th International Conference on Software, Telecommunications and Computer Networks*. pp: 1-4.
41. Song, Ha; Park, Jongwook. 2006. *Design of an interoperable middleware architecture for digital data broadcasting*. En: *IEEE Transactions on Consumer Electronics*. Volumen: 52. Número: 4. pp.:1433-1441.
42. Cox, Mike; Mulder, Ellen; Tadic, Linda. 2006. *Descriptive Metadata for Television.* Focal Press. pp: 541
43. MPEG-7 Overview (versión 10). ISO/IEC. 2004 http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm. Consultado: 4 Diciembre 2011.
44. MPEG-21 Overview (versión 5). ISO/IEC. 2004 www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm. Consultado: 4 Noviembre 2011.
45. *European Telecommunications Standards Institute, ETSI TS 102 822-3-1 v1.6.1. Broadcast and On-line Services: Search, select and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 1: Phase 1 - Metadata schemas.*

46. Collins, Gerald W. 2001. *Fundamentals Of Digital Television Transmission.* John Wiley. ,pp 267.
47. Hölbling, G. Rabl, Tilmann; Kosch, Harald. 2008. *Overview of Open Standards for Interactive TV (iTV). Studies in Computational Intelligence*. Volumen: 101. p. 45–64.
48. Amatller Clarasó, Jordi; Baldo, David; Benelli, Giuliano; Daino, Giovanni Luca; Zambon, Riccardo. 2009. *Interactive Digital Terrestrial Television: The Interoperability Challenge in Brazil*. En: *International Journal of Digital Multimedia Broadcasting*. Volumen: 2009. Hindawi Publishing Corporation. pp. 1-18.
49. Ferreira Moreno, Marcio. 2009. *Ginga-NCL: Relating Imperative, Declarative and Media Objects*. En: *Networked Television Adjunct proceedings of EuroITV 2009*. Leuven, Bélgica. pp: 152-155.
50. *European Telecommunications Standards Institute, ETSI TS 102 796 (V1.1.1). Hybrid Broadcast Broadband TV.*
51. L. Eronen. 2006. *Five qualitative research methods to make iTV applications universally accessible*, *Universal Access in the Information Society*, vol. 5, no. 2, pp. 219-238.
52. *DVB Project, 2010. Globally Executable MHP (GEM) Specification* 1.3 vol. 3, , pp. 1-860.
53. Google TV. http://www.google.com/tv. Consultado: 10 de mayo de 2012
54. *Applications Devolepment Guide - Samsung Smart TV Apps Developer Forum*. http://www.samsungdforum.com/Guide/GuideList. Consultado:10 de mayo de 2012
55. *Synchronized Multimedia Integration Language* (SMIL 3.0). http://www.w3.org/TR/SMIL/ Consultado: 13 de Mayo de 2012
56. P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, and J. Stafford, *Documenting Software Architectures*, 2da. Edición. Addison-Wesley, 2010, p. 537.
57. L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2da. Edición. Addison-Wesley, 2003, p. 560.
58. D. Spinellis y G. Gousios, Editores., *Beautiful architecture*. 2009. p. 404.
59. R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, 2000, p. 162.

60. N. May, *A survey of software architecture viewpoint models*, *The Sixth Australasian Workshop on Software and System Architectures*, 2005.
61. P. Kruchten, Architectural Blueprints - The '4+1' View Model of Software Architecture, *IEEE Software*, vol. 12, no. 6, pp. 42–50, 1995.
62. G. Bradski and A. Kaehler, *Learning OpenCV*, vol. 16, no. 3. O'Reilly, 2008, p. 555.
63. R. Laganiere, *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing Ltd., 2011, p. 287.
64. V. Silva, *Practical Eclipse Rich Platform Projects*. Apress, 2009, p. 335
65. M. Lindvall, *An Empirically-Based Process for Software Architecture Evaluation*, *Empirical Software Engineering*, vol. 8, pp. 83–108, 2003.
66. M. Hoffmann, *Quality Evaluation of Software Architecture with Application to OpenH.323 Protocol*, University of Jyväskylä, 2006.
67. F. Mårtensson, H. Grahn, and M. Mattsson, An Approach for Performance Evaluation of Software Architectures using Prototyping, *7th IASTED International Conference on Software Engineering and Applications*, 2003.
68. J. Bardram, H. Christensen, and A. *Corry, Exploring Quality Attributes Using Architectural Prototyping*. En: *QoSA-SOQUA*, 2005, pp. 155–170.
69. J. Bardram, H. Christensen, and K. Hansen, *Architectural prototyping: an approach for grounding architectural design and learning.* En: *Proceedings. Fourth Working IEEE/IFIP Conference on Software Architecture*, pp. 15–24, 2004.
70. E. Folmer, J. V. Gurp, and J. Bosch, *Software Architecture Analysis of Usability*, *Engineering Human Computer Interaction …*, pp. 1–19, 2005.
71. Visauta Vinacua, B. A., & Martori I Cañas, J. C. 2003. Análisis Estadístico Con SPSS Para Windows: Estadística Multivariante. Madrid: MCGRAW-HILL.

ANEXO A DOCUMENTO DE REQUISITOS Y CASOS DE USO

A.1 DEFINICIÓN DE REQUISITOS

En esta sección se pretende plasmar de manera formal los diferentes requisitos funcionales y atributos de calidad que la arquitectura debe satisfacer.

El sistema permitirá crear aplicaciones a partir de la selección de objetos en una secuencia de video, sobre los que se podrán adicionar metadatos y recursos, y posteriormente exportar a diferentes plataformas de televisión interactiva.

A.1.1 Identificación de Actores del Sistema

Tabla 15 Actores del sistema

Nombre del Actor	Descripción del actor y Responsabilidad dentro del sistema
Usuario	Es el actor principal de la arquitectura. Es el encargado de crear las aplicaciones.
Televidente	Es el usuario final de la aplicación. A través del uso de un STB asociado a una plataforma específica, el televidente interactuará con las aplicaciones resultantes

A.1.2 Descripción de casos de uso

Casos de uso para el Actor: Usuario

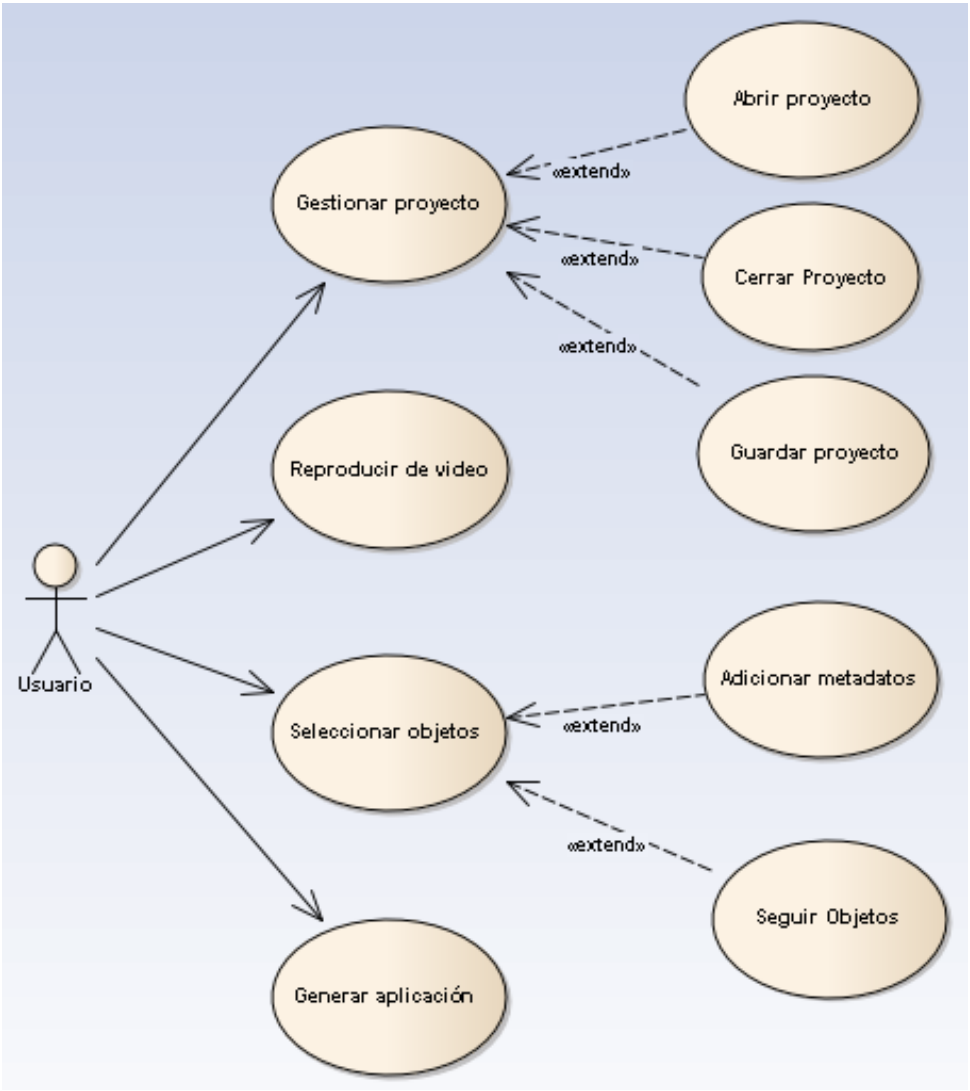

Figura 22 Diagrama de Casos de Uso del actor Usuario

Leyenda

CU = Caso de Uso

Tabla 16 Casos de Uso para el actor Usuario

ID	Nombre	Descripción
CU1	**Gestionar proyecto**	Este caso de uso permite al usuario ejecutar acciones tipo CRUD sobre los proyectos creados para el desarrollo de televisión interactiva. Un proyecto se creará con la selección de un video y una asignación de un nombre.
CU1.1	**Abrir proyecto**	El usuario estará en la capacidad de abrir proyectos previamente creados que estén cerrados.
CU1.2	**Cerrar proyecto**	El usuario estará en la capacidad cerrar proyectos que estén abiertos.
CU1.3	**Guardar proyecto**	El usuario estará en capacidad de guardar un proyecto.
CU2	**Reproducir Video**	Este caso de uso permite al usuario la utilización de un reproductor de video dentro de la aplicación.
CU3	**Seleccionar objetos**	El sistema debe permitir dibujar sobre el video que está siendo reproducido para seleccionar objetos.
CU3.1	**Adicionar metadatos**	El sistema debe permitir enriquecer los objetos seleccionados mediante la adición de metadatos.
CU3.2	**Seguir objetos**	El sistema debe permitir hacer seguimiento automático de los objetos seleccionados.
CU4	**Generar Aplicación**	El sistema debe permitir exportar las aplicaciones diseñadas a diferentes plataformas de televisión.

Casos de uso para el Actor: Televidente

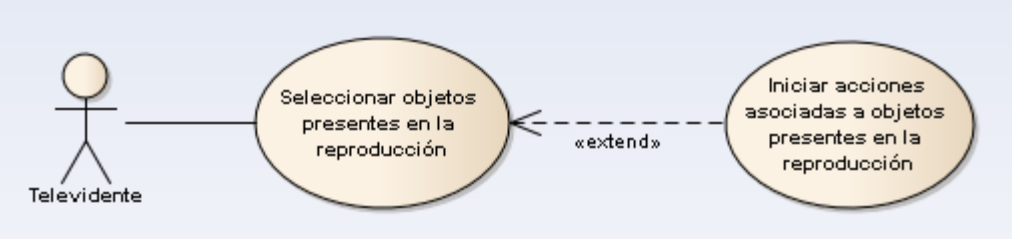

Figura 23 Diagrama de Casos de Uso del actor Televidente

Tabla 17 Casos de Uso para el actor Televidente

ID	**Nombre**	**Descripción**
CU5	**Seleccionar objetos**	El televidente podrá seleccionar y utilizar los objetos o áreas existentes en la reproducción de video.
CU5.1	**Iniciar acciones**	El sistema ejecutará las acciones disponibles sobre cada objeto existente en la secuencia de video.

A.1.3 Descripción detallada de los Requisitos Funcionales

Leyenda

RF = Requisito Funcional

IR = Importancia del requisito: (E)sencial, (D)eseable, (O)pcional

DI = Dificultad de Implementación (A)lta, (M)edia, (B)aja

ER = Estado del Requisito: (D)efinido, (A)probado, (R)echazado, (I)mplementado

Tabla 18 Requisitos Funcionales

ID	**Requisito**	**IR**	**DI**	**ER**	**CU Asociado**
RF1	**Permitir crear proyectos de desarrollo para TV interactiva.** Se permitirá crear un proyecto de desarrollo para aplicaciones de televisión interactiva a partir de la selección de un video como entrada. El proyecto deberá	E	M	I	CU1

	ser representado con un nombre. La información del proyecto contendrá: El nombre del proyecto La ruta del video al cual pertenece el proyecto La ruta del archivo en el que se almacenarán los metadatos. El archivo seleccionado se cargará en el reproductor de video.				
RF1.1	**Permitir la extracción automática de metadatos del audio y video.** Se deberán extraer automáticamente metadatos relacionados con los flujos de audio y video durante la creación del proyecto. Campos requeridos (no se limita solo a estos): • Duración • Cuadros por segundo y cuadros totales • Ancho y alto del video • Información de códecs de audio y video • Estos datos se almacenarán en los archivos del proyecto.	E	A	I	CU1
RF1.2	**Permitir abrir proyectos.** El usuario podrá abrir un proyecto a partir de los archivos creados en el requerimiento RF1. Si existe un proyecto abierto se debe ejecutar el caso de uso CU1.2 Cerrar proyecto actual antes de abrir el nuevo proyecto El video relacionado con el proyecto que acaba de abrirse se cargará en el reproductor de video.	E	B	I	CU1.1 CU1.2
RF1.3	**Permitir cerrar proyectos.** El usuario podrá cerrar un proyecto que esté actualmente cargado. Si el proyecto tiene cambios sin guardar se debe dar la posibilidad al usuario de ejecutar el caso de uso CU1.3 Guardar proyecto antes de cerrar el proyecto. El video abierto actualmente en el reproductor será	E	B	I	CU1.2

	cerrado.				
RF1.4	**Permitir guardar proyectos.** El sistema permitirá al usuario guardar los cambios realizados en un proyecto en cualquier momento, estos cambios se reflejarán en el archivo del proyecto y en los repositorios de metadatos	E	A	I	CU1.3
RF2	**Permitir al usuario utilizar un reproductor de video.** El usuario podrá usar un reproductor para visualizar el vídeo asociado al proyecto y para realizar el diseño de la aplicación que se desarrollará sobre este. Debe existir un proyecto para habilitar las funciones de reproducción, estas son: reproducir, pausar, detener, adelantar y retroceder rápidamente y cuadro a cuadro.	E	A	I	CU2
RF2.1	**Permitir al usuario la visualización de diferentes formatos de video.** El sistema deberá permitir al usuario la manipulación de múltiples formatos de video y la adición de nuevos códecs.	D	A	I	CU2
RF3	**Permitir dibujar en el video zonas que indiquen selección de objetos.** El usuario podrá dibujar sobre el video, un área que indique que un objeto ha sido seleccionado. Esto aplica para cualquier cuadro de video. Está área quedará relacionada con el cuadro que esté actualmente en el reproductor. Los colores a usar son rojo, verde, azul y amarillo.	E	A	I	CU3
RF3.1	**Permitir el borrado de selecciones hechas previamente.** El sistema permitirá borrar las áreas seleccionadas por los usuarios. El usuario deberá seleccionar una o más áreas para borrar y el sistema las eliminará del cuadro en el que fue seleccionado y en adelante.	E	M	I	CU3
RF3.2	**Permitir el enriquecimiento con metadatos de zonas seleccionadas.** El usuario tendrá la posibilidad de adicionar metadatos en los campos habilitados para dicho	E	A	I	CU3.1

	fin. Esta funcionalidad se podrá dividir en: • Identificación del área seleccionada mediante una imagen y un nombre. • Adición de texto libre mediante una descripción. • Adición de metadatos compuestos por nombre y valor. • Adición de recursos o archivos asociados. Los metadatos a almacenar deben contener: • Información general y técnica del video como tamaño, duración, cuadros por segundo. • Información de la segmentación espacio-temporal del video. Es decir especificación de marcas de tiempo y de coordenadas. • Información adicionada a los objetos seleccionados. • Información de los archivos asociados a los objetos seleccionados				
RF3.2.1	**Permitir compartir recursos.** Los archivos asociados a un objeto deben quedar disponibles para las diferentes aplicaciones generadas para evitar la replicación de archivos.	O	M	I	CU3.1 CU4
RF3.2.2	**Permitir la reutilización de metadatos.** El usuario podrá asignar metadatos creados con anterioridad a un área seleccionada, con el fin de reutilizar la información, esto quiere decir, que la información de un objeto puede aparecer en varios videos y/o varias veces en un mismo video, además de estar asociado a diferentes áreas.	D	B	I	CU3.1
RF3.3	**Permitir al usuario hacer seguimiento de objetos.** El usuario podrá seleccionar el algoritmo para el seguimiento de los objetos seleccionados.	E	A	I	CU3 CU3.2

	Se deberán implementar algoritmos de visión artificial que permitan realizar seguimientos de objetos. Las áreas seleccionadas en el caso de uso CU3 serán usadas por estos algoritmos para encontrar coincidencias en los cuadros siguientes.				
RF4	**Permitir al usuario transformar el proyecto a una o varias plataformas de televisión interactiva.** El usuario podrá exportar el diseño creado (áreas y metadatos) a una aplicación para una plataforma de televisión interactiva.	E	A	I	CU4
RF4.1	**Permitir al usuario generar diferentes aplicaciones con el mismo diseño.** El usuario podrá exportar el diseño creado (áreas y metadatos) a diferentes tipos de aplicaciones para una misma plataforma. Por ejemplo el usuario podría generar una aplicación para desplegar la información entrada en diseño y/o podría generar otra que use la misma información pero esta vez para crear un carrito de compras.	D	A	I	CU4
RF5	**Permitir al televidente ejecutar la aplicación cliente.** Cuando el televidente seleccione un vídeo o seleccione un canal, la aplicación generada deberá ser reproducida sincrónicamente con el flujo de video.	E	M	I	CU5
RF5.1	**Permitir al televidente la selección de objetos.** El televidente podrá efectuar selecciones de los objetos que se encuentren presentes en el flujo de video. Se debe tener en cuenta que el usuario solo podrá seleccionar objetos que se encuentren visibles en un momento dado, es decir, si un color no se encuentra presente en la pantalla, su botón asociado en el control remoto no realizará ninguna acción.				CU5
RF5.2	**Permitir al televidente iniciar acciones asociadas a su selección.** El televidente podrá ejecutar las acciones	E	A	I	CU5.1

	asociadas a los objetos que seleccione, dependiendo de la especificación de la aplicación creada.				

A.1.4 Descripción de atributos de calidad

Leyenda

QA = Atributo de Calidad

IR = Importancia del requisito: (E)sencial, (D)eseable, (O)pcional

DI = Dificultad de Implementación (A)lta, (M)edia, (B)aja

ER = Estado del Requisito: (D)efinido, (A)probado, (R)echazado, (I)mplementado

Tabla 19 Atributos de Calidad

ID	Atributo	IR	DI	ER
QA1	**Disponibilidad.** El sistema deberá permitir al usuario mantener el estado de un proyecto de desarrollo para que en caso de fallas se puede recuperar desde el último estado guardado.	E	M	I
QA2	**Manejo de Errores.** La arquitectura contará con una bitácora de mensajes que informen sobre la evolución de los procesos, posibles problemas o errores de ejecución, con el fin de ayudar a la realización de labores de mantenimiento y/o detección de problemas.	D	B	I
QA3	**Internacionalización.** Debe ser posible especificar los diferentes textos y mensajes visibles en las herramientas en diferentes idiomas sin la necesidad de cambiar el código fuente.	D	B	I
QA4	**Usabilidad**. La tarea de selección de un área no puede tardar más de un minuto.	E	B	I
QA5	**Rendimiento.** La velocidad de los algoritmos de seguimiento de objetos no pueden tomar más de un segundo por cuadro de video.	E	A	I
QA6	**Rendimiento (sincronización).** El tiempo de retraso máximo permitido para la sincronización entre el video y la aplicación será de un segundo,	E	A	I

	en caso de no cumplirse se desecharán los gráficos que no alcancen a reproducirse.			
QA7	**Extensibilidad.** La arquitectura debe soportar que un desarrollador adicione funcionalidades para dibujar y para realizar seguimiento de objetos en menos de 1 día sin tener que modificar el código actual. Este tiempo solo contempla unir los componentes, crear las opciones y adicionar iconos entre otros, pero no el tiempo de desarrollo de la funcionalidad como tal.	D	A	I
QA8	**Extensibilidad de plataformas.** La arquitectura debe soportar que un desarrollador adicione la transformación a plataformas de televisión no implementadas en menos una semana.	E	A	I

A.2 ESPECIFICACIÓN DE REQUISITOS SOFTWARE / HARDWARE

La arquitectura será diseñada para soportar cualquiera de las plataformas de televisión interactiva presentes en el laboratorio de televisión digital. Para esto se requiere que cada una de las plataformas cuentan con su propio decodificador, ya sea una caja independiente en el lado del cliente, o como en el caso de HBBTV, un decodificador embebido en el propio televisor.

Por ser una etapa tan temprana en el diseño de la arquitectura aún no hay suficiente información sobre los requisitos de software y hardware que puedan ser necesarios.

ANEXO B METADATOS

B.1 ESQUEMA ACTIVA

El archivo activa_metadata.xsd describe la extensión de TV-Anytime

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://maestria.eafit.edu/activa/metadata"
    elementFormDefault="qualified" attributeFormDefault="unqualified"
    xmlns:activa="http://maestria.eafit.edu/activa/metadata"
    xmlns:activampeg7="http://maestria.eafit.edu/activa/metadata/mpeg7"
    xmlns:tva="urn:tva:metadata:2011" xmlns:mpeg7="urn:tva:mpeg7:2008"
    xmlns="http://www.w3.org/2001/XMLSchema">

    <annotation>
            <documentation xml:lang="en">
             This schema consists of the structure necessary to define video objects
            </documentation>
    </annotation>

    <import namespace="http://maestria.eafit.edu/activa/metadata/mpeg7"
      schemaLocation="activa_mpeg7_v2.xsd" />
    <import namespace="urn:tva:metadata:2011"
      schemaLocation="tva_metadata_3-1_v171.xsd" />
    <import namespace="urn:tva:metadata:extended:2011"
      schemaLocation="tva2_metadata_3-3_v151.xsd" />
    <import namespace="urn:tva:mpeg7:2008"
      schemaLocation="tva_mpeg7_2008.xsd" />
    <import namespace="http://www.w3.org/XML/1998/namespace"
      schemaLocation="xml.xsd" />

     <!-- ################################################## -->
     <!-- Definition of Property type of an Entity          -->
     <!-- ################################################## -->
        <complexType name="PropertiesType">
            <sequence>
                <element name="Property" maxOccurs="unbounded">
                    <complexType>
                        <sequence>
                            <element name="Key" type="string"/>
                            <element name="Value" type="string"/>
                        </sequence>
                    </complexType>
                </element>
            </sequence>
        </complexType>

```

```
    <!-- ################################################ -->
    <!-- Definition of Resource type                      -->
    <!-- ################################################ -->
       <complexType name="ResourcesType">
           <sequence>
               <element name="Resource" maxOccurs="unbounded">
                   <complexType>
                       <sequence>
                           <element name="Name" type="string"/>
                           <element name="HREF"
                                type="mpeg7:termReferenceType"/>
                       </sequence>
                   </complexType>
               </element>
           </sequence>
       </complexType>

    <!-- ################################################ -->
    <!-- Definition of Entity Type                        -->
    <!-- ################################################ -->
       <complexType name="EntityType">
           <sequence>
               <element name="Name" type="string"/>
               <element name="Description" type="mpeg7:TextualType"/>
               <element name="Image" type="mpeg7:termReferenceType"/>
               <element name="Properties" type="activa:PropertiesType"
                 minOccurs="0"/>
               <element name="Resources" type="activa:ResourcesType"
                 minOccurs="0"/>
           </sequence>
           <attribute name="entityId" type="ID" use="required"/>
           <attribute name="type" type="string" use="required"/>
       </complexType>

    <!-- ################################################ -->
    <!-- Definition of the table of Entities              -->
    <!-- ################################################ -->
       <complexType name="EntityInformationTableType">
           <sequence>
               <element name="Entity" type="activa:EntityType"
                 maxOccurs="unbounded"/>
           </sequence>
       </complexType>

    <!--##################################################################-->
    <!--Definition of a extended version of tva:programaDescriptionType -->
    <!--##################################################################-->
       <complexType name="ExtendedProgramDescriptionType">
           <complexContent>
               <extension base="tva:ProgramDescriptionType">
                   <sequence>
                       <element name="EntityInformationTable"
                         type="activa:EntityInformationTableType"
                         minOccurs="0"/>
                   </sequence>
               </extension>
           </complexContent>
       </complexType>

```

```
    <!-- ################################################ -->
    <!-- Definition of Entity Region Type                 -->
    <!-- ################################################ -->
       <complexType name="EntityRegionType">
           <sequence>
               <element name="Region" type="activampeg7:RegionLocatorType"/>
           </sequence>
           <attribute name="entityRef" type="IDREF" use="required"/>
       </complexType>

    <!-- ################################################################ -->
    <!-- Definition of a extended version of tva:SegmentInformationType -->
    <!-- ################################################################ -->
       <complexType name="ExtendedSegmentInformationType">
           <complexContent>
               <extension base="tva:SegmentInformationType">
                   <sequence>
                       <element name="EntityRegion" type="activa:EntityRegionType"
                         maxOccurs="unbounded"/>
                   </sequence>
               </extension>
           </complexContent>
       </complexType>
   </schema>
```

B.2 ESQUEMA ACTIVA MPEG-7

El archivo activa_mpeg7.xsd incluye los elementos del estándar MPEG-7 necesarios para describir la información espacio-temporal de las entidades.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://maestria.eafit.edu/activa/metadata/mpeg7"
    xmlns:activampeg7="http://maestria.eafit.edu/activa/metadata/mpeg7"
    xmlns:mpeg7="urn:tva:mpeg7:2008"
    xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
    <annotation>
  <documentation>
   This schema is composed of description tools in the MPEG-7
   necessary to describe objects in a video
  </documentation>
    </annotation>

    <import namespace="urn:tva:mpeg7:2008"
        schemaLocation="tva_mpeg7_2008.xsd"/>
    <import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="xml.xsd"/>

 <!-- ################################################ -->
```

```
<!-- Definition of 'mpeg7:dim" for Matrix Datatype   -->
<!-- ################################################## -->
<!-- definition of listOfPositiveIntegerForDim datatype -->
<simpleType name="listOfPositiveIntegerForDim">
 <list itemType="positiveInteger"/>
</simpleType>
<!-- definition of mpeg7:dim attribute -->
<attribute name="dim">
 <simpleType>
  <restriction base="activampeg7:listOfPositiveIntegerForDim">
   <minLength value="1"/>
  </restriction>
 </simpleType>
</attribute>

<!-- ############################################################ -->
<!-- Definition of RegionLocator Datatype            -->
<!-- ############################################################ -->
<!-- Definition of RegionLocator Datatype -->
<complexType name="RegionLocatorType" final="#all">
 <sequence>
  <element name="CoordRef" minOccurs="0">
   <complexType>
    <attribute name="ref" type="IDREF" use="required"/>
    <attribute name="spatialRef" type="boolean" use="optional"
       default="false"/>
   </complexType>
  </element>
  <element name="Box" minOccurs="0" maxOccurs="unbounded">
   <complexType>
    <simpleContent>
     <extension base="activampeg7:BoxListType">
      <attribute name="unlocatedRegion" type="boolean" use="optional"
           default="false"/>
     </extension>
    </simpleContent>
   </complexType>
  </element>
  <element name="Polygon" minOccurs="0" maxOccurs="unbounded">
   <complexType>
    <sequence>
     <element name="Coords" type="activampeg7:IntegerMatrixType"/>
    </sequence>
    <attribute name="unlocatedRegion" type="boolean" use="optional"
       default="false"/>
   </complexType>
  </element>
 </sequence>
</complexType>
<!-- Definition of BoxList Datatype -->
<complexType name="BoxListType">
 <simpleContent>
  <restriction base="activampeg7:IntegerMatrixType"/>
 </simpleContent>
</complexType>

<!-- ######################################### -->
<!-- Definition of vector datatypes (5.4.2) -->
<!-- ######################################### -->
```

```
80.   <!-- Definition of integerVector datatype -->
81.   <simpleType name="integerVector">
82.    <list itemType="integer"/>
83.   </simpleType>
84.   <!-- ######################################## -->
85.   <!-- Definition of Matrix datatypes (5.4.4)  -->
86.   <!-- ######################################## -->
87.   <!-- Definition of IntegerMatrix datatype -->
88.   <complexType name="IntegerMatrixType">
89.    <simpleContent>
90.     <extension base="activampeg7:integerVector">
91.      <attribute ref="activampeg7:dim" use="required"/>
92.     </extension>
93.    </simpleContent>
94.   </complexType>
95. </schema>
```

ANEXO C CÓDIGO FUENTE SEGUIMIENTO DE OBJETOS

C.1 TEMPLATE MATCHING

Código fuente de *Template Matching*, por simplicidad se quitaron los mensajes de depuración *y* verificaciones de variables nulas entre otras.

```
1.  package edu.eafit.maestria.activa.objecttracking.services;
2.
3.  import static com.googlecode.javacv.cpp.opencv_core.IPL_DEPTH_32F;
4.  import static com.googlecode.javacv.cpp.opencv_core.cvAddWeighted;
5.  import static com.googlecode.javacv.cpp.opencv_core.cvCopy;
6.  import static com.googlecode.javacv.cpp.opencv_core.cvCreateImage;
7.  import static com.googlecode.javacv.cpp.opencv_core.cvGet2D;
8.  import static com.googlecode.javacv.cpp.opencv_core.cvGetSize;
9.  import static com.googlecode.javacv.cpp.opencv_core.cvRect;
10. import static com.googlecode.javacv.cpp.opencv_core.cvRectangle;
11. import static com.googlecode.javacv.cpp.opencv_core.cvResetImageROI;
12. import static com.googlecode.javacv.cpp.opencv_core.cvSize;
13. import static com.googlecode.javacv.cpp.opencv_highgui.cvSaveImage;
14. import static com.googlecode.javacv.cpp.opencv_imgproc.CV_TM_SQDIFF_NORMED;
15. import static com.googlecode.javacv.cpp.opencv_imgproc.cvMatchTemplate;
16.
17. import java.awt.Point;
18. import java.awt.image.BufferedImage;
19. import java.io.File;
20. import java.util.ArrayList;
21. import java.util.List;
22.
23. import com.googlecode.javacv.cpp.opencv_core.CvPoint;
24. import com.googlecode.javacv.cpp.opencv_core.CvRect;
25. import com.googlecode.javacv.cpp.opencv_core.CvScalar;
26. import com.googlecode.javacv.cpp.opencv_core.IplImage;
27.
28. public class TemplateMatching extends Tracker{
29.
30.     public TemplateMatching(File video, String dir) {
31.         super(video, dir);
32.     }
33.
34. private static final double THRESHOLD_VAL = 0.07;
35.
36.     /*
37.      * tracks a shape specified by the coord in the params beggining in
38.      * currentTime and saves the initial shape in fileName
39.      * The results is the list of consecutive points, once the track
```

```
40.       * algorithm doesn't find anything the tracking stops and returns.
41.       * The espected content of the params is a rectangle
42.       * params[0] = x
43.       * params[1] = y
44.       * params[2] = width
45.       * params[3] = height
46.       *
47.       * The Tracker must be initialized.
48.       *
49.       */
50.      public List<? extends Object> track(long currentTime, int[] params, BufferedImag
         e template){
51.
52.          //setting the template image from image given
53.          IplImage templateimg = IplImage.createFrom(template);
54.
55.          //getting the base frame
56.          IplImage frame = fg.getFrame();
57.
58.          // take the next frame where the comparison begins
59.          // and scale the grayscale (to speed up detection)
60.          IplImage scaledFrame = grab();
61.
62.          IplImage scaledTemplate = Tracker.
63.            getScaledVersion(Tracker.getGrayVersion(templateimg));
64.
65.          //setting working area
66.          int[] roiData = Tracker.populateRoiInit(cvRect(params[0],
67.            params[1], params[2], params[3]), scaledFrame);
68.          CvRect roiRect = Tracker.setWorkingArea(scaledFrame, roiData);
69.
70.          IplImage result = cvCreateImage(
71.            cvSize(cvGetSize(scaledFrame).width(),
72.                   cvGetSize(scaledTemplate).width() + 1,
73.                   cvGetSize(scaledFrame).height() â€“
74.                   cvGetSize(scaledTemplate).height() + 1),
75.            IPL_DEPTH_32F, 1);
76.
77.          int matches = 0;
78.          List<Point> results = new ArrayList<Point>();
79.          int rep=1;
80.          boolean firstMatch = false;
81.
82.          do {
83.              matches = 0;
84.
85.              cvMatchTemplate(scaledFrame,scaledTemplate,result,
86.                   CV_TM_SQDIFF_NORMED);
87.              int yPart = 0;
88.              int xPart = 0;
89.              cvResetImageROI(scaledFrame);
90.              IplImage tmp = cvCreateImage(cvGetSize(scaledFrame),scaledFrame.depth(),
                     scaledFrame.nChannels());
91.              cvCopy(scaledFrame, tmp);
92.              for(int y = 0 ; y < result.height(); y++ ) {
93.                  for(int x = 0 ; x < result.width() ; x++ ) {
94.                      CvScalar s = cvGet2D(result, y, x);
95.                      if (s.val(0) <= THRESHOLD_VAL) {
96.                          matches++;
```

```
97.                         yPart += y;
98.                         xPart += x;
99.                     }
100.                }
101.            }
102.            int newX = 0;
103.            int newY = 0;
104.            if (matches == 0 && firstMatch)
105.                break;
106.            else if (matches != 0 && !firstMatch) {
107.                firstMatch = true;
108.            }
109.
110.            if (matches != 0) {
111.                int yMean = Double.valueOf(Math.ceil(yPart/matches)).intValue();
112.                int xMean = Double.valueOf(Math.ceil(xPart/matches)).intValue();
113.                newX = (roiData[0] + xMean)*Tracker.scale;
114.                newY = (roiData[1] + yMean)*Tracker.scale;
115.                results.add(new Point(newX, newY));
116.
117.                IplImage scaledTemplateFuture = Tracker.
118.                    getTrackingImg(scaledFrame, newX, newY,
119.                        scaledTemplate.width(),
120.                        scaledTemplate.height());
121.                cvAddWeighted(scaledTemplate, 0.6f,
122.                    scaledTemplateFuture, 0.4f, 0, scaledTemplate);
123.            }
124.
125.            //New Frame
126.            scaledFrame = grab();
127.
128.            //modifying the pivod point, the base point where to
129.            // set the working area
130.            if (matches != 0) {
131.                roiRect.x(newX);
132.                roiRect.y(newY);
133.                roiData = Tracker.populateRoiData(roiRect, scaledFrame);
134.            }
135.
136.            roiRect = Tracker.setWorkingArea(scaledFrame, roiData);
137.
138.            //while no matches and not the firstMatch jet, or there
139.            // is matches or there isn't found
140.            //the first match and the process repeats 50 times
141.            //already
142.        } while ((matches ==0 && !firstMatch && rep < 50)
143.            || matches != 0);
144.        return results;
145.    }
146.}
```

C.2 SURF

```
1.  package edu.eafit.maestria.activa.objecttracking.services;
2.
3.  import static com.googlecode.javacv.cpp.opencv_core.CV_RGB;
4.  import static com.googlecode.javacv.cpp.opencv_core.cvCircle;
5.  import static com.googlecode.javacv.cpp.opencv_core.cvCopy;
6.  import static com.googlecode.javacv.cpp.opencv_core.cvCreateImage;
7.  import static com.googlecode.javacv.cpp.opencv_core.cvGetSize;
8.  import static com.googlecode.javacv.cpp.opencv_core.cvLine;
9.  import static com.googlecode.javacv.cpp.opencv_core.cvPoint;
10. import static com.googlecode.javacv.cpp.opencv_core.cvRect;
11. import static com.googlecode.javacv.cpp.opencv_core.cvRectangle;
12. import static com.googlecode.javacv.cpp.opencv_core.cvResetImageROI;
13. import static com.googlecode.javacv.cpp.opencv_highgui.cvSaveImage;
14.
15. import java.awt.Rectangle;
16. import java.awt.image.BufferedImage;
17. import java.io.File;
18. import java.util.ArrayList;
19. import java.util.List;
20.
21. import com.googlecode.javacv.ObjectFinder;
22. import com.googlecode.javacv.cpp.opencv_core.CvPoint;
23. import com.googlecode.javacv.cpp.opencv_core.CvRect;
24. import com.googlecode.javacv.cpp.opencv_core.CvScalar;
25. import com.googlecode.javacv.cpp.opencv_core.IplImage;
26.
27. public class ObjectFinderExecutor extends Tracker {
28.
29.     public ObjectFinderExecutor(File video, String dir) {
30.         super(video, dir);
31.     }
32.
33.     /*
34.      * tracks a shape specified by the coord in the params beggining in
35.      * currentTime and saves the initial shape in fileName
36.      * The results is the list of consecutive points, once the track
37.      * algorithm doesn't find anything the tracking stops and returns.
38.      * The espected content of the params is a rectangle
39.      * params[0] = x
40.      * params[1] = y
41.      * params[2] = width
42.      * params[3] = height
43.      *
44.      * The Tracker must be initialized.
45.      *
46.      */
47.     public List<? extends Object> track(long currentTime, int[] params, BufferedImag
        e template){
48.
49.         //setting the template image from image given
50.         IplImage templateimg = IplImage.createFrom(template);
51.         //getting the base frame
52.         IplImage frame = fg.getFrame();
```

```
53.
54.         // take the next frame where the comparison begins
55.         // and scale the grayscale (to speed up detection)
56.         IplImage scaledFrame = grab();
57.
58.         IplImage scaledTemplate = Tracker.
59.             getScaledVersion(Tracker.getGrayVersion(templateimg));
60.
61.         //setting working area
62.         int[] roiData = Tracker.
63.             populateRoiInit(cvRect(params[0], params[1], params[2], params[3]),
64.                   scaledFrame);
65.         CvRect roiRect = Tracker.setWorkingArea(scaledFrame, roiData);
66.
67.         List<Rectangle> results = new ArrayList<Rectangle>();
68.
69.         double[] h = null;
70.         boolean firstMatch = false;
71.         do {
72.             ObjectFinder objFinder = null;
73.             ObjectFinder.Settings settings = new ObjectFinder.Settings();
74.             settings.setObjectImage(scaledTemplate);
75.             settings.setDistanceThreshold(0.8f);
76.             settings.setHessianThreshold(300);
77.             settings.setMatchesMin(4);
78.             settings.setRansacReprojThreshold(5);
79.             objFinder = new ObjectFinder(settings);
80.
81.             h = objFinder.find(scaledFrame);
82.
83.             long xmincorner = (roiData[0])*Tracker.scale;
84.             long ymincorner = (roiData[1])*Tracker.scale;
85.             long xmaxcorner = (roiData[0])*Tracker.scale;
86.             long ymaxcorner = (roiData[1])*Tracker.scale;
87.             cvResetImageROI(scaledFrame);
88.             int newX = 0;
89.             int newWidth = 0;
90.             int newY = 0;
91.             int newHeight = 0;
92.
93.             for (int i = 0; i < h.length;i++){
94.                 if (h[i] < 0)
95.                     h[i]=0;
96.             }
97.
98.             //comparing the Xs
99.             int xminpos = 0;
100.            int xmaxpos = 0;
101.            for (int i = 0; i < h.length-2;i+=2){
102.                xminpos = (h[xminpos] < h[i+2]) ? xminpos : i+2;
103.                xmaxpos = (h[xmaxpos] > h[i+2]) ? xmaxpos : i+2;
104.            }
105.
106.            xmincorner = (roiData[0] + (long)Math.ceil(h[xminpos]))
107.                * Tracker.scale;
108.            xmaxcorner = (roiData[0] + (long)Math.floor(h[xmaxpos]))
109.                * Tracker.scale;
110.
111.            //comparing the Ys
```

```
            int yminpos = 1;
            int ymaxpos = 1;
            for (int i = 1; i < h.length-2;i+=2){
                yminpos = (h[yminpos] < h[i+2]) ? yminpos : i+2;
                ymaxpos = (h[ymaxpos] > h[i+2]) ? ymaxpos : i+2;
            }

            ymincorner = (roiData[1] + (long)Math.ceil(h[yminpos]))
                * Tracker.scale;
            ymaxcorner = (roiData[1] + (long)Math.floor(h[ymaxpos]))
                * Tracker.scale;

            newX = Long.valueOf(xmincorner).intValue();
            newWidth = Long.valueOf(xmaxcorner).intValue() - newX;
            newY = Long.valueOf(ymincorner).intValue();
            newHeight = Long.valueOf(ymaxcorner).intValue() - newY;
            if (firstMatch
                && (xmaxcorner - xmincorner < newWidth/2
                || ymaxcorner - ymincorner < newHeight/2)) {
                break;
            } else if (xmaxcorner -xmincorner > newWidth/2
                  && ymaxcorner - ymincorner > newHeight/2) {
                firstMatch = true;
                results.add(new Rectangle(newX, newY, newWidth, newHeight));
                scaledTemplate = Tracker.
                  getTrackingImg(scaledFrame, newX, newY, newWidth, newHeight);
            }

            scaledFrame = grab();

            //modifying the pivod point, the base point where
            //to set the working area
            if (firstMatch) {
                roiRect.x(newX);
                roiRect.y(newY);
                roiRect.width(newWidth);
                roiRect.height(newHeight);
                roiData = Tracker.populateRoiInit(roiRect, scaledFrame);
            }

            roiRect = Tracker.setWorkingArea(scaledFrame, roiData);

            //while no matches and not the firstMatch jet, or there
            //is matches or there isn't found
            //the first match and the process repeats 50 times
            //already
        } while (((h==null || h.length==0)
            && !firstMatch && rep <50) || h!=null );

        return results;
    }

}
```

ANEXO D CUESTIONARIO DEL EXPERIMENTO PARA LA EVALUACIÓN DE LA ARQUITECTURA

A continuación se presenta el cuestionario aplicado durante el experimento diseñado para la evaluación de la arquitectura.

Este Formulario también puede ser encontrando el siguiente enlace:

https://docs.google.com/spreadsheet/viewform?formkey=dDJoQXBtd0lOMENFTUdieDZpdHlZN0E6MQ#gid=0

Experimento: Evaluación de la arquitectura acTiVa para el desarrollo de aplicaciones para televisión interactiva

Agradecemos su colaboración en este experimento. • Por favor diligencie la parte 1 y lea la parte 2 de la encuesta antes de comenzar su participación. • Diligencie la Parte 3 luego de haber participado en la experiencia "Desarrollo de una aplicación interactiva para video en Internet usando métodos tradicionales" • Diligencie la Parte 4 luego de haber participado en la experiencia "Desarrollo de una aplicación interactiva para video en Internet usando acTiVa"

---

* Required

PARTE 1. INFORMACIÓN GENERAL

Nombre:

¿Cuantas horas al día desarrolla?*

¿Las herramientas que usa para desarrollar tiene elementos WYSIWYG?* WYSIWYG: http://es.wikipedia.org/wiki/WYSIWYG

¿Tiene experiencia en desarrollo de aplicaciones para video? * No importa la plataforma (Web, MHP, etc.)

¿En cuantos proyectos de desarrollo de aplicaciones que involucren video ha participado? *

PARTE 2. ESPECIFICACIÓN DE LA APLICACIÓN A DESARROLLAR

Dado el siguiente caso de uso

DESCRIPCIÓN: Este caso de uso describe la interacción entre un televidente y una aplicación de televisión interactiva para el video "Best of 2011: Slow Motion, http://www.youtube.com/watch?v=kGIjetX6TBk", en la que algunos objetos del video tienen información adicional que puede ser desplegada cuando el usuario los seleccione. Los interacción se da usando el mismo objeto (no hay un menú, o un catalogo, o algo similar). El siguiente ejemplo les da una idea del objetivo: "Neiman Marcus - Shoppable Video. www.wirewax.com" en el que unos círculos señalan los accesorios de una modelo y cuando se da click sobre ellos se muestra una imagen del objeto y otra información adicional.

ACTOR: Televidente

RESTRICCIONES Y ESPECIFICACIONES DE IMPLEMENTACIÓN: Los colores que se pueden utilizar para señalar los objetos son los de las teclas de interacción del control remoto de un TV (Rojo, Verde, Amarillo, Azul).

ESCENARIO PRINCIPAL

1. El televidente sintoniza o carga la fuente de video que quiere ver.
2. El sistema muestra del segundo 0:22 al 0:28 de reproducción un área de color rojo resaltando el objeto 1
3. El televidente oprime en su control remoto la tecla que represente el color rojo mientras el objeto 1 aparece resaltado.

4. El sistema muestra la información adicional asociada al objeto seleccionado.
5. El sistema muestra del segundo 0:39 al 0:48 de reproducción un área de color azul resaltando el objeto 2
6. El televidente oprime en su control remoto la tecla que represente el color azul mientras el objeto 2 aparece resaltado.
7. El sistema muestra la información adicional asociada al objeto seleccionado.
8. El sistema muestra del minuto 4:05 al 4:25 de reproducción un área de color amarillo resaltando el objeto 3
9. El televidente oprime en su control remoto la tecla que represente el color amarillo mientras el objeto 3 aparece resaltado.
10. El sistema muestra la información adicional asociada al objeto seleccionado.

Objeto 1
Nombre: Marco Simoncelli
Descripción: (Cattolica, Italia, 20 de enero de 1987 - Sepang, Malasia; 23 de octubre de 2011)2 fue un piloto italiano de motociclismo de velocidad. Pilotaba una Honda del equipo San Carlo Honda Gresini Team en el Campeonato Mundial de Motociclismo de Velocidad de MotoGP, hasta el momento de su fallecimiento, en el Circuito de Sepang, en Malasia.
Tipo: tipo 1
Propiedades:

Apodo Pippo, Supersic
Competición MotoGP
Nacionalidad Italiano

Objeto 2
Nombre: Valentino Rossi
Descripción: (Urbino, 16 de febrero de 1979) es un piloto italiano de motociclismo que ha ganado 9 títulos mundiales en categorías diferentes del Mundial de Motociclismo (125c.c., 250c.c., 500c.c., y MotoGP de 990c.c. y 800c.c.), único motociclista en la historia en lograrlo. Conocido por los apodos que se ha dado a sí mismo en diferentes momentos de su carrera: The Doctor (El Doctor), Valentinik y Rossifumi. Es considerado por la mayoría de expertos y prensa especializada como el mejor piloto de la historia del motociclismo. Sus padres son Graziano Rossi y Stefanía Palma. Tiene un hermano (por parte de

madre), Luca Marini, y una hermanastra, Chiara Rossi. N. podios consecutivos en la categoría de MotoGP (22) igualando a Giacomo Agostini, es el piloto con más podios (167), poles (59) y puntos en la historia (4087), y tiene también el mayor número de victorias en la máxima categoría del motociclismo (78).

Tipo: tipo 1

Propiedades:

Apodo The Doctor, Il Dottore

Competición MotoGP

Nacionalidad Italiano

Objeto 3

Nombre: Casco de Casey Stoner

Descripción: Se trata del casco desarrollado por Jorge Lorenzo y Casey Stoner en el campeonato del Mundo de MotoGP. Las 3 tallas de calota ultra resistente, hacen que se adapte en peso y volumen al usuario. La nueva pantalla X-802 ofrece un amplio campo de visión. Copyright Motocard.com. Dispone de un mecanismo de doble acción: permite bloquear la visera para evitar su apertura accidental y dispone de una posición entreabierta para circular a baja velocidad. Acolchados interiores Unitherm muy transpirables, totalmente desmontables y lavables. Dispone de un spoiler trasero ajustable. Ventilación dinámica con entradas de aire en la zona frontal y extractores en la zona posterior. Cierre con doble anilla. Se vende con pantalla clara.

Tipo: tipo 3

Propiedades:

Precio 371

Referencia Ref. M-01906610

URL www.nolan.it/catalogo_10.jsp?itemtype=2

PARTE 3. DESARROLLO DE UNA APLICACIÓN INTERACTIVA PARA VIDEO EN INTERNET USANDO MÉTODOS TRADICIONALES

El siguiente formato fue definido para realizar una estimación de tiempos del proceso de desarrollo de la aplicación usando HTML y JavaScript. Si existe alguna actividad que

usted piense que no se encuentra en esta lista, le puede asignar el tiempo en el apartado Otros.
Puede suponer que cuenta con las herramientas y los recursos de hardware y software que considere necesarios.
Los tiempos deben ser dados en horas (p.e, 1, 2.5) y equivalentes al trabajo de una sola persona.

Análisis y Diseño

Implementación*

Pruebas

Otros

En caso de haber escrito un valor en el campo Otros por favor especifique a que tareas corresponde

¿Si la aplicación se tuviera que migrar a otra plataforma de televisión interactiva cuanto tiempo se requeriría?*

PARTE 4.DESARROLLO DE UNA APLICACIÓN INTERACTIVA PARA VIDEO EN INTERNET USANDO ACTIVA

Ahora para la misma especificación dada en el numeral 2 diseñe y genere la aplicación usando acTiVa y luego responda la siguientes preguntas.

Califique los siguiente enunciados de acuerdo a la siguiente escala:* Siendo 1. Completamente en desacuerdo y 5. Completamente de acuerdo

	1	2	3	4	5
Es fácil seleccionar objetos					
Es fácil enriquecer los objetos					
Es fácil realizar el seguimiento de objetos					
Es fácil generar la aplicación					
En general, estoy satisfecho con la facilidad para completar las tareas					
En general, estoy satisfecho con el tiempo que me tomo completar las tareas					
Es necesario aprender muchas cosas					
Las operaciones están estructuradas con coherencia					
El tiempo de desarrollo usando acTiVa se reduce significativamente si se compara con un desarrollo tradicional					
La reproducción del video y la selección de objetos son fluidas					
Los algoritmos de seguimiento redujeron la selección de objetos significativamente					
La interfaz es compleja					
Me siento satisfecho con la aplicación resultante de usar acTiVa					

¿Cual es la peor característica de la aplicación, y por qué?

¿Qué es lo mejor de la aplicación, y por qué?

Otros Comentarios o sugerencias

Puede contener pero no limitarse a: mejoras, funciones nuevas o eliminación de funciones

GRACIAS POR PARTICIPAR. POR FAVOR VERIFIQUE LA INFORMACIÓN QUE VA A ENVIAR.

ANEXO E CATEGORIZACIÓN DE VARIABLES

Para una mejor comprensión de los datos recogidos para los algoritmos de *Template Matching* y SURF se categorizaron las variables. Las categorías se definieron sacando el valor correspondiente al 20%.

Tabla 20 Categorización de las variables

Categoría	**Intervalo *Template Matching* Tiempo1**	**Intervalo SURF Tiempo2**
1	<=10	<=37
2	11-12	38-54
3	13-15	55-82
4	16-19	83-121
5	>=20	>=122

Tabla 21 Frecuencia para Tiempo1

Categoría	**Frecuencia**	**Frecuencia Relativa**	**Frecuencia Acumulada**	**Frecuencia Relativa Acumulada**
1	231	0,2786	231	0,2786
2	102	0,1230	333	0,4017
3	196	0,2364	529	0,6381
4	154	0,1858	683	0,8239
5	146	0,1761	829	1,0000

Tabla 22 Frecuencia para Tiempo2

Categoría	Frecuencia	Frecuencia Relativa	Frecuencia Acumulada	Frecuencia Relativa Acumulada
1	172	0,2075	172	0,2075
2	159	0,1918	331	0,3993
3	168	0,2027	499	0,6019
4	165	0,1990	664	0,8010
5	165	0,1990	829	1,0000

ANEXO F TABULACIÓN DE TIEMPOS DEL EXPERIMENTO

La siguiente tabla contiene la tabulación de los tiempos tomados (en segundos) durante la segunda parte del experimento.

Tabla 23 Tabulación de tiempos del experimento con la arquitectura

Participante	Dibujo 1	Enriquecimiento 1	Seguimiento 1	Dibujo 2	Enriquecimiento 2	Seguimiento 2	Dibujo 3	Enriquecimiento 3	Seguimiento 3	Generación App	Pruebas App
1	7	147	22	11	95	17	4	82	4	28	350
2	41	288	47	37	78	11	20	79	8	39	106
3	22	109	12	24	99	5	73	41	5	34	185
4	14	134	15	40	70	6	35	55	5	12	165
5	8	127	4	56	83	3	10	78	9	38	251
6	29	221	6	6	66	14	12	79	3	19	118