Universidad Autónoma de Querétaro
Facultad de Informática
Maestría en Software Embebido

Automatización de pruebas funcionales de Bluetooth AVRCP 1.0 en un sistema de entretenimiento para vehículos automotrices

TESIS

Que como parte de los requisitos para obtener el grado de
Maestro en Software Embebido

Presenta:
Carlos Octavio Mendívil Vázquez

Dirigido por:
M. en A. Juan José Gómez Cornejo

SINODALES

M.A. Juan José Gómez Cornejo Presidente	Firma
M.C. José Arturo Gaona Cuadra Secretario	Firma
M.C. Raúl Villarreal Sánchez Vocal	Firma
M.C. Ruth Angélica Rico Hernández Suplente	Firma
Dr. Ubaldo Chávez Morales Suplente	Firma

M. en C. Ruth Angélica Rico Hernández
Directora de la Facultad

Dra. Ma. Guadalupe Flavia Loarca Piña
Directora de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Mayo 2015
México

RESUMEN

El presente trabajo de tesis tiene por objeto proveer una librería de software que permita desarrollar pruebas funcionales automatizadas de los comandos de AVRCP 1.0 del protocolo de Bluetooth, empleados en el escenario de transmisión de audio entre un dispositivo móvil y un sistema de entretenimiento para vehículos automotrices. Se diseñaron una serie de casos de prueba funcionales de acuerdo a los requisitos de usuario contenidos en las especificaciones de AVRCP 1.0. Posteriormente, se desarrolló una librería de software, bajo el esquema del modelo en “V”. Después, se empleó la librería desarrollada para implementar pruebas funcionales automatizadas. Además, como resultado del proceso de desarrollo seguido, se obtuvo una librería actualizable y orientada al mantenimiento. Finalmente, se concluye que es necesario que el ingeniero de pruebas, adicionalmente a las competencias relacionadas a las pruebas de software, posea conocimientos de diseño de software para tener las habilidades requeridas para diseñar y desarrollar librerías que permitan ejecutar pruebas automatizadas.

(**Palabras clave**: pruebas, automatización, funcional, AVRCP, Bluetooth)

SUMMARY

This thesis aims to provide a software library that allows to develop automated functional tests of the Bluetooth protocol AVRCP 1.0 commands, Used for audio streaming applications between a mobile device and an entertainment system for automotive vehicles. A series of functional test cases were designed according to user requirements contained in the specifications for AVRCP 1.0. Subsequently, a software library was developed under the scheme of the "V" model. Then, this library was used to implement the automated functional testing. Also, as a result of the development process, an updatable and maintainable library was obtained. Finally, it has been concluded that the test engineer requires, in addition to the skills related to software testing, skills to design and develop libraries that allow to perform automated testing.

(**Key words**: testing, automation, functional, AVRCP, Bluetooth)

Dedicado a mi Familia:

Mi Mamá, Juanita Vázquez García

Mi Papá, Carlos Octavio Mendívil Gámez

Mi Hermanita, Dulce María Mendívil Vázquez

AGRADECIMIENTOS

Doy gracias a Dios por todas las bendiciones que me ha otorgado a lo largo de mi vida. También, doy gracias a la Virgen de Guadalupe por su protección de madre.

Quiero agradecer a mis papás y mi hermana por el amor, el apoyo y la confianza incondicional que me han dado.

Agradezco a mi director de tesis M. en .A. Juan José Gómez Cornejo por sus atinados comentarios, apreciables consejos y tiempo que me ha brindado para la culminación de este proyecto. También, agradezco a mis profesores M. en C. José Arturo Gaona Cuadra y M. en C. Raúl Villarreal Sánchez por el conocimiento y las enseñanzas que han compartido conmigo para el desarrollo de esta tesis.

De igual manera, agradezco a mis profesores, amigos y compañeros que me han asesorado y acompañado durante mis estudios de posgrado.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1 IMPORTANCIA DEL TEMA Y DEFINICIÓN DEL PROBLEMA

Bluetooth es un protocolo muy comercial, pues ofrece una red comunicación inalámbrica de área personal, con un alcance de conexión de aproximadamente 10 metros. Pero, aunque es respaldado por el “Grupo de Interés Especial” de Bluetooth (SIG por sus siglas en Inglés), integrado por grandes compañías tecnológicas, tiene tendencia a presentar discrepancias entre dispositivos de distintos modelos y marcas.

Para el usuario moderno, la transmisión de audio desde un dispositivo móvil hacia el sistema de entretenimiento del automóvil, a través de Bluetooth, es una de las características más utilizadas (Kern & Schmidt, 2009). Por lo que, la funcionalidad de Bluetooth sufre una metódica y exhaustiva etapa de pruebas durante la vida del producto. El problema en estas pruebas es que la gran variabilidad de dispositivos móviles y la curva de aprendizaje causada por la inexperiencia de los ingenieros para llevar a cabo las pruebas de Bluetooth, puede generar pruebas manuales prolongadas y tediosas.

Por lo descrito anteriormente, se entiende que la automatización de las pruebas de Bluetooth AVRCP presenta una gran oportunidad y reto para los ingenieros que desarrollan sistemas de entretenimiento para vehículos automotrices. Además, se puede generar un valor agregado para el consumidor, pues la funcionalidad de Bluetooth representa una característica que puede marcar la diferencia entre los competidores dentro de la gama automotriz.

1.2 OBJETIVOS

1.2.1 Objetivo General:

Proveer una librería de software que permita desarrollar pruebas funcionales automatizadas de los comandos del perfil de “Control Remoto de Audio/Video versión 1.0” (AVRCP por sus siglas en Inglés) del protocolo de Bluetooth, empleados en el escenario de transmisión de audio entre un dispositivo móvil y un sistema de entretenimiento para vehículos automotrices, que se enumeran a continuación:

- Iniciar Reproducción, el cual será denominado como “Reproducir” en el resto del documento.
- Detener Reproducción, el cual será denominado como “Detener” en el resto del documento.
- Reproducir la siguiente pista, el cual será denominado como “Siguiente” en el resto del documento.
- Reproducir la pista anterior, el cual será denominado como “Previo” en el resto del documento.

1.2.2 Objetivos Particulares:

- Analizar los requisitos funcionales de usuario, de acuerdo al escenario de Control Remoto Mutuo, contenidos en el documento de especificaciones del perfil de control de AVRCP de Bluetooth.

- Diseñar los casos de prueba con base en a los requisitos funcionales de usuario.

- Definir los casos de uso de los siguientes comandos en el escenario de transmisión de audio entre un dispositivo móvil y un sistema de entretenimiento para vehículos automotrices:
  - “Reproducir”
  - “Detener”
  - “Siguiente”
  - “Previo”

- Diseñar las pruebas funcionales con base en a los casos de uso de los comandos de AVRCP 1.0.

- Desarrollar una librería de software que permita automatizar los casos de pruebas especificados anteriormente:

  - Definir los requisitos de la librería.
  - Diseñar la arquitectura.
  - Implementar funciones.
  - Realizar pruebas de la librería.

- Implementar la automatización de los casos de pruebas diseñados previamente.

1.3 ALCANCE

En el proceso de transmisión de audio por medio de Bluetooth se involucran diversos “perfiles” o interfaces de Bluetooth definidos por el SIG, pero de acuerdo al alcance de esta tesis no se analizará el protocolo de Bluetooth ni tampoco se llevará a cabo un análisis profundo de ningún perfil de Bluetooth. El alcance de ésta tesis, se centrará en la propuesta de una librería de software para automatizar casos de pruebas funcionales de Bluetooth, así como el diseño de una serie de pruebas automatizadas de Bluetooth que permitan la detección de las

fallas originadas en la transmisión de audio por medio del perfil AVRCP 1.0 del protocolo de Bluetooth que existen entre un sistema de entretenimiento automotriz y un dispositivo móvil.

Ésta Tesis se limitará a realizar un análisis de los requisitos funcionales de usuario, de acuerdo al escenario de Control Remoto Mutuo, contenidos en el documento de especificaciones del perfil de “Control de Audio/Video Remoto” versión 1.0 (AVRCP por sus siglas en Inglés) de Bluetooth. Con base en dicho análisis, se diseñarán los casos de pruebas funcionales que serán automatizados empleando la librería de software desarrollada como resultado final del presente proyecto.

Posteriormente, se definirán los requisitos necesarios para diseñar una herramienta de software que posibilite automatizar los casos de pruebas. Luego, se realizará un prototipo de la herramienta, se verificará contra las especificaciones definidas anteriormente. Por último, se implementarán los casos de prueba de manera automatizada mediante la librería.

1.3.1 Resultado que se pretende obtener con el proyecto

Como resultado de esta tesis se espera obtener una librería de software que permita implementar una serie de pruebas funcionales automatizadas que posibiliten la detección de las fallas funcionales originadas en el protocolo AVRCP 1.0 de Bluetooth en el desarrollo de un sistema de entretenimiento automotriz.

2. MARCO TEORICO

2.1 Pruebas de Software

Las pruebas de software son un conjunto de actividades o tareas llevadas a cabo con los objetivos de encontrar errores (Myers, Sandler, & Badgett, 2011), verificar que satisface los requisitos especificados (Li & Wu, 2006) e identificar fallas (Homès, 2013).

Las pruebas de software deben enfocarse en proveer información acerca del producto bajo prueba y encontrar tantos defectos como sea posible, tan temprano como sea posible en el proceso de desarrollo, bajo restricciones determinadas de costo y tiempo ("Software and systems engineering Software testing Part 1:Concepts and definitions", 2013).

2.1.1 Términos Básicos

A continuación se enlistan algunos de los términos básicos, empleados en el campo de las pruebas de software, de acuerdo al estándar ISO/IEC/IEEE 24765 ("Systems and software engineering – Vocabulary", 2010):

- **Prueba:** Actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, los resultados son observados o grabados, y una evaluación es hecha sobre algún aspecto en particular del sistema o componente.
- **Pruebas de Software:** Verificación dinámica del comportamiento de un programa con base en un conjunto finito de casos de prueba.
- **Caso de Prueba:** Conjunto de entradas de prueba, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular, así como, ejercitar un flujo o camino de ejecución en particular de un programa o para verificar el cumplimiento con un requerimiento específico.
- **Error:** Una acción humana que produce un resultado incorrecto, así como una falta contenida en un software.

- **Falla:** Imposibilidad de un producto para ejecutar una función requerida o su inhabilidad para desempeñarse con respecto a los límites especificados previamente.
- **Falta:** Una manifestación de un error en software.
- **Defecto:** Un problema, el cual sí no es corregido, puede provocar que una aplicación falle o que produzca resultados incorrectos.
- **Verificación:** El proceso de evaluar un sistema o componente para determinar sí los productos de una fase de desarrollo especifica satisface las condiciones impuestas al inicio de dicha fase.
- **Validación:** Confirmación, a través de evidencia objetiva, de que los requisitos, para un uso específico o aplicación, han sido cumplidos.
- **Ciclo de Desarrollo de Software:** Periodo de tiempo que inicia con la decisión de desarrollar un producto de software y termina cuando el software es entregado.
- **Proceso de Desarrollo de Software:** Proceso mediante el cual las necesidades del usuario son convertidas en un producto de software.

2.1.2 Importancia de las pruebas de Software

De manera general, es aceptado que no es posible crear software perfecto. Por lo tanto, es necesario probar el software antes de liberarlo a los usuarios, con el fin de reducir el riesgo de que errores en el software de producción provoquen un impacto negativo cuando el software sea utilizado. Se pueden derivar serias complicaciones como resultado, como por ejemplo: se puede comprometer la reputación y/o sustentabilidad económica del negocio, así como, la seguridad del usuario y/o del medio ambiente.("Software and systems engineering Software testing Part 1:Concepts and definitions", 2013)

El Instituto Nacional de Estándares y Tecnología de Estados Unidos (NIST por sus siglas en inglés), en el 2002, llevó a cabo una investigación con el propósito de estimar el impacto económico del uso inadecuado de los métodos y

herramientas de pruebas de software, cuyo resultado obtenido es un impacto que tiene un rango desde $22.2 a $59.5 billones de dólares (Tassey, 2002).

2.1.3 Relación entre Pruebas de Software y Aseguramiento de Calidad de Software

De acuerdo al estándar internacional ISO/IEC/IEEE 29119-1 de pruebas de software ("Software and systems engineering Software testing Part 1:Concepts and definitions", 2013), el aseguramiento de calidad consiste en una serie de procesos y actividades, planeadas sistemáticamente, requeridas para suministrar la confianza adecuada de que un proceso o producto cumplirán con los requisitos de calidad establecidos. Mientras que, los resultados de las actividades de pruebas, son empleados por el proceso de aseguramiento de calidad, para medir el cumplimiento de las características de calidad del proceso o producto bajo prueba.

2.1.4 Tipos de Pruebas de Software

Dependiendo del criterio y el contexto, en la literatura existen múltiples clasificaciones con respecto a la pruebas de software.

2.1.4.1 Pruebas Estáticas y Dinámicas

Sí las pruebas son llevadas a cabo sin que el sistema bajo prueba sea ejecutado o no, son conocidas respectivamente como pruebas dinámicas o estáticas. Dichas pruebas a su vez, según el estándar internacional ISO/IEC/IEEE 29119-1 ("Software and systems engineering Software testing Part 1:Concepts and definitions", 2013) se clasifican de la siguiente manera:

- Pruebas Estáticas: Emplean técnicas que buscan defectos en el software sin que éste sea ejecutado, como pueden ser inspecciones, revisiones, análisis estático, entre otros. No se considera necesario describir éstas técnicas ya que no serán objeto de ésta tesis.
- Pruebas Dinámicas: De manera contraria, las pruebas dinámicas, utilizan métodos que llevan a cabo las pruebas en el software durante su ejecución. Además, incluyen la preparación y el seguimiento de las actividades. Estas pruebas se pueden clasificar a su vez en:
    - Pruebas Basadas en Especificaciones o Funcionales: Son conocidas también como pruebas de caja negra, son diseñados sin tener conocimiento de la estructura o flujo interno del código. Su objetivo es evaluar sí el comportamiento observado del software bajo prueba, cumple o no, con las especificaciones o requisitos funcionales (Beizer, 1990).
    - Pruebas Basadas en Estructura: Se basan directamente en el código. Consisten en elegir distintos caminos que puede tomar el software durante la ejecución de las pruebas (Whittaker, 2000).
    - Basadas en Experiencia: Se sustentan en experiencia de pruebas previas, conocimiento de un sistema o software en particular, métricas de proyectos pasados.

2.1.4.2 Pruebas según la fase o nivel del ciclo de desarrollo de software

Como cualquier otro producto, el software tiene un ciclo de vida determinado, desde su conceptualización inicial hasta su eventual liberación. Dentro del mismo, se pueden identificar dos principales aspectos: el desarrollo y el mantenimiento de software. Los tres modelos de desarrollo de software más populares son (González, Mayorga, Prado, & Bedoya, 2013) :

- Modelo en Cascada.
- Modelo en Espiral.
- Modelo en V.

El modelo tipo V se caracteriza por tomar en cuenta la verificación y validación del producto a diferentes niveles de abstracción, es decir, hace énfasis en la relación que existe entre cada etapa de planeación y diseño con respecto a la etapa correspondiente de pruebas. En la Figura 1 se muestra un diagrama con todas las etapas del modelo de desarrollo tipo “V”.

Las etapas de pruebas del modelo de desarrollo tipo “V” se enumeran a continuación (Mathur & Malik, 2010):

- Pruebas de Unidad
- Pruebas de Integración
- Pruebas de Sistema
- Pruebas de Aceptación

Las pruebas de integración tienen como objetivo probar los componentes de software integrados en un paquete completo, sustentándose en pruebas basadas en especificaciones funcionales (Mathur & Malik, 2010).

El presente trabajo se centrará en las pruebas concernientes a la etapa de pruebas de integración. Por lo que no considera necesario abordar el resto de las etapas de pruebas del modelo V.

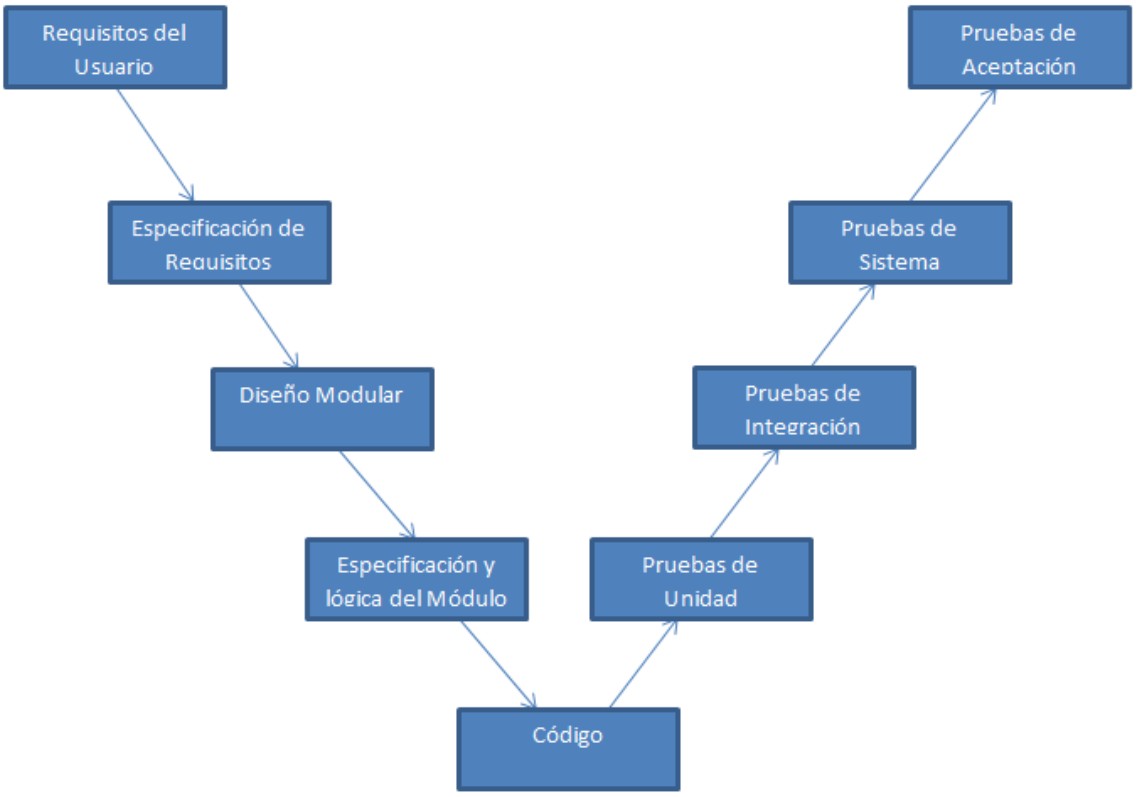

Figura 1 - Modelo de desarrollo tipo "V"

2.1.5 Técnicas de diseño de Pruebas Basadas en Especificaciones

De acuerdo a la norma ISO ("IEEE Draft International Standard for Software and Systems Engineering–Software Testing–Part 4: Test Techniques", 2014) las técnicas empleadas para diseñar pruebas basadas en especificaciones son las siguientes:

Partición de Equivalencia: Consiste en clasificar las entradas y salidas del sistema bajo prueba en partes equivalentes o clases equivalentes, donde cada clase equivalente debe ser definida como una condición de prueba. Cada clase

equivalente es elegida de tal manera que todos los valores contenidos respectivamente en cada clase equivalente, deben ser tratados de manera similar por el sistema bajo prueba.

Método de Clasificación de Árbol: Utiliza un modelo del sistema bajo prueba que clasifica las entradas del mismo y las representa gráficamente en forma de árbol. Es similar al proceso de partición de equivalencia, la diferencia, es que en la clasificación de árbol, las clasificaciones no deben traslaparse.

Análisis del valor límite: Emplea un modelo que clasifica las entradas y salidas del sistema bajo prueba en una serie de conjuntos ordenados con límites identificables, donde los límites de cada conjunto son las condiciones de prueba.

Pruebas de Sintaxis: Consiste en un modelo de la sintaxis de las entradas del sistema bajo prueba que son representadas como un número de reglas, donde cada regla define el formato de un parámetro de entrada para el sistema bajo prueba en términos de secuencias, iteraciones o selecciones entre los elementos de sintaxis.

Pruebas Combinatorias: Son utilizadas para derivar sistemáticamente un subconjunto manejable de casos de pruebas mediante una serie de combinaciones definidas en términos de los parámetros del sistema bajo prueba, así como, los valores que estos parámetros pueden tomar.

Tablas de Decisión: Utiliza un modelo de relaciones lógicas, o reglas de decisión, entre condiciones o causas y acciones o efectos para el sistema bajo prueba en forma de una tabla de decisiones. Cada condición es una entrada o combinación de entradas para el sistema bajo prueba, expresadas en forma binaria, mientras que, una acción o efecto, es una salida o una combinación de salidas para el sistema bajo prueba, expresadas de igual manera, en forma binaria.

Grafo de Causa-Efecto: Emplea un modelo de relaciones lógicas entre causas y efectos para el sistema bajo prueba en la forma de un grafo causa-

efecto. Cada causa es una entrada o una combinación de entradas, para el sistema bajo prueba, expresada en forma de dato lógico. De igual manera, cada efecto es una salida o combinación de salidas expresadas en forma de dato lógico.

Estados de Transición: Se representa el sistema bajo prueba en forma de los estados que puede ocupar, las transiciones entre los mismos, los eventos que causan las transiciones y las acciones que pueden resultar de las transiciones.

Pruebas de Escenario: Consiste en una serie de interacciones secuenciales entre el sistema bajo prueba y uno o más actores, los cuales a su vez, pueden ser usuarios y/u otros sistemas (Hsia et al., 1994). Una forma muy común de pruebas de escenario es el método de pruebas de caso de uso, donde un modelo de caso de uso del sistema bajo prueba es empleado para describir, como el sistema bajo prueba interactúa con uno o más actores, con el propósito de probar secuencias de interacciones.

2.2 Diseño de Experimentos

Según la literatura de pruebas de software revisada para llevar a cabo este trabajo, el Diseño de Experimentos (DOE por sus siglas en inglés), es un método eficiente que proporciona una alta cobertura del dominio de las entradas con un relativamente pequeño conjunto de pruebas (Brownlie, Prowse, & Phadke, 1992; Burroughs, Jain, & Erickson, 1994; Cohen, Dalal, Parelius, & Patton, 1996; Dunietz, Ehrlich, Szablak, Mallows, & Iannino, 1997; McGregor & Sykes, 2001; Pressman, 2001; Kuhn & Reilly, 2002).

Dentro de la teoría de DOE, un factor es una variable que afecta potencialmente la respuesta del sistema bajo prueba, mientras que los niveles son los valores que el factor puede tomar. Entre los distintos modelos empleados para llevar a cabo un DOE, se encuentran los diseños factoriales, los cuales, pueden ser completos o fraccionados (Antony, 2014). Los primeros contemplan todas las posibles

combinaciones de niveles para todos los factores, mientras que los segundos, sólo consideran una fracción de dichas combinaciones.

No se considera necesario explicar más a detalle la teoría del DOE ya que no es objeto del presente trabajo.

2.3 Análisis de Modos y Efectos de Falla de Software

El estándar de ISO/IEC/IEEE 24765 ("Systems and software engineering – Vocabulary", 2010) define el Análisis de Modos y Efectos de Falla (AMEF) como “Un procedimiento analítico en el cual, cada modo potencial de falla en cada componente de un producto es analizado para determinar su efecto en la confiabilidad del producto o sistema y en la requerida funcionalidad del componente” . De igual manera también define un Modo de Falla como “La manifestación física o funcional de una falla”.

El AMEF es principalmente empleado para análisis de confiabilidad de hardware, pero de manera similar, puede ser utilizado como una técnica sistemática para identificar posibles modos de falla en el campo de desarrollo de software (Nguyen, 2001). En lo que compete a este trabajo, el AMEF será realizado con el objetivo de complementar a las pruebas funcionales, mediante la identificación de los modos de fallas que no son considerados por los requisitos funcionales.

Existen diversas versiones y estándares industriales para llevar a cabo el AMEF, según las particularidades de su producto, por mencionar algunos, la industria aeroespacial emplea el MIL-STD-1629ª, mientras que, las compañías automotrices como Chrysler, Ford y General Motors usan el SAE J-1739. Particularmente el SAE J-1739, ha agregado el concepto de Número de Prioridad de Riesgo (RPN por sus siglas en inglés), el cual está compuesto por tres métricas: Severidad, Ocurrencia y Detección (Haapanen & Helminen, 2002).

Como se mencionó anteriormente existen distintas versiones de AMEF, por lo cual con respecto a este trabajo se utilizará el formato ilustrado en la Tabla 1.

Id.	Funcionalidad	Modo de Falla	Causa Potencial de la Falla	Efecto de la Falla	Acciones a Tomar	Sev.	Ocu.	Det.	RPN

Tabla 1 - Formato AMEF

No se considera necesario explicar más a detalle la teoría del AMEF pues no es objeto del presente trabajo.

2.4 Software Embebido

El Software Embebido forma parte de un sistema más grande y lleva a cabo algunos de los requisitos del sistema del que forma parte ("Systems and software engineering – Vocabulary", 2010), dicho sistema es conocido como Sistema Embebido, el cual a su vez, es una computadora especializada, con un diseño y objetivos particulares de acuerdo al dispositivo y/o aplicación a la que está destinada(Kang, Kwon, & Lee, 2005). Los sistemas embebidos ocupan un rol muy importante en distintos campos industriales como el automotriz, aeronáutico, electrodoméstico, consumo electrónico en general, entre otros (Graaf, Lormans, & Toetenel, 2003).

Los componentes de hardware y software en un sistema embebido están estrechamente relacionados, de los cuales, la parte de software corresponde de un 10% a un 20% del sistema, sin embargo, la creciente complejidad y la sustitución de algunas funciones de hardware mediante software, han ocasionado que más del 80% de las fallas sean originadas por el software (Seo, Ki, Choi, & La, 2008).

2.4.1 Software Embebido Automotriz

El Software Embebido en la industria automotriz, representa un papel muy importante, debido al creciente aumento de las funcionalidades eléctricas y electrónicas desempeñadas por software (Jo, Piao, Cho, & Jung, 2008). Particularmente, para el mercado de sistemas embebidos en el sector automotriz, se espera un crecimiento del 35% para el 2020 (Kumar, Fernando, & Panicker, 2013).

A continuación se enlistan las características del Software Embebido Automotriz (Kasoju, Petersen, & Mäntylä, 2013):

- **Subsistemas Heterogéneos:** Los automóviles, actualmente, están integrados por un conjunto de distintos tipos de sistemas con el objetivo de coexistir y operar como parte de un sistema más grande, dichos sistemas pueden ser: mecánicos, eléctricos, electrónicos, software, etc. Sus propósitos pueden ser muy variados, como por ejemplo: confort para el pasajero, seguridad, control del chasis, multimedia, telemática, interface humano-computadora, entre otros.
- **Unidades Organizacionales Claramente Divididas**: Históricamente, la industria automotriz se ha organizado de manera vertical, pues cada unidad es responsable de diferentes componentes del automóvil. Con respecto al software, esta manera de trabajar es mucho más compleja, debido a la integración de los sistemas.
- **Distribución del software**: Componentes y funciones mecánicas que anteriormente no se encontraban relacionados, por ejemplo, la interacción de los sistemas de manejo con los de confort y entretenimiento; ahora se ven afectadas por los canales de comunicación que existen entre las unidades funcionales, así como entre los múltiples sistemas embebidos en el automóvil.

- **Sistemas altamente configurables:** Según Kasoju (Kasoju et al., 2013), en el 2013, La cantidad de accesorios electrónicos en un automóvil era de 80, los cuales tienen que ser soportados por el software. Además, las configuraciones cambian con el paso del tiempo, lo que puede requerir diferentes versiones de software.
- **La presión de los costos y el enfoque en costos basados en unidades modelo:** La industria automotriz se basa en costos basados en unidades modelos, por lo que es común que se mejore el costo por unidad mediante el desarrollo de software para empleo de un procesador y/o memoria con recursos limitados, lo cual puede llevar a problemas posteriores en la extensión o mantenimiento del mismo software.

2.5 Sistemas de Entretenimiento para vehículos Automotrices

Un sistema de entretenimiento provee una serie de funcionalidades que pueden variar, desde reproductores multimedia hasta asistencia al conductor, como por ejemplo navegación con GPS (Tchankue, Wesson, & Vogts, 2011).

Las siguientes son las funcionalidades más comunes que se pueden encontrar en un sistema de entretenimiento:

- Navegación
- Entretenimiento en asiento posterior (películas, juegos, televisión, redes sociales, etc.)
- Reproductores multimedia
- Servicios basados en la ubicación
- Conectividad interna a dispositivos móviles
- Conectividad a Internet y comunicaciones externas

Actualmente el automóvil, además de ser un medio de transporte, es considerado un entorno de entretenimiento. La mayoría de los usuarios pasan alrededor de una hora diaria en su vehículo, ya sea como pasajeros o

conductores, lo que ha originado sistemas de entretenimiento con el fin de hacer más placentero el viaje y evitar el aburrimiento (Kern & Schmidt, 2009).

Ha crecido tanto el interés del usuario en éstos sistemas, que las compañías automotrices compiten por producir automóviles con sistemas de entretenimiento más eficientes y útiles, mejorando el estilo de vida del mismo usuario (Sharma & Ramani, 2009).

2.6 Sistemas de Entretenimiento y la conectividad Bluetooth

2.6.1 Bluetooth

Bluetooth es una tecnología (protocolo de comunicación) inalámbrica con el objetivo de conectar dispositivos que se encuentran a corta distancia, mejor conocida como Área de Red Personal (PAN por sus siglas en Inglés). Los dispositivos que incorporan este protocolo pueden comunicarse entre ellos cuando se encuentran dentro de su alcance. Las comunicaciones se realizan por radiofrecuencia, de forma tal que, los dispositivos no tienen que estar alineados y pueden incluso, estar en habitaciones separadas sí la potencia de transmisión es suficiente (Haartsen, Naghshineh, Inouye, Joeressen, & Allen, 1998).

Dependiendo de la aplicación que se está desarrollando por medio de Bluetooth, existe una gran variedad de protocolos conocidos como “perfiles”, entre los cuales se pueden mencionar los siguientes:

- LMP
- AVRCP
- L2CAP
- A2DP
- PBAP
- SDP
- RFCOMM

- BNEP
- AVCTP
- AVDTP
- TCS
- Entre otros

2.6.2 Aplicación de la tecnología de Bluetooth en Sistemas de Entretenimiento para vehículos automotrices

Como se mencionó anteriormente, una de las funciones más comunes dentro de un sistema de entretenimiento es la conectividad Bluetooth con un dispositivo móvil. Esta interacción ha surgido como necesidad de proporcionar al conductor un medio para hablar por teléfono en la modalidad conocida como “Manos Libres”, pero también se ha utilizado el celular como unidad de reproducción de medios, como por ejemplo audio y video.

Uno de los perfiles utilizados para éste propósito es el AVRCP.

2.6.3 Bluetooth AVRCP

El protocolo o “perfil” de AVRCP define las funcionalidades y los procedimientos requeridos para enlazar dispositivos con Bluetooth, con el fin de controlar la transmisión de Audio/Video de un dispositivo a otro. De acuerdo al documento de especificaciones del perfil de AVRCP (Bluetooth SIG, 2012), existen las versiones 1.0, 1.3, 1.4 y 1.5, y en cualquiera de ellas, los dispositivos pueden representar uno de los siguientes dos roles:

El controlador: Es el dispositivo que envía los comandos para controlar la transmisión de Audio/Video. Por ejemplo: una computadora, unos audífonos o un sistema de entretenimiento para vehículos.

El objetivo: Es el dispositivo que recibe los comandos de control y genera una respuesta de Audio/Video. Por ejemplo: Un reproductor de audio o un dispositivo móvil.

Ésta tesis se delimitará a las aplicaciones de la versión 1.0 de AVRCP

2.6.4 Control de Audio en sistemas entretenimiento mediante AVRCP 1.0

Entre un sistema de entretenimiento para vehículos automotrices y un dispositivo móvil, el protocolo AVRCP 1.0 es empleado en modo de transmisión de audio por medio de Bluetooth. Las funciones o comandos de control de la reproducción de audio son los siguientes (LaRussa & Gabel, 2010):

- “Reproducir”: Comando para iniciar la reproducción del audio
- “Detener”: Comando para detener la reproducción
- “Siguiente”: Comando iniciar la reproducción de la siguiente pista de audio
- “Previo”: Comando para reproducir la pista de audio anterior.

2.7 Pruebas de Software Embebido en un Sistema de Entretenimiento para Vehículos Automotrices

Por lo general, el software embebido requiere un conjunto de pruebas exhaustivas antes de que los desarrolladores lo pongan en el mercado. El objetivo de las pruebas funcionales es evitar que el producto llegue con fallas funcionales al consumidor final, por lo que, en la mayoría de los casos, se considera que las pruebas funcionales son de gran importancia para el proyecto, de igual manera,

las pruebas funcionales de integración representan un gran reto (Tsai, Yu, Zhu, & Paul, 2005).

Debido a que el software y el hardware se encuentran estrechamente ligados, no pueden ser probados de manera parcial (Seo et al., 2008). Por lo tanto, las pruebas de software embebido son ejecutadas generalmente con el sistema completo.

2.7.1 Pruebas Funcionales de Bluetooth AVRCP 1.0

Las pruebas funcionales de Bluetooth AVRCP 1.0 consisten en ejecutar los comandos de control de audio con el objetivo de verificar que el comportamiento del sistema de entretenimiento automotriz corresponda al comportamiento especificado por los requisitos funcionales de dicha funcionalidad. A continuación se enlistan los comandos involucrados:

- “Reproducir”: Se verifica que inicie la reproducción de los pistas del dispositivo móvil.
- “Detener”: Se verifica que se detenga la reproducción de los pistas del dispositivo móvil.
- “Siguiente”: Se verifica la reproducción de la siguiente pista de audio.
- “Previo”: Se verifica la reproducción de la pista de audio anterior.

2.8 Automatización de pruebas de software

La automatización de pruebas de software, según Meng (Meng, 2011), consiste en el empleo de una herramienta automática con el fin de cumplir con las necesidades de una variedad de pruebas de software , lo cual incluye el manejo y la implementación de las actividades de pruebas. También es considerada como la administración y realización de actividades de pruebas que involucran el desarrollo y ejecución de secuencias de pruebas, así como, la verificación de los requerimientos de prueba (Dustin, Rashka, & Paul, 1999).

Una secuencia de pruebas es un conjunto de códigos o programa que es ejecutado por las herramientas de pruebas. Las tecnologías para crear secuencias de pruebas pueden clasificarse de la siguiente manera (Hanna, El-Haggar, & Sami, 2014):

- Lineales: Consisten en grabar las acciones manuales para ser ejecutadas posteriormente.
- Estructuradas: Están compuestas por estructuras de control y estructuras de llamada. Respectivamente, controlan los diferentes caminos que se pueden ejecutar durante las pruebas y dividen secuencias grandes en secuencias más pequeñas y sencillas de administrar.
- Compartidas: Son secuencias con una tarea específica que pueden ser llamadas por otras secuencias.
- Controladas por datos: Los datos de la pruebas son obtenidos de un archivo externo de la secuencia, en lugar de ser definidos dentro de la misma secuencia.
- Controladas por palabras clave: Previamente, son diseñadas palabras clave que definen las acciones que se realizan durante las pruebas.

Los beneficios de automatizar una prueba con respecto a una prueba manual son: bajo costo para ejecución de la prueba, reusar secuencias de pruebas anteriores para generar nuevas versiones de código y la ejecución de pruebas de alto estrés por un prolongado periodo de tiempo (Collins, Dias-Neto, & Lucena Jr., 2012), entre otros.

Las pruebas automatizadas están basadas en la existencia de pruebas manuales, ya que típicamente, las pruebas de software están acompañadas con muchas operaciones manuales repetitivas, no intelectuales y no creativas, que a

su vez, pueden ser ejecutadas mediante herramientas de automatización, que por ende, pueden disminuir el error humano, permitir que el ingeniero ejecute un mayor número de pruebas en menor tiempo, liberar al mismo de tareas monótonas y en su lugar dedicarse a tareas de mayor complejidad, como por ejemplo enfocarse a niveles más profundos de pruebas (Meng, 2011). Por lo descrito anteriormente, existe una predisposición para omisión en la detección de fallas o generación de errores.

3. METODOLOGÍA Y DESARROLLO

3.1 Fundamentos Teóricos

Se lleva a cabo la investigación de los fundamentos teóricos necesarios para cumplir los objetivos planteados previamente en ésta tesis, la cual, se encuentra descrita en la sección MARCO TEORICO del presente documento.

3.2 Análisis de los Requisitos Funcionales

Se analizaron los requisitos funcionales de los comandos de AVRCP 1.0 “Reproducir”, “Detener”, “Siguiente” y “Previo”; en el contexto de transmisión de audio entre un dispositivo móvil y un sistema de entretenimiento para vehículos automotrices contenidos en el documento de especificaciones para AVRCP 1.0 de Bluetooth.

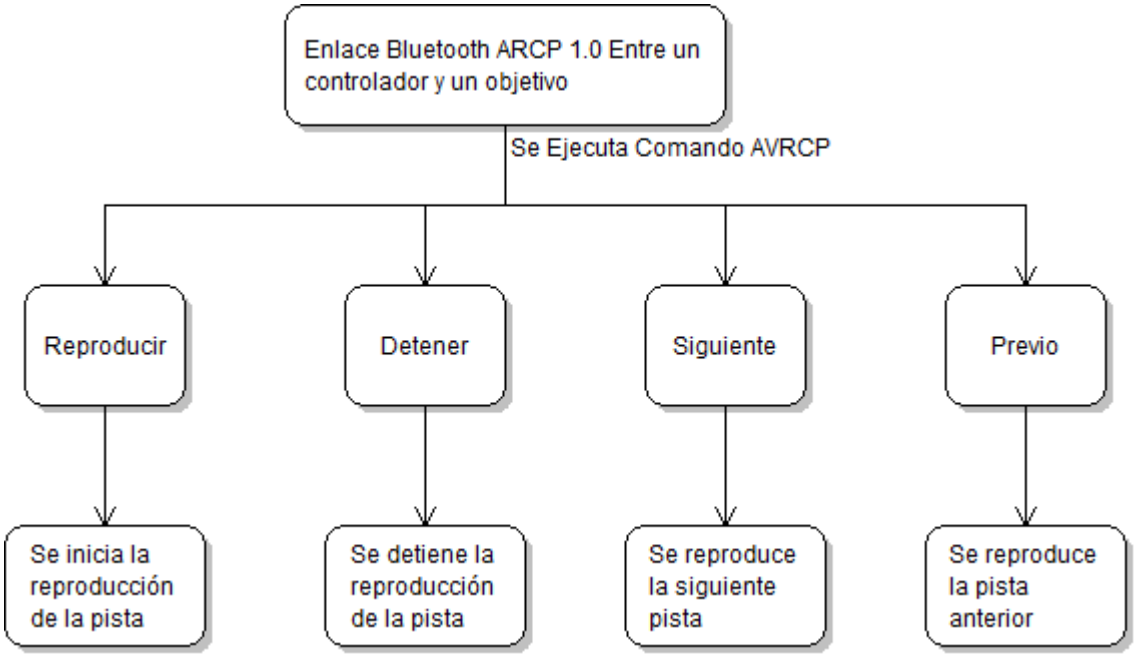

Figura 2 - Diagrama de Árbol de los requisitos funcionales de los comandos "Reproducir", "Detener", "Siguiente" y "Previo" usado

3.3 Diseño de las pruebas funcionales con base en los requisitos de AVRCP 1.0

En el contexto estudiado en ésta tesis, AVRCP permite que el sistema de entretenimiento reaccione funcionalmente a los cambios de estado de reproducción generados desde el dispositivo móvil, permitiendo la ejecución de las pruebas (Bluetooth SIG, 2012). Por lo que, se ha propuesto que los comandos de AVRCP sean ejecutados desde el dispositivo móvil. Se optó por emplear un dispositivo móvil con el sistema operativo Android (Butler, 2011), ya que según los resultados de un estudio llevado a cabo por la empresa de análisis estadístico de mercado de consumo tecnológico "International Data Corporation" (IDC) en el 2014 ("IDC: Smartphone OS Market Share 2014, 2013, 2012, and 2011", 2014), el 84% del mercado de los teléfonos móviles inteligentes usa el sistema operativo Android, particularmente, entre los dispositivos Android, la compañía líder es Samsung con un 23.7%. Por lo tanto, el dispositivo móvil a emplear fue un

Samsung Galaxy S3 Mini con la aplicación de reproducción de audio que tiene de fábrica.

Además, las funcionalidades descritas anteriormente, en el contexto entre un sistema de entretenimiento automotriz y un dispositivo móvil, pueden verse afectadas por la aplicación empleada para reproducir el contenido de audio de acuerdo a las precondiciones de “Tiempo de Reproducción” y “Estado de Reproducción”. Por lo tanto, se ha propuesto incluir dichas precondiciones en el diseño de las pruebas. En la Tabla 2 se pueden ver los efectos de éstas precondiciones en los comandos de AVRCP 1.0.

Precondiciones	Efecto en AVRCP 1.0
Tiempo de Reproducción	Dependiendo de un tiempo “X” de reproducción, el comando de “Previo” puede regresar el tiempo de reproducción al inicio de la pista actual o puede cambiar la reproducción a la pista anterior
Estado de Reproducción	Si la reproducción está en curso o se encuentra detenida puede variar el efecto de los comandos de “Siguiente” y “Previo”, ya que pueden o no iniciar la reproducción, sí ésta estaba previamente detenida.

Tabla 2 - Precondiciones relacionados con los comandos AVRCP 1.0

Con base en el análisis de requisitos realizado anteriormente, las precondiciones involucradas y las particularidades del dispositivo Android empleado en este trabajo, se ha elaborado la tabla de decisiones, representada en la Tabla 3, en la cual, se pueden observar las precondiciones y los resultados esperados, según cada acción.

Precondiciones	Presencia de Precondiciones			
Tiempo MAYOR a X segundos	Si	No	No	Si
Reproducción detenida	No	Si	No	Si
Reproducción en curso	Si	No	Si	No
Tiempo MENOR a X segundos	No	Si	Si	No
Acciones	**Resultado Esperado según la acción y el conjunto de precondiciones**			
"Reproducir"	Reproducción sigue en curso	Inicio de Reproducción en pista actual	Reproducción sigue en curso	Inicio de Reproducción en pista actual
"Siguiente"	Reproducción de Pista siguiente	Reproducción Detenida en Pista siguiente	Reproducción de Pista siguiente	Reproducción Detenida en Pista siguiente
"Previo"	Reproducción de Pista actual desde el inicio de la pista	Reproducción Detenida en Pista anterior	Reproducción de Pista anterior	Reproducción Detenida en Pista anterior
"Detener"	Alto a Reproducción	Reproducción sigue detenida	Alto a Reproducción	Alto a Reproducción

Tabla 3 - Tabla de decisiones

Como resultado de la tabla de decisiones anterior, se detectaron tres factores, un factor de cuatro niveles y dos de dos niveles, enlistados a continuación y descritos en detalle en la Tabla 4:

- Tiempo de Reproducción
- Estado de Reproducción
- Comandos de Ejecutados

Factores	Nivel 1	Nivel 2	Nivel 3	Nivel 4
Tiempo de Reproducción	Tiempo MAYOR a X segundos	Tiempo MENOR a X segundos		
Estado de Reproducción	Reproducción detenida	Reproducción en curso		
Comandos de Ejecutados	“Reproducir”	“Siguiente”	“Previo”	“Detener”

Tabla 4 - Factores y sus respectivos niveles

Los factores descritos anteriormente, fueron utilizados para llevar a cabo el DOE, mediante el software “Minitab” ("Minitab", 2015). Se decidió realizar un diseño factorial completo, ya que los elementos involucrados representan dieciséis diferentes combinaciones, las cuales son mostradas en la Tabla 5. Cabe aclarar,

que para abreviar el nombre de los niveles de algunos de los factores se ha empleado la siguiente nomenclatura, de este punto en adelante:

- **X+** representa “Tiempo MAYOR a X segundos”.
- **X-** representa “Tiempo MENOR a X segundos”.
- **DET** representa “Reproducción detenida”.
- **REP** representa “Reproducción en curso”.
- **X** se deja como variable abierta ya que es diferente para cada dispositivo**.**

Comandos Ejecutados	Estado de Reproducción	Tiempo de Reproducción
"Previo"	DET	X+
"Reproducir"	DET	X+
"Reproducir"	REP	X+
"Previo"	REP	X-
"Siguiente"	REP	X+
"Siguiente"	REP	X-
"Detener"	DET	X-
"Detener"	REP	X+
"Detener"	REP	X-
"Siguiente"	DET	X-
"Previo"	DET	X-
"Detener"	DET	X+
"Reproducir"	DET	X-
"Reproducir"	REP	X-
"Siguiente"	DET	X+
"Previo"	REP	X+

Tabla 5 - DOE Factorial Completo empleado

Finalmente, utilizando la información otorgada por la tabla de decisiones y el DOE, fue posible desarrollar el conjunto de pasos de prueba, precondiciones y resultados esperados para llevar a cabo las pruebas funcionales de AVRCP, las cuales son descritas en la Tabla 6.

Pasos de Prueba	Acciones a Ejecutar	Precondiciones		Resultado Esperado
		Estado de Reproducción	Tiempo de Reproducción	
1	"Reproducir"	DET	X-	Inicio de Reproducción de pista
2	"Reproducir"	REP	X-	Sigue la Reproducción de la pista actual
3	"Siguiente"	REP	X-	Reproducción de la siguiente pista
4	"Previo"	REP	X-	Reproducción de la pista anterior
5	"Detener"	REP	X-	Detiene Reproducción de la pista actual
6	"Detener"	DET	X-	Sigue Detenida la Reproducción en la pista actual
7	"Siguiente"	DET	X-	Inicia Reproducción de la siguiente pista
8	"Detener"	REP	X-	Detiene Reproducción de la pista actual
9	"Previo" y "Reproducir"	DET	X-	Inicia Reproducción de la pista anterior
10	Esperar que el tiempo de reproducción sea X+	REP	X-	Este paso establece una precondición, y se considera terminado hasta que el tiempo de reproducción sea X+
11	"Detener"	REP	X+	Detiene Reproducción de la pista actual
12	"Reproducir"	DET	X+	Inicio de Reproducción de la pista actual
13	"Reproducir"	REP	X+	Sigue la Reproducción de la pista actual
14	"Siguiente"	REP	X+	Reproducción de la siguiente pista
15	"Previo"	REP	X+	Reproducción de la pista actual
16	"Detener"	REP	X+	Detiene Reproducción de la pista actual
17	"Detener"	DET	X+	Sigue Detenida la Reproducción en la pista actual
18	"Siguiente"	DET	X+	Inicia Reproducción de la siguiente pista
19	"Detener"	REP	X+	Detiene Reproducción de la pista actual
20	"Previo" y "Reproducir"	DET	X+	Inicia Reproducción de la misma pista

Tabla 6 - Pruebas Funcionales de ACRCP 1.0

3.3.1 Ejecución y resultados del AMEF

Como se mencionó anteriormente, el AMEF es una actividad complementaria y de valor agregado para las pruebas funcionales diseñadas en el apartado previo 3.3 (Diseño de las pruebas funcionales con base en los requisitos de AVRCP 1.0). El AMEF fue llevado a cabo por un equipo multidisciplinario y que juntos suman 26 años de experiencia en el ámbito de desarrollo software y pruebas de sistemas de entretenimiento para vehículos automotrices.

Los resultados del AMEF indican los siguientes modos de falla para los comandos de AVRCP 1.0 estudiados en el presente:

- El comando de “Siguiente” no responde a las acciones del usuario
- El comando de “Previo” no responde a las acciones del usuario

- El comando de “Reproducir” no responde a las acciones del usuario
- El comando de “Detener” no responde a las acciones del usuario

A continuación en la Tabla 7, se encuentran los resultados documentados del AMEF:

Id.	Funcionalidad	Modo de Falla	Causa Potencial de la Falla	Efecto de la Falla	Acciones a Tomar	Sev.	Ocu.	Det.	RPN
AMEF-1	"Siguiente"	"Siguiente" no cambia de pista acorde a las acciones del usuario	El usuario ejecutó cambios de pista más rápido de lo que soporta la funcionalidad de manera seguida inenterrumpida	Cambio lento de pista	Se diseñará una prueba Funcional	5	6	8	240
AMEF-2	"Previo"	"Previo" no cambia de pista acorde a las acciones del usuario	El usuario ejecutó cambios de pista más rápido de lo que soporta la funcionalidad de manera seguida inenterrumpida	Cambio lento de pista	Se diseñará una prueba Funcional	5	6	8	240
AMEF-3	"Reproducir"	"Reproducir" no inicia la reproducción de la pista acorde a las acciones del usuario	El usuario ejecutó los comandos de "Reproducir" y "Detener" manera concecutiva más rápido de lo que soporta la	No es posible iniciar la reproducción de la pista	Se diseñará una prueba Funcional	5	2	8	80
AMEF-4	"Detener"	"Detener" no detiene la reproducción de la pista acorde a las acciones del usuario	El usuario ejecutó los comandos de "Reproducir" y "Detener" manera concecutiva más rápido de lo que soporta la	No es posible detener la reproducción de la pista	Se diseñará una prueba Funcional	5	2	8	80

Tabla 7 - Resultados del AMEF

El objetivo de asignar un RPN a cada modo de falla es para priorizarlos y tomar decisiones con respecto a los que se consideran más relevantes, pero como en el caso presente sólo hay cuatro modos de falla, se diseñaron casos de prueba para todos. A raíz de los resultados del AMEF, se realizó un diagrama de casos de uso, ilustrado en la Figura 3.

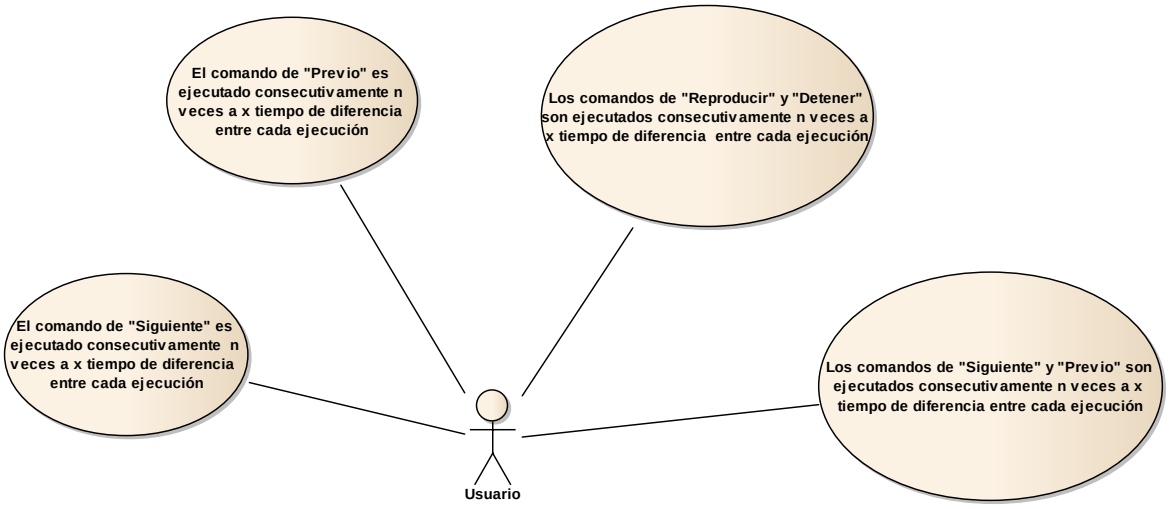

Figura 3 - Diagrama de Caso de Uso con base en los resultados del AMEF

Con base en el análisis anterior, se han diseñado una serie de pruebas que pueden generar éstos modos de falla. Dichas pruebas pueden observarse en la Tabla 8.

Pasos de Prueba	Acciones a Ejecutar	Tiempo entre comandos ejecutados	Cantidad de veces que se ejecutan los comandos	Resultado Esperado
1	"Siguiente"	1 seg.	20 veces	Al terminar la prueba, el comando de "Siguiente" debe cambiar a la siguiente pista
2	"Previo"	1 seg.	20 veces	Al terminar la prueba, el comando de "Previo" debe cambiar a la pista anterior
3	"Reproducir" y "Detener"	1 seg.	20 veces	Al terminar la prueba, los comandos de "Reproducir" y "Detener" deben iniciar y detener la reproducción respectivamente
4	"Siguiente" y "Previo"	1 seg.	20 veces	Al terminar la prueba, los comandos de "Siguiente" y "Previo" deben cambiar a la pista siguiente y anterior respectivamente

Tabla 8 - Pruebas diseñadas a partir de los resultados del AMEF

3.4 Automatización de las pruebas de AVRCP 1.0

3.4.1 Desarrollo de la Librería

Con el propósito de automatizar los casos de prueba manuales, se emplea el modelo en “V” para desarrollar una librería en el lenguaje de C# para implementar dicha automatización. En la Figura 4, se pueden observar las etapas de desarrollo de la librería. Consecutivamente, en los siguientes puntos, se describen las etapas llevadas a cabo para obtener dicha librería.

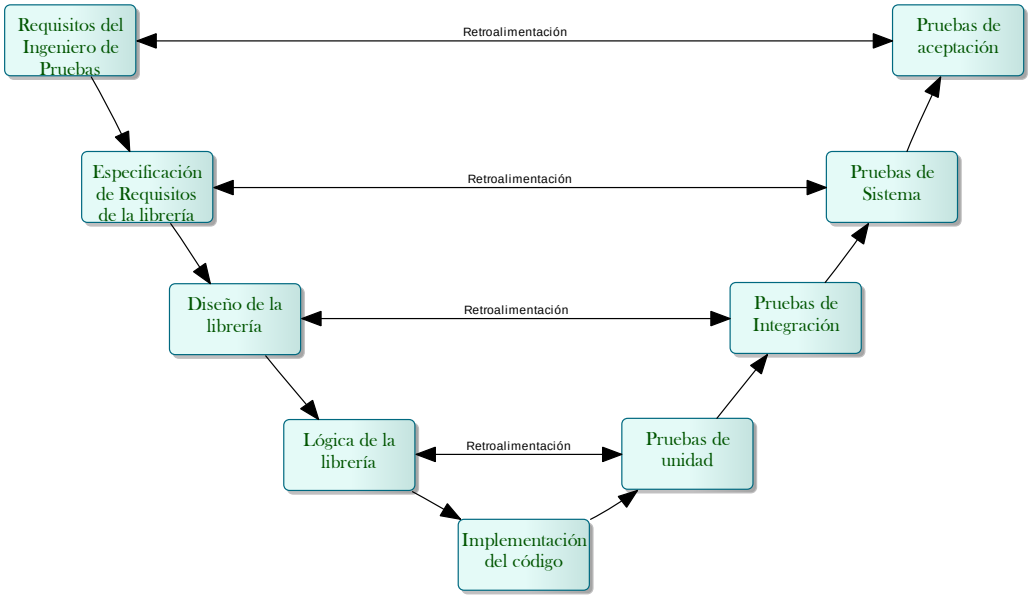

Figura 4 - Etapas de Desarrollo de la librería

3.4.1.1 Requisitos del Ingeniero de Pruebas

En el presente trabajo, el Ingeniero de Pruebas es considerado como el cliente. Por lo que se han analizado las acciones manuales concernientes a la

funcionalidad de Bluetooth AVRCP 1.0 llevadas a cabo durante las pruebas manuales definidas anteriormente.

Primeramente, se emplea un diagrama de casos de uso para analizar las acciones que requieren ser automatizadas por la librería, éste diagrama puede verse en la Figura 5.

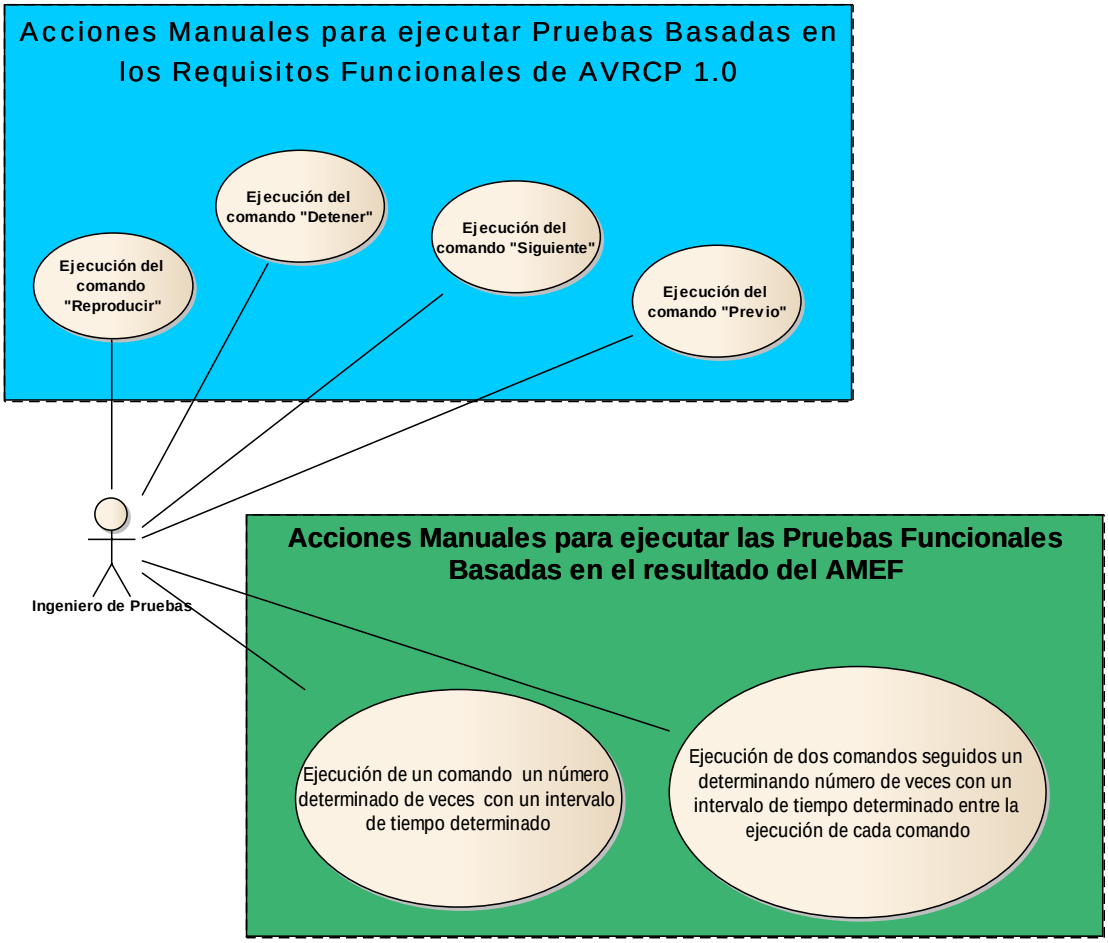

Figura 5 - Acciones manuales ejecutadas por el Ingeniero de Pruebas

Posteriormente, con base en el diagrama de casos de uso, se definen los requisitos del Ingeniero de Pruebas para automatizar dichas acciones manuales, los cuáles se encuentran descritos en la Tabla 9.

Id.	Descripción de Requisitos del Ingeniero de Pruebas
Ing-1	Automatizar la ejecución de los siguientes comandos de Bluetooth AVRCP 1.0: -"Reproducir" -"Siguiente" -"Previo" -"Detener"
Ing-2	Automatizar la ejecución de un comando un número determinado de veces con un intervalo de tiempo determinado
Ing-3	Automatizar la ejecución de dos comandos seguidos un determinado número de veces con un intervalo de tiempo determinado entre la ejecución de cada comando

Tabla 9 - Requisitos del Ingeniero de Pruebas

3.4.1.1.1 Diseño de las Pruebas de Aceptación

Paralelamente, se definieron las pruebas de aceptación para ejecutarlas en su respectiva etapa. El objetivo de esto es obtener retroalimentación temprana sobre los requisitos que se están definiendo, debido a que las pruebas de aceptación están diseñadas con base en los requisitos funcionales de la librería. Dichas pruebas se describen en la Tabla 10.

Pasos de Prueba	Acciones a Ejecutar	Precondiciones		Resultado Esperado
		Estado de Reproducción	Tiempo de Reproducción	
1	Conectar un dispositivo Android por medio de USB a la PC, previamente enlazado por medio de Bluetooth con un sistema de Entretenimiento	"	"	Este paso no requiere un resultado esperado
2	Solicitar la ejecución de los siguientes comandos desde la librería	"	"	Todos los comandos deber ser ejecutados
3	"Reproducir"	DET	X-	Inicio de Reproducción de pista
4	"Reproducir"	REP	X-	Sigue la Reproducción de la pista actual
5	"Siguiente"	REP	X-	Reproducción de la siguiente pista
6	"Previo"	REP	X-	Reproducción de la pista anterior
7	"Detener"	REP	X-	Detiene Reproducción de la pista actual
8	"Detener"	DET	X-	Sigue Detenida la Reproducción en la pista actual
9	"Siguiente"	DET	X-	Inicia Reproducción de la siguiente pista
10	"Detener"	REP	X-	Detiene Reproducción de la pista actual
11	"Previo" y un segundo después solicitar "Reproducir"	DET	X-	Inicia Reproducción de la pista anterior
12	Esperar que el tiempo de reproducción sea X+	REP	X-	Este paso establece una precondición, y se considera terminado hasta que el tiempo de reproducción sea X+
13	"Detener"	REP	X+	Detiene Reproducción de la pista actual
14	"Reproducir"	DET	X+	Inicio de Reproducción de la pista actual
15	"Reproducir"	REP	X+	Sigue la Reproducción de la pista actual
16	"Siguiente"	REP	X+	Reproducción de la siguiente pista
17	"Previo"	REP	X+	Reproducción de la pista actual
18	"Detener"	REP	X+	Detiene Reproducción de la pista actual
19	"Detener"	DET	X+	Sigue Detenida la Reproducción en la pista actual
20	"Siguiente"	DET	X+	Inicia Reproducción de la siguiente pista
21	"Detener"	REP	X+	Detiene Reproducción de la pista actual
22	"Previo" y un segundo después solicitar "Reproducir"	DET	X+	Inicia Reproducción de la misma pista
23	Solicitar el comando "Siguiente" consecutivamente cada segundo , una cantidad total de veinte veces	REP o DET	X- o X+	Al terminar la prueba, el comando de NEXT debe cambiar a la siguiente pista
24	Solicitar el comando "Previo" consecutivamente cada segundo , una cantidad total de veinte veces	REP o DET	X- o X+	Al terminar la prueba, el comando de "Previo" debe cambiar a la pista anterior
25	Solicitar los comandos de "Reproducir" y "Detener" alternada y consecutivamente cada segundo , una cantidad total de veinte veces	REP o DET	X- o X+	Al terminar la prueba, los comandos de "Reproducir" y "Detener" deben iniciar y detener la reproducción respectivamente
26	Solicitar los comandos de "Siguiente" y "Previo" alternada y consecutivamente cada segundo , una cantidad total de veinte veces	REP o DET	X- o X+	Al terminar la prueba, los comandos de "Siguiente" y "Previo" deben cambiar a la pista siguiente y anterior respectivamente

Tabla 10 - Pruebas de aceptación

3.4.1.2 Especificación de los requisitos de la librería de software

Como se mencionó en apartados anteriores, el dispositivo elegido para ejecutar las pruebas automatizadas es el Samsung Galaxy S3 Mini. Por lo tanto, se decide emplear la interface proporcionada por Google para tener acceso a los comandos de Android, esta es conocida como “Puente de depuración de Android” (ADB por sus siglas en Inglés) (Google, 2015).

Como paso consecuente, se generaron los requisitos funcionales para la librería considerando las características del dispositivo Android, el análisis de las acciones manuales para ejecutar los comandos de AVRCP 1.0 y las particularidades de las pruebas diseñadas previamente. Tales requisitos se encuentran descritos en la Tabla 11.

Id.	Descripción de Requisitos de la librería
Req-1	La librería debe proveer un medio para ejecutar el comando de "Reproducir" una vez.
Req-2	La librería debe proveer un medio para ejecutar el comando de "Detener" una vez.
Req-3	La librería debe proveer un medio para ejecutar el comando de "Siguiente" una vez.
Req-4	La librería debe proveer un medio para ejecutar el comando de "Previo" una vez.
Req-5	Los comandos de AVRCP 1.0 deben ser ejecutados en un dispositivo Android
Req-6	La librería debe proveer un medio para ejecutar un comando un número determinado de veces con un intervalo de tiempo determinado
Req-7	La librería debe proveer un medio para ejecutar dos comandos seguidos un determinado número de veces con un intervalo de tiempo determinado entre la ejecución de cada comando
Req-8	La librería debe comunicarse con el dispositivo Android por medio de USB a través de la interface ADB

Tabla 11 - Requisitos de la Librería

3.4.1.2.1 Diseño de las Pruebas de Sistema

Las pruebas de sistema se diseñan paralelamente, con el objetivo de proporcionar retroalimentación a la definición de los requisitos de la librería.

3.4.1.3 Diseño de la librería

En ésta etapa se diseña a alto nivel la arquitectura de la librería, con base en los requisitos generados previamente.

Con el fin de analizar los requisitos de la librería, se hace un diagrama de casos de uso, que se puede apreciar en la Figura 6. Con base en este diagrama, se ha desarrollado el diagrama de clases, ilustrado en la Figura 7. La clase “AVRCP” define los métodos y propiedades para un objeto que permite cumplir los requisitos estipulados para la librería. La definición de los comandos en la enumeración “ComandosEnum” está de acuerdo a la interface definida por ADB.

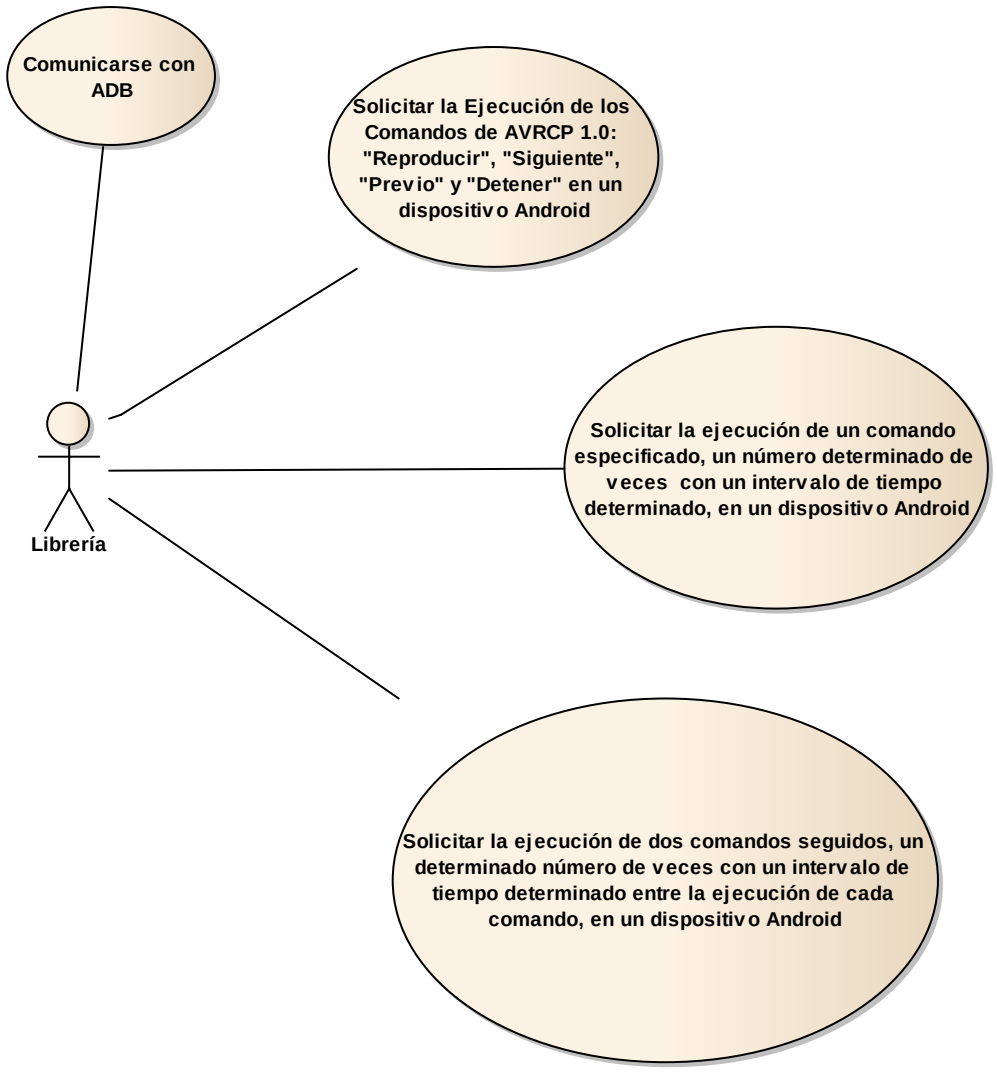

Figura 6 - Diagrama de casos de usos de librería

3.4.1.3.1 Diseño de las Pruebas de Integración

De igual manera, las pruebas de integración se definen tempranamente con el fin de emplearse como retroalimentación para la arquitectura de la librería.

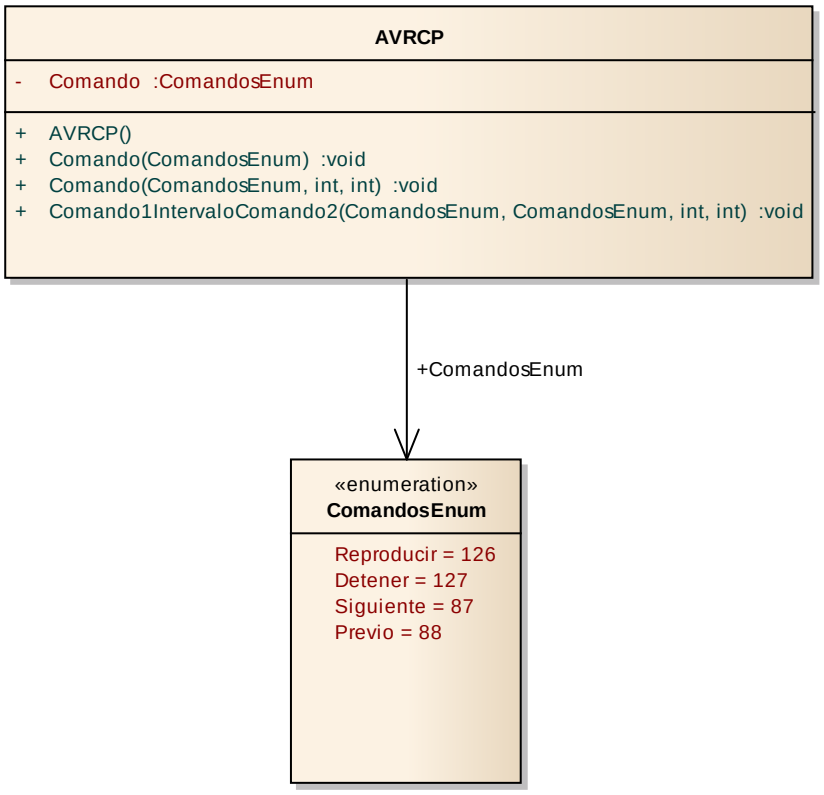

Figura 7 - Diagrama de clases de la librería

3.4.1.4 Lógica de la librería

En esta etapa se definió la lógica de funcionamiento de la librería, la cual sólo es descrita a grandes rasgos en el presente documento, por motivos de confidencialidad. Se siguieron los estándares de calidad de desarrollo de software definidos por la empresa, para garantizar la reusabilidad, modularidad, escalabilidad y mantenimiento de la librería.

De manera general, la clase “AVRCP” crea un objeto capaz de comunicarse con el ADB, que a su vez, solicita la ejecución de los comandos al dispositivo Android por medio de ADB. De igual manera, administra el tipo de comandos, la cantidad y el intervalo de tiempo entre la ejecución de los mismos.

3.4.1.4.1 Diseño de las Pruebas de Unidad

Al igual que el resto de las pruebas mencionadas anteriormente, las pruebas de unidad se diseñan con el objetivo de proporcionar retroalimentación temprana.

3.4.1.5 Implementación del Código

La implementación del código se lleva a cabo empleando el lenguaje C# con base en la arquitectura diseñada previamente, siguiendo los estándares de calidad de desarrollo de software definidos por la empresa, para garantizar la reusabilidad, modularidad, escalabilidad y mantenimiento de la librería.

3.4.1.6 Revisión del proceso y resultados de las pruebas

3.4.1.6.1 Revisión del proceso

Se realizan reuniones periódicas de revisión durante todas las etapas del proceso de desarrollo de la librería, con ingenieros de distintas especialidades, con el objetivo de retroalimentar cada una de las etapas. Los ingenieros participantes de dichas reuniones son los siguientes:

- 3 Ingenieros de Pruebas
- 2 Ingenieros de Software
- 1 Ingeniero de Sistema del Producto

La suma de la experiencia de todos los ingenieros revisores del desarrollo de la librería es de 36 años en total.

3.4.1.6.2 Resultados de las etapas de pruebas

Todas las pruebas son ejecutadas de manera iterativa y regresiva, es decir, al detectar un falla esta es corregida y la prueba es ejecutada de nuevo hasta que la librería pasa todas las pruebas.

Aunque todas las etapas de prueba obtuvieron resultados positivos, en el presente trabajo sólo es de interés describir los resultados de las pruebas de aceptación, ya que ésta determina si la librería cumple o no con lo que requiere el ingeniero de pruebas. En la Tabla 12 se pueden observar dichos resultados.

3.4.1.7 Pruebas de desempeño

Con el objetivo de proporcionar información sobre el consumo de recursos de la librería, se llevan a cabo las pruebas de desempeño, cuya descripción y resultados se encuentran en la Tabla 13.

Las pruebas son llevadas a cabo en una computadora con las siguientes características:

- Sistema Operativo Windows 7 Profesional 64-bit
- Procesador Intel Core i7-4770 CPU @ 3.40GHz.
- Memoria RAM de 16 GB.

Pasos de Prueba	Acciones a Ejecutar	Precondiciones		Resultado Esperado	Resultado del Paso de Prueba
		Estado de Reproducción	Tiempo de Reproducción		
1	Conectar un dispositivo Android por medio de USB a la PC, previamente enlazado por medio de Bluetooth con un sistema de Entretenimiento	"	"	Este paso no requiere un resultado esperado	Aprobado
2	Solicitar la ejecución de los siguientes comandos desde la librería	"	"	Todos los comandos deber ser ejecutados	Aprobado
3	"Reproducir"	DET	X-	Inicio de Reproducción de pista	Aprobado
4	"Reproducir"	REP	X-	Sigue la Reproducción de la pista actual	Aprobado
5	"Siguiente"	REP	X-	Reproducción de la siguiente pista	Aprobado
6	"Previo"	REP	X-	Reproducción de la pista anterior	Aprobado
7	"Detener"	REP	X-	Detiene Reproducción de la pista actual	Aprobado
8	"Detener"	DET	X-	Sigue Detenida la Reproducción en la pista actual	Aprobado
9	"Siguiente"	DET	X-	Inicia Reproducción de la siguiente pista	Aprobado
10	"Detener"	REP	X-	Detiene Reproducción de la pista actual	Aprobado
11	"Previo" y un segundo después solicitar "Reproducir"	DET	X-	Inicia Reproducción de la pista anterior	Aprobado
12	Esperar que el tiempo de reproducción sea X+	REP	X-	Este paso establece una precondición, y se considera terminado hasta que el tiempo de reproducción sea X+	Aprobado
13	"Detener"	REP	X+	Detiene Reproducción de la pista actual	Aprobado
14	"Reproducir"	DET	X+	Inicio de Reproducción de la pista actual	Aprobado
15	"Reproducir"	REP	X+	Sigue la Reproducción de la pista actual	Aprobado
16	"Siguiente"	REP	X+	Reproducción de la siguiente pista	Aprobado
17	"Previo"	REP	X+	Reproducción de la pista actual	Aprobado
18	"Detener"	REP	X+	Detiene Reproducción de la pista actual	Aprobado
19	"Detener"	DET	X+	Sigue Detenida la Reproducción en la pista actual	Aprobado
20	"Siguiente"	DET	X+	Inicia Reproducción de la siguiente pista	Aprobado
21	"Detener"	REP	X+	Detiene Reproducción de la pista actual	Aprobado
22	"Previo" y un segundo después solicitar "Reproducir"	DET	X+	Inicia Reproducción de la misma pista	Aprobado
23	Solicitar el comando "Siguiente" consecutivamente cada segundo , una cantidad total de veinte veces	REP o DET	X- o X+	Al terminar la prueba, el comando de NEXT debe cambiar a la siguiente pista	Aprobado
24	Solicitar el comando "Previo" consecutivamente cada segundo , una cantidad total de veinte veces	REP o DET	X- o X+	Al terminar la prueba, el comando de "Previo" debe cambiar a la pista anterior	Aprobado
25	Solicitar los comandos de "Reproducir" y "Detener" alternada y consecutivamente cada segundo , una cantidad total de veinte veces	REP o DET	X- o X+	Al terminar la prueba, los comandos de "Reproducir" y "Detener" deben iniciar y detener la reproducción respectivamente	Aprobado
26	Solicitar los comandos de "Siguiente" y "Previo" alternada y consecutivamente cada segundo , una cantidad total de veinte veces	REP o DET	X- o X+	Al terminar la prueba, los comandos de "Siguiente" y "Previo" deben cambiar a la pista siguiente y anterior respectivamente	Aprobado

Tabla 12 - Resultados de las Pruebas de Aceptación

Pruebas de desempeño de librería	Tiempo total de los comandos ejecutados	Tiempo Promedio por comando	Consumo de Recursos del CPU durante las pruebas	Consumo de Recursos antes de las pruebas
Sin ejecutar comandos	NA	NA	pico de 6%, estable 1%	1%
Ejecución de 10 comandos	9 s	0.9 s	pico de 3%	1%
Ejecución de 100 comandos	98 s	0.98 s	pico de 7%	1%
Ejecución de 1000 comandos	1000 s	1 s	pico de 9%	1%

Tabla 13- Pruebas de Desempeño

3.4.2 Implementación y Ejecución de las pruebas automatizadas

Empleando la librería previamente desarrollada, se decide elaborar una secuencia estructurada que permite automatizar las acciones manuales concernientes a la ejecución de los comandos de Bluetooth AVRCP 1.0 durante las pruebas funcionales definidas previamente.

Para programar la secuencia se utiliza el software secuenciador de pruebas “TestStand” de “National Instruments”(TestStand - National Instruments, 2015).

La secuencia automatizada se diseña con base en las pruebas funcionales elaboradas en la Tabla 10. Las verificaciones de los resultados de las pruebas se llevan a cabo de manera visual por parte del ingeniero, pero pueden ser sustituidas según el criterio del mismo, ya sea empleando una herramienta complementaria de retroalimentación visual o auditiva. Éstas verificaciones se mantienen manuales con el propósito de establecer un equilibrio entre las pruebas automatizadas y manuales, ya que una no sustituye a la otra, sino que son complementarias (Meng, 2011). La automatización de los comandos de Bluetooth AVRCP 1.0 en las pruebas funcionales es ilustrada en la Figura 8.

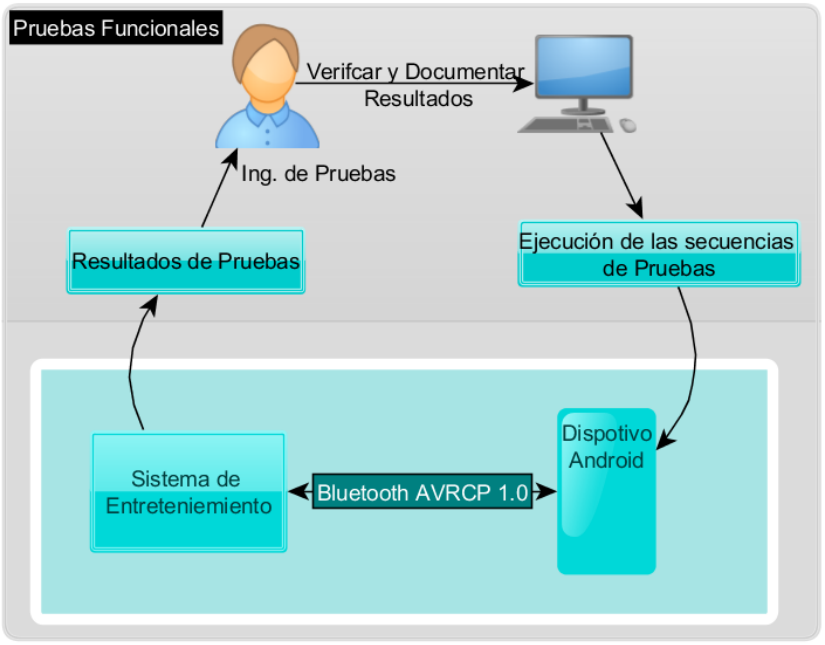

Figura 8 - Automatización de comandos de Bluetooth AVRCP 1.0 en pruebas funcionales

Las pruebas funcionales automatizadas son ejecutadas de manera metódica y documentada.

4. RESULTADOS

Esta tesis se enfocó a las pruebas funcionales o de caja negra de Bluetooth AVRCP 1.0, por lo tanto, se empleó el modelo de diseño de pruebas funcionales basadas en requisitos. Se diseñaron una serie de casos de prueba funcionales de acuerdo a los requisitos de usuario contenidos en las especificaciones del perfil de AVRCP 1.0 del protocolo de Bluetooth. Posteriormente, se llevó a cabo un AMEF para determinar modos de falla no contemplados por las pruebas basadas en requisitos. Consecutivamente, se desarrolló una librería de software, bajo el esquema del modelo en “V”, que permitió automatizar los casos de pruebas. Finalmente, se implementó la automatización de las pruebas, utilizando la librería desarrollada en esta tesis, mediante una secuencia estructurada elaborada en “TestStand”, permitiendo el

reúso de las pruebas en cualquier sistema de entretenimiento que soporte Bluetooth AVRCP 1.0 conectado a un dispositivo Android. Cabe añadir que, gracias a la documentación y al proceso de desarrollo seguido, la librería es actualizable y orientada al mantenimiento, en caso de ser requerido. Además, como valor agregado a este proyecto, se han definido los casos de prueba del AMEF para ser ejecutados mediante la librería. Este aspecto se considera importante, ya que se posibilita la verificación de los modos de falla como resultado del AMEF.

5. CONCLUSIONES Y TRABAJO FUTURO

La automatización de las pruebas funcionales de los sistemas de entretenimiento, independientemente de la complejidad de la funcionalidad, representan un gran reto para el ingeniero de pruebas, ya que requiere de un profundo conocimiento de dicha funcionalidad y de una herramienta o librería que permita dicha automatización. En caso de que sea requerido diseñar una librería, como en el presente trabajo, además de la teoría relacionada a las pruebas de software, es necesario que el ingeniero de pruebas posea conocimientos de diseño de software para tener las habilidades requeridas para diseñar y desarrollar software, como pueden ser: modelos de desarrollo, creación de requisitos, arquitectura de software, procesos de revisión de código, documentación, por mencionar algunos ejemplos.

Particularmente, las pruebas que involucran la interacción con dispositivos externos al sistema de entretenimiento, como es el caso de las pruebas de Bluetooth AVRCP 1.0, simbolizan un desafío aún mayor, ya que también se deben conocer las características del dispositivo involucrado.

5.1 TRABAJO FUTURO

La librería desarrollada en ésta tesis sienta las bases para que en un trabajo futuro se agreguen más protocolos de AVRCP, por ejemplo el 1.3 y el 1.5. De igual manera, se puede complementar la librería implementando métodos de verificación automatizada que permitan sustituir aún más acciones manuales, repetitivas y monótonas.

Además, si se considera necesario, como una buena mejora, se pueden añadir a la arquitectura de la librería, dispositivos con otros sistemas operativos como por ejemplo iOS.

6. REFERENCIAS

Antony, J. (2014). *Design of experiments for engineers and scientists*. Elsevier.

Beizer, B. (1990). *Software Testing Techniques (2Nd Ed.)*. New York, NY, USA: Van Nostrand Reinhold Co.

Bluetooth SIG. (2012). *Audio/Video Remote Control Profile*. Versión 1.5. Consultado en http://teleorigin.com/download/BT/docs/descr2/AVRCPSpecv10.pdf

Brownlie, R., Prowse, J., & Phadke, M. S. (1992). Robust Testing of AT&T PMX/StarMAIL Using Oats. *AT&T Technical Journal*, *71*(3), 41–47.

Burroughs, K., Jain, A., & Erickson, R. L. (1994). Improved quality of protocol testing through techniques of experimental design. En *Communications, 1994. ICC'94, SUPERCOMM/ICC'94, Conference Record,'Serving Humanity Through Communications.'IEEE International Conference on* (pp. 745–752). IEEE.

Butler, M. (2011). Android: Changing the Mobile Landscape. *Pervasive Computing, IEEE*, *10*(1), 4-7. http://doi.org/10.1109/MPRV.2011.1

Cohen, D. M., Dalal, S. R., Parelius, J., & Patton, G. C. (1996). The combinatorial design approach to automatic test generation. *IEEE software*, *13*(5), 83–88.

Collins, E., Dias-Neto, A., & Lucena Jr., V. F. de. (2012). Strategies for Agile Software Testing Automation: An Industrial Experience (pp. 440-445). IEEE. http://doi.org/10.1109/COMPSACW.2012.84

Dunietz, I., Ehrlich, W. K., Szablak, B., Mallows, C. L., & Iannino, A. (1997). Applying design of experiments to software testing: experience report. En *Proceedings of the 19th international conference on Software engineering* (pp. 205–215). ACM.

Dustin, E., Rashka, J., & Paul, J. (1999). *Automated software testing: introduction, management, and performance*. Addison-Wesley Professional.

González, F. A. D., Mayorga, J. A. A., Prado, M. X. C., & Bedoya, A. C. C. (2013). Development Model for Technological Products. *International Journal of Applied Science and Technology*, *3*(8). Consultado en http://www.ijastnet.com/journals/Vol_3_No_8_December_2013/7.pdf

Google. (2015). Android debug bridge. Consultado en http://developer.android.com/tools/help/adb.html

Graaf, B., Lormans, M., & Toetenel, H. (2003). Embedded software engineering: the state of the practice. *Software, IEEE*, *20*(6), 61–69.

Haapanen, P., & Helminen, A. (2002). *Failure mode and effects analysis of software-based automation systems*. Helsinki: Radiation and Nuclear Safety Authority.

Haartsen, J., Naghshineh, M., Inouye, J., Joeressen, O. J., & Allen, W. (1998). Bluetooth: Vision, goals, and architecture. *ACM SIGMOBILE Mobile Computing and Communications Review*, *2*(4), 38–45.

Hanna, M., El-Haggar, N., & Sami, M. (2014). A Review of Scripting Techniques Used in Automated Software Testing. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *5*(1). Consultado en http://thesai.org/Publications/ViewPaper?Volume=5&Issue=1&Code=IJACSA&SerialNo=28

Homès, B. (2013). *Fundamentals of Software Testing*. John Wiley & Sons.

Hsia, P., Gao, J., Samuel, J., Kung, D., Toyoshima, Y., & Chen, C. (1994). Behavior-based acceptance testing of software systems: a formal scenario approach. En *Computer Software and Applications Conference, 1994. COMPSAC 94. Proceedings., Eighteenth Annual International* (pp. 293–298). IEEE.

IDC: Smartphone OS Market Share 2014, 2013, 2012, and 2011. (2014). [Estadística de Mercado de Consumo Tecnológico]. Consultado el 27 de enero de 2015, en http://www.idc.com/prodserv/smartphone-os-market-share.jsp

IEEE Draft International Standard for Software and Systems Engineering–Software Testing–Part 4: Test Techniques. (2014). *ISO/IEC/IEEE P29119-4-DISMay2013*, 1-132.

Jo, H. C., Piao, S., Cho, S. R., & Jung, W. Y. (2008). Rte template structure for autosar based embedded software platform. En *Mechatronic and Embedded Systems and Applications, 2008. MESA 2008. IEEE/ASME International Conference on* (pp. 233–237). IEEE.

Kang, B., Kwon, Y.-J., & Lee, R. Y. (2005). A design and test technique for embedded software. En *Software Engineering Research, Management and Applications, 2005. Third ACIS International Conference on* (pp. 160–165). IEEE.

Kasoju, A., Petersen, K., & Mäntylä, M. V. (2013). Analyzing an automotive testing process with evidence-based software engineering. *Information and Software Technology*, *55*(7), 1237-1259.

Kern, D., & Schmidt, A. (2009). Design space for driver-based automotive user interfaces (pp. 3–10). ACM. Consultado en http://dl.acm.org/citation.cfm?id=1620511

Kuhn, D. R., & Reilly, M. J. (2002). An investigation of the applicability of design of experiments to software testing. En *Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE* (pp. 91–95). IEEE.

Kumar, A., Fernando, S., & Panicker, R. C. (2013). Project-Based Learning in Embedded Systems Education Using an FPGA Platform. *IEEE Transactions on Education*, *56*(4), 407-415. http://doi.org/10.1109/TE.2013.2246568

LaRussa, J. J., & Gabel, M. (2010). Standardization Proposal for «Automotive-Grade AVRCP» with Respect to In-Car use of Bluetooth Devices. *Development*, *1*, 0104.

Li, K., & Wu, M. (2006). *Effective software test automation: developing an automated software testing tool*. John Wiley & Sons.

Mathur, S., & Malik, S. (2010). Advancements in the V-Model. *International Journal of computer applications*, *1*(12).

McGregor, J. D., & Sykes, D. A. (2001). *A practical guide to testing object-oriented software*. Addison-Wesley Professional.

Meng, X. (2011). Analysis of software automation test protocol. En *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on* (Vol. 8, pp. 4138–4141). IEEE.

Minitab. (2015, enero 19). Consultado el 19 de enero de 2015, a partir de http://www.minitab.com/es-mx/

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.

Nguyen, D. (2001). Failure modes and effects analysis for software reliability. En *Reliability and Maintainability Symposium, 2001. Proceedings. Annual* (pp. 219–222). IEEE. Pressman, R. S. (2001). Software engineering: a practitioner's approach.

Seo, J., Ki, Y., Choi, B., & La, K. (2008). Which Spot Should I Test for Effective Embedded Software Testing? (pp. 135-142). IEEE. http://doi.org/10.1109/SSIRI.2008.38

Sharma, H., & Ramani, A. K. (2009). Context Aware In-Vehicle Infotainment Interfaces. En *Advances in Electrical Engineering and Computational Science* (pp. 343–354). Springer.

Software and systems engineering Software testing Part 1:Concepts and definitions. (2013). *ISO/IEC/IEEE 29119-1:2013(E)*, 1-64. http://doi.org/10.1109/IEEESTD.2013.6588537

Systems and software engineering – Vocabulary. (2010). *ISO/IEC/IEEE 24765:2010(E)*, 1-418. http://doi.org/10.1109/IEEESTD.2010.5733835

Tassey, G. (2002). The economic impacts of inadequate infrastructure for software testing. *National Institute of Standards and Technology, RTI Project*, *7007*(011).

Tchankue, P., Wesson, J., & Vogts, D. (2011). The impact of an adaptive user interface on reducing driver distraction (pp. 87–94). ACM. Consultado en http://dl.acm.org/citation.cfm?id=2381430

TestStand - National Instruments. (2015, marzo 9). Consultado el 9 de marzo de 2015, a partir de http://www.ni.com/teststand/

Tsai, W.-T., Yu, L., Zhu, F., & Paul, R. (2005). Rapid embedded system testing using verification patterns. *Software, IEEE*, *22*(4), 68–75.

Whittaker, J. A. (2000). What is software testing? And why is it so hard? *Software, IEEE*, *17*(1), 70–79.

7. APÉNDICE

ADB	Puente de depuración de Android
AMEF	Análisis de Modos y Efectos de Falla
AVRCP	Control de Audio/Video Remoto
DOE	Diseño de Experimentos
NIST	Instituto Nacional de Estándares y Tecnología de Estados Unidos
PAN	Área de Red Personal
RPN	Número de Prioridad de Riesgo
SIG	Grupo de Interés Especial

Tabla 14 - Abreviaturas empleadas