**Universidad de las Ciencias Informáticas**

**Facultad 6**


TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

**Título:** *Plugin* para la síntesis de voz de las noticias en la Plataforma de Televisión Informativa PRIMICIA.

**Autora:**

Dayana Pérez Sariol

**Tutores:**

Ing. Mairena Arzuaga Limonta

Ing. Miguel Morciego Varona

La Habana, julio 2016

"Año 54 de la Revolución."

*Incluso la gente que afirma que no podemos hacer nada para cambiar nuestro destino, mira antes de cruzar la calle.*

*Stephen Hawking*

## Dedicatoria

*A mi mamá, a mi papá y a mi abuela Nena donde quiera que se encuentre.*

## Agradecimientos

*Quiero agradecerle a esa persona tan maravillosa que me curó cuando estaba enferma, secó mis lágrimas cuando lloraba, limpió y cuidó mis heridas cuando me caía, a esa que sufre incluso más que yo cuando sufro, a ti mamá.*

*Madurar es darse cuenta que el primer, único y verdadero amor siempre será papá, a ti papi te agradezco por guiarme y apoyarme siempre.*

*Ya han pasado algunos años y ella no está aquí para verme hoy, abue, donde quiera que estés, quiero decirte gracias, siempre te llevo en mi corazón.*

*A Dariel por ayudarme a comprender mejor la vida, por sacrificarse en las madrugadas para ayudarme a estudiar y por compartir momentos tan hermosos que se quedarán grabados en mi mente por siempre.*

*A mi tía Elisa, mis primos Harold y Elizabeth y mi tío Pepe por educarme y complacerme en todo. A toda mi familia en general. A mi tan querido hermano Alejandro, aunque no compartamos la misma sangre siempre lo serás en mi corazón. A mi amigo Medina. A Eddy, un millón de gracias amigo, te quiero un montón. A mis compañeros de clases. A Maggi. A Yuneiry, Julio César, Lázaro, Mauricio. A mis tutores Mairena y Miguel por tener mucha paciencia y estar presentes a cualquier hora. A todos mis profesores. Muy especial a los profes Yanelis Benítez, Yeleny Zulueta y al profe Yusdel. A las profes Indira y Yanary por comprenderme y ayudarme a poder exponer este último trabajo. Al tribunal por siempre escuchar nuestros puntos de vista.*

**Declaración de autoría**

Declaro ser autor de la tesis "*Plugin* para la síntesis de voz de las noticias en la Plataforma de televisión Informativa PRIMICIA" y reconozco los derechos patrimoniales de la misma, con carácter exclusivo, a la Universidad de las Ciencias Informáticas (UCI).

Para que así conste firmo la presente a los_____días del mes de _____del año_____.

Dayana Pérez Sariol

_____

Firma del Autor

Ing. Mairena Arzuaga Limonta

_____

Firma del tutor

Ing. Miguel Morciego Varona

_____

Firma del tutor

## Datos de contacto

Autora: Dayana Pérez Sariol

Dirección de correo electrónico: dsariol@estudiantes.uci.cu

Tutor: Ing. Mairena Arzuaga Limonta

Dirección de correo electrónico: mlimonta@.uci.cu

Tutor: Ing. Miguel Morciego Varona

Dirección de correo electrónico: mmorciego@.uci.cu

**Resumen:**

En la Universidad de la Ciencias Informáticas (UCI) se desarrolló una Plataforma Televisión Informativa denominada PRIMICIA. La plataforma constituye una solución integral que permite la transmisión automática y constante de informaciones en diferentes formatos a través de un canal de televisión. PRIMICIA no tiene incluido un proceso de síntesis de voz que permita la lectura de las noticias, lo que provoca que no pueda ser desplegado en ambientes donde a las personas se le dificulte seguir la noticia de manera visual.

El presente trabajo de diploma tiene como objetivo desarrollar un *plugin* para la síntesis de voz de las noticias en PRIMICIA, con el propósito de adquirir nuevos clientes interesados en la visualización de noticias haciendo uso de la televisión. El *plugin* ha sido desarrollado completamente en software libre, contribuyendo de esta manera a la soberanía tecnológica del país. Además una de sus principales características es ser multiplataforma.

**Palabras claves:**

Síntesis de voz, *plugin*, información, televisión.

***Abstract:***

*In the University of Informatics Sciences (UCI by its Spanish acronyms) was developed an Informative TV Platform called PRIMICIA. It is an integral solution that allows automatic and constant transmission of information in different formats through a TV channel. PRIMICIA has no process of voice synthesis included to read news, therefore it is impossible to be deployed in environments where people find it difficult to follow news visually.*

*The present diploma paper aims at developing a plug-in for news voice synthesis in PRIMICIA, in order to attract new clients interested in visualizing news via television. The plug-in has been totally developed in free software, contributing this way to the technological sovereignty of the country. Apart from that, one of its main features is that it is multiplatform.*

***Keywords:***

*Speech synthesis, plugin, information, television.*

## Índice

**Índice de figuras**

## Índice de tablas

## Introducción

El hombre, a través de la historia ha utilizado diferentes vías para comunicarse, la voz le ha permitido el habla y por ende la comunicación clara y precisa, la cual, combinada con información visual le ha facilitado incluso sin conocer lugares o cosas tener una visión de los mismos. La forma de transmitir la información cuando nos comunicamos con otro ser humano no es única ni uniforme. Además del significado de las palabras o frases que pronunciamos existen una serie de factores adicionales que son igual de importantes, como la voz de la persona que habla, su timbre, la entonación, los gestos que hace cuando transmite el mensaje, las palabras que escoge para hacerlo, el acento, entre otros factores.

El complejo proceso que subyace debajo de la comunicación oral es constante fuente de investigación. A finales del siglo XX y principios del XXI, se han desarrollado técnicas para la simulación de la voz humana, denominada síntesis de voz. El desarrollo de dichas técnicas se ha visto influenciado por diversos factores que han dado por resultado una gama muy extensa de esquemas y tecnologías, siendo los principales factores: la calidad deseada, la disponibilidad electrónica capaz de ejecutar los algoritmos de síntesis de voz en el tiempo requerido y el costo de cada tecnología. El estudio de la síntesis de voz se divide en dos áreas que se han desarrollado en forma separada, la síntesis de bajo nivel y la síntesis de alto nivel. La síntesis de bajo nivel se refiere a la producción real del sonido y simula la señal del lenguaje hablado mientras que la síntesis de alto nivel maneja la conversión de textos escritos.

La síntesis de voz está completamente ligada al avance de las tecnologías. El desarrollo investigativo y tecnológico que existe en el procesamiento digital de voz (síntesis de voz) ha sido desarrollado principalmente en Estados Unidos, por lo cual el área de síntesis de voz se ha impulsado en el idioma inglés. Algunos países han tenido que ampliar sus conocimientos desarrollando en lenguaje propio como ha sido el finlandés, japonés, francés, entre otros. Cada lenguaje tiene sus propios fonemas, grafos, pronunciación y reglas ortográficas, los cuales hay que adecuar a las teorías y técnicas diferentes que existen para poder llevar a cabo el proceso de síntesis de voz.

Dentro del Centro Geoinformática y Señales Digitales de la Universidad de la Ciencias Informáticas se desarrolló una plataforma denominada PRIMICIA. Dicha plataforma es una solución integral, multiplataforma, desarrollada usando software libre y capaz de proveer un canal de televisión para la transmisión automática y constante de informaciones en distintos formatos. Actualmente PRIMICIA se encuentra en fase de desarrollo de una nueva versión con el objetivo de elevar la accesibilidad y aumentar su alcance. La condición que posee PRIMICIA no le permite que pueda ser desplegado en diversos

ambientes donde se acumulan personas con necesidad de obtener información que les facilite comprender la televisión de manera visual. Existen diversas esferas en las cuales PRIMICIA no puede ser desplegado, a continuación se expresan con mayor profundidad.

En algunos establecimientos a muchas personas se les dificulta seguir la noticia de manera visual, ejemplo de ellos son las terminales de ómnibus y trenes, aeropuertos, hospitales, círculos infantiles, hogares de ancianos, entre otros, que por sus características necesitan mantener informados a las personas que allí se reúnen. En ocasiones, en dichos lugares se escuchan quejas, por no contar con vías accesibles que proporcionen información que sirva de guía o la información es escasa.

En Cuba se lleva a cabo una campaña de atención a los discapacitados visuales para mejorar su calidad de vida, pero en ocasiones no se cuenta con los recursos necesarios para que reciban una buena información. Las personas con discapacidades visuales tienen dificultad al no poder comprender íntegramente la información por vía televisiva. Para que la información llegue apropiadamente a dichos clientes se hace uso de una combinación de resto visual, oído y audiodescripción, la cual consiste en una voz en off que va describiendo lo que sucede en pantalla.

Por otra parte, la población cubana envejece a ritmo acelerado y la esperanza de vida se aproxima a los 80 años, pero en esta etapa de la vida comienzan a disminuir la visión y con ello la imposibilidad de estar al tanto de todo lo que sucede en el ámbito nacional e internacional por vía televisiva. Es importante destacar también los círculos infantiles, en pos de ayudar al desarrollo cognitivo del infante, se necesita crear condiciones para que los niños participen de todo lo que sucede a su alrededor y poder contar con una vía para hacerle llegar la información.

La situación descrita anteriormente permite formular el siguiente **problema científico**: La forma en que la plataforma PRIMICIA transmite los contenidos limita el acceso de los televidentes a las noticias.

Se determina como **objeto de estudio** los procesos de síntesis de voz.

A partir de la relación entre el problema científico, el objeto de estudio y el objetivo general de la investigación, se deriva como **campo de acción**: los procesos de síntesis de voz en la Plataforma de Televisión Informativa PRIMICIA.

El **objetivo general** de la investigación es desarrollar un *plugin* para la síntesis de voz en la Plataforma de Televisión Informativa PRIMICIA que contribuya al acceso de los televidentes a los contenidos que se transmiten.

Para dar cumplimiento al objetivo general planteado se especifican como **tareas de la investigación:**

1) Caracterizar los posibles algoritmos de síntesis de voz que puedan ser usados en la Plataforma de Televisión Informativa PRIMICIA.
2) Caracterizar soluciones existentes que permitan la síntesis de voz dentro de plataformas digitales de transmisión de contenidos audiovisuales.
3) Definir y caracterizar las herramientas y metodologías a utilizar durante el desarrollo del *plugin*.
4) Definir los requisitos funcionales y no funcionales del *plugin* para síntesis de voz de la plataforma PRIMICIA.
5) Modelar el sistema mediante el uso de una herramienta CASE.
6) Implementar el *plugin* para la síntesis de voz de las noticias en la plataforma PRIMICIA.
7) Validar la solución propuesta.

Los **Métodos Científicos** empleados son:

**Métodos Teóricos:**

 Analítico - Sintético: se empleó para analizar las diversas teorías de la síntesis de voz y definir sus características generales.

 Análisis Histórico – Lógico: se empleó en el estudio de los antecedentes, la evolución y el desarrollo que han tenido los sistemas de síntesis de voz actuales, así como valorar las tendencias actuales de los algoritmos de síntesis de voz.

 Inductivo – deductivo: se empleó para la identificación de la problemática planteada y para poder darle solución.

 Modelación: se empleó para crear abstracciones con vistas a explicar la realidad por medio de la realización de un modelo.

**Métodos Empíricos:**

 Método de observación: se empleó en la etapa inicial de la investigación para obtener conocimientos acerca del comportamiento del problema a investigar y para el diseño de la investigación.

La investigación está estructurada de la siguiente forma:

**Capítulo 1: Fundamentación teórica.** En este capítulo serán expuestos los principales conceptos asociados al problema en cuestión para lograr un mejor entendimiento del mismo. Se explican además todos los argumentos del objeto de estudio así como una síntesis de algunas soluciones existentes para el problema identificado.

**Capítulo 2: Herramientas y Tecnologías.** Este capítulo tratará acerca de las herramientas y tecnologías que se utilizan en la actualidad a nivel internacional para el desarrollo de la síntesis de voz. Se estudia aquí un conjunto de metodologías ágiles o pesadas que se manejan en el mundo para el desarrollo de software de forma general, con el objetivo de seleccionar de ellas la más adecuada para guiar el proceso de desarrollo del software. También se expone qué lenguaje de modelado y herramientas CASE son las más adecuadas para la construcción de los artefactos correspondientes a cada uno de los flujos de trabajo.

**Capítulo 3: Descripción de la solución propuesta.** Se presenta la solución propuesta a partir de la descripción del modelo de dominio y la identificación de requisitos funcionales y no funcionales del sistema, se identifican los casos de uso del sistema, se realiza una descripción textual de cada uno de ellos y se conforma el Modelo de Casos de Uso del Sistema.

**Capítulo 4: Validación de requisitos.** En este capítulo se realiza la validación de requisitos mediante las técnicas correspondientes.

# Capítulo 1
# Fundamentación teórica

## Introducción

En este capítulo se describen las bases teóricas del proceso de síntesis de voz. Para ello, se esclarecen algunos conceptos importantes para el entendimiento y el dominio del problema. También se realiza un análisis de los distintos sistemas que utilizan síntesis de voz existentes en el mundo.

## 1.1. Conceptos asociados al dominio del problema

Con el objetivo de que el lector pueda tener una mejor comprensión de los temas que se relacionan en el capítulo, se describen detalladamente a continuación un grupo de conceptos asociados al dominio del problema, entre los que se destacan: concepto de voz, síntesis del habla, fonos, difonos y tonos.

### Concepto de voz

La palabra voz se originó en el latín “vox”. La voz humana consiste en un sonido emitido por un ser humano usando las cuerdas vocales para hablar, cantar, reír, llorar y gritar. Los pulmones deben producir un flujo de aire adecuado para que las cuerdas vocales vibren. Las cuerdas vocales son una estructura vibradora, que realizan un 'ajuste fino' de tono y timbre. Los articuladores (tracto vocal) consisten en lengua, paladar y labios. Articulan y filtran el sonido. Las cuerdas vocales, en combinación con los articulares, son capaces de producir grandes rangos de sonidos (I. R., 2006).

### Síntesis del habla

La síntesis de habla es la producción artificial del habla. El sistema computarizado que es usado con este propósito es llamado computadora de habla o sintetizador de voz y puede ser implementado en productos software o hardware. Un sistema *text-to-speech* (TTS) o síntesis del habla, convierte el lenguaje de texto normal en habla; otros sistemas recrean la representación simbólica lingüística como transcripciones fonéticas en habla (Allen , y otros, 1987).

### Fono

En el habla humana, fono es cada uno de los segmentos de características acústicas particulares y con duración típica en que podemos dividir la secuencia sonora.

Cada fono viene caracterizado por un espectro de frecuencias características y un tiempo de emisión característico (típicamente 20-60 ms). En el análisis del habla se usa el análisis espectrográfico que permite descomponer las ondas sonoras del habla en superposición de ondas más simples de frecuencias fijas. Los fonos similares entre sí se representan por signos alfabéticos entre corchetes (Dürr, y otros, 2006).

## Fonemas

Los fonemas (del griego *φώνημα*, sonido de la voz) son la menor articulación de un sonido vocálico y consonántico. Los fonemas no son sonidos con entidad física, sino abstracciones mentales o abstracciones formales de los sonidos del habla. En este sentido, un fonema puede ser representado por una familia o clase de equivalencia de sonidos (técnicamente denominados fonos), que los hablantes asocian a un sonido específico durante la producción o la percepción del habla. Así por ejemplo en español el fonema /d/ [+ obstruyente, + alveolar, + sonoro] puede ser articulado como oclusiva [d] a principio de palabra o tras nasal o pausa larga, pero es pronunciado como aproximante [ð] entre vocales o entre vocal y líquida, así /dedo/ se pronuncia [deðo] donde el primer y tercer sonido difieren en el grado de obstrucción aunque son similares en una serie de rasgos, por ejemplo los propios del fonema (Ladefoged, 2005).

## Difonos

Un difono representa el sonido que abarca desde la mitad de la realización de un fonema hasta la mitad de la realización del fonema siguiente. El propósito de esta unidad de sonido es incorporar a la unidad de síntesis la transición de sonido entre fonemas, hecho que había causado tanta dificultad en los sistemas iniciales. La síntesis consiste, entonces, en la concatenación de segmentos de señal en el tiempo, siendo los segmentos difonos (Ladefoged, 2005).

## Prosodia

La prosodia es una rama de la lingüística que analiza y representa formalmente aquellos elementos de la expresión oral, tales como el acento, los tonos y la entonación. La prosodia trata la manifestación concreta en la producción de las palabras desde el punto de vista fonético-acústico, la variación de la frecuencia fundamental, la duración y la intensidad que constituyen los parámetros prosódicos físicos.

Puede dividirse convenientemente en dos aspectos. El primero considera aspectos suprasegmentales, es decir, que trata la entonación de la frase en su conjunto. El segundo controla la melodía, fenómenos locales de coarticulación, acentuación.

La prosodia se generaba mediante sistemas basados en reglas obtenidas a partir de estudios lingüísticos y retocados empíricamente hasta conseguir un habla sintética aceptable. Actualmente se empiezan a utilizar métodos estadísticos sobre bases de datos para generar automáticamente modelos prosódicos.

### 1.2. Procesamiento de texto a voz

Recientes progresos en la síntesis de voz han producido sintetizadores con mejor inteligibilidad, pero el sonido y la naturalidad aún siguen siendo un problema. Un elemento para juzgar la calidad de la síntesis de voz es su parecido con la voz humana y su potencialidad para ser entendida. Estos sistemas pueden lograr que personas con discapacidad visual, problemas de lectura o que se encuentren en actividades en las cuales los ojos y manos están ocupados puedan escuchar instrucciones.

Un sintetizador de voz convierte el lenguaje escrito en habla, por esta característica también es conocido como sistema TTS *(Text To Speech).* El habla sintetizada se genera concatenando segmentos de grabaciones que se encuentran almacenados en una base de datos. Los sistemas *Text To Speech* difieren en diversos aspectos, uno de ellos es el tamaño de las unidades de habla almacenadas. Los sistemas que almacenen fonemas y difonemas proveen el rango de salida más amplio, sin embargo, es posible que su calidad sea baja. De forma alternativa, un sintetizador puede incorporar un modelo del tracto vocal y otras características de la voz humana para generar una voz completamente “sintética” o “electrónica” (Birkholz, y otros, 2013).

El procedimiento de síntesis de texto a voz consiste de dos fases principales. La primera fase es el análisis del texto, donde la cadena de caracteres de entrada es transcrita en una fonética o algunas otras representaciones lingüísticas. La segunda etapa es la generación de formas de onda de voz. Estas dos fases son usualmente llamadas síntesis de alto y bajo nivel.

La Figura 1 muestra una versión simplificada del proceso de síntesis de voz, la entrada del texto puede ser por ejemplo de un procesador de palabras. La cadena de caracteres es entonces procesada y analizada en una representación fonética la cual es usualmente una cadena de fonemas con alguna información adicional para la correcta entonación, duración y énfasis. Finalmente, con el sintetizador de bajo nivel el sonido de la voz es generado por la información de un sintetizador de alto nivel (Lemmetty, 1999).

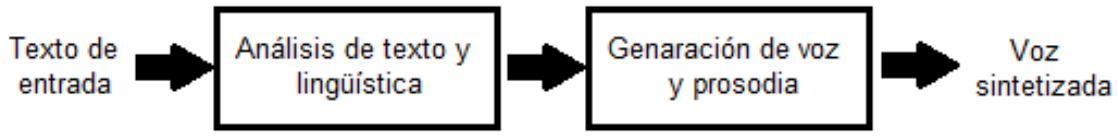



Figura 1. Procedimiento de síntesis de voz (Lemmetty, 1999)

En la Figura 2 se detalla el procesamiento de texto a voz de los sistemas TTS comunes. Se puede observar que existen dos bloques principales que forman el sistema: el bloque de Procesamiento de Lenguaje Natural (Natural Language Processing o NLP) y el bloque de Proceso de síntesis.

El bloque de NLP se encarga de producir una transcripción fonética del texto leído, además de la entonación y el ritmo deseados para la voz de salida. Después el bloque de Proceso de síntesis transforma la información simbólica que recibe del bloque anterior, en una voz de salida.

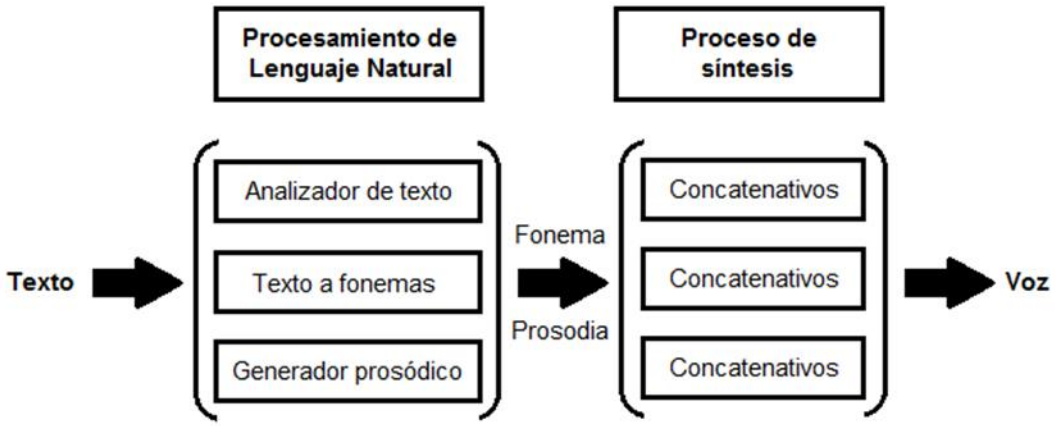



Figura 2. Arquitectura común para los sistemas TTS (Lemmetty, 1999)

El texto de entrada consiste en una colección de símbolos que deben interpretarse como palabras con el fin de tener la idea de lo que se ha escrito. La explicación de cada bloque del NLP se muestra a continuación:

### Analizador de texto

La función principal del analizador de texto es tomar como entrada cualquier texto y darle el formato adecuado para que sea entendible por el siguiente módulo. El formato que se desea obtener es una secuencia de palabras libres de ruido. Además, el analizador de texto realiza la asignación de pausas

entre frases.

### Convertidor de texto a fonemas

Después del análisis de texto, el siguiente paso es convertir de texto a fonemas. Una vez que se obtienen las palabras, es necesario asignar su pronunciación para poder generar la señal de voz. Estas pausas se usan para organizar las palabras en un número adecuado de frases. La transcripción fonética es la asociación de una palabra con los fonemas que lo comprenden. Algunas transcripciones fonéticas se muestran en la tabla.

Tabla 1 Ejemplos de transcripciones fonéticas del español (Lemmetty, 1999)

| **Palabra** | **Transcripción fonética** |
|---|---|
| **español** | **/e/ /s/ /p/ /a/ /ny/ /o/ /l/** |
| **pastel** | **/p/ /a/ /s/ /t/ /e/ /l/** |
| **cinco** | **/s/ /i/ /N/ /kc/ /k/ /o/** |
| **ardilla** | **/a/ /r/ /d/ /i/ /dZc/ /dZ/ /a/** |
| **blanco** | **/bc/ /b/ /l/ /a/ /N/ /kc/ /k/ /o/** |

La idea básica de un sistema TTS consiste en convertir un texto de entrada en una secuencia de fonemas correspondientes a la transcripción fonética de ese texto.

### Generador prosódico

Una de las metas principales en el proceso de TTS es la producción de voz con la mayor naturalidad posible. El generador prosódico se encarga de asignar la duración correcta a cada uno de los fonemas, así como una entonación adecuada.

## 1.3. Tecnologías del sintetizador

Las cualidades más importantes de los sistemas de síntesis de voz son la "naturalidad" y la "inteligibilidad". La naturalidad se refiere a cuánto suena como la voz de una persona real, mientras que la inteligibilidad a la facilidad de poder ser entendida. El sintetizador de voz ideal es tanto natural como inteligible. Los sistemas de síntesis de voz usualmente tratan de aumentar estas características (Gahlawata, y otros, 2014).

Las dos tecnologías primarias que generan formas de ondas sintéticas de voz son la "síntesis concatenativa" y la "síntesis de formantes". Cada tecnología tiene sus fortalezas y debilidades, dependiendo de su uso se podrá determinar qué acercamiento será usado (Schröder, 2001).

### 1.3.1 Síntesis concatenativa

La síntesis concatenativa se basa en la concatenación de segmentos de voz grabados. Generalmente, la síntesis concatenativa produce los resultados más naturales. Sin embargo, la variación natural del habla y las técnicas automatizadas de segmentación de formas de onda resultan en defectos audibles, que conllevan una pérdida de naturalidad (Zhang, 2011). Hay tres tipos básicos de síntesis concatenativa:

***Tipo 1. Síntesis de selección de unidades***

Emplea bases de datos de voces grabadas. Durante la creación de la base de datos, cada enunciado grabado es segmentado en: fonos, difonos, medios fonos, sílabas, morfemas, palabras, frases y oraciones. Normalmente la división en segmentos es hecha con ayuda de un sistema de reconocimiento del habla modificado, usando representaciones visuales como la forma de onda y un espectrograma (Black, 2002). Un índice de las unidades de voz en la base de datos es creado basado en la segmentación y en parámetros acústicos como la frecuencia fundamental (tono), duración, posición de la sílaba y fonemas cercanos. Durante el tiempo de ejecución, el enunciado deseado es creado determinando la mayor cadena posible de unidades (selección de unidades) (Yang Wang, y otros, 2011).

La selección de unidades permite una naturalidad mayor debido a que emplea un menor procesamiento digital de señales (DSP) en el habla grabada. El procesamiento digital de señales usualmente ocasiona que el sonido de la voz no sea tan natural, aunque algunos sistemas emplean una pequeña cantidad de procesamiento de la señal en el punto de la concatenación para ajustar la forma de onda. El audio de salida de la mejor selección de unidades usualmente es indistinguible de las voces humanas reales, especialmente en contextos con sistemas TTS. Sin embargo, una mayor naturalidad requiere de bases de

datos de selección de unidades muy grandes, en algunos sistemas llegando a ser de gigabytes de datos grabados, representando docenas de horas de voz (Kominek, y otros, 2003).

***Tipo 2. Síntesis de difonos***

La síntesis de difonos usa una pequeña base de datos de voz que contiene todos los difonos que ocurren en el lenguaje. Solo un ejemplo de cada difono es almacenado en la base de datos de voces. El sintetizador genera la voz combinando dichos difonemas de acuerdo con la transcripción generada. La calidad del discurso que resulta no es generalmente tan buena como la de la síntesis de selección de unidades, pero más natural comparada con la salida de los sintetizadores de formantes. El uso de la síntesis de difonos en aplicaciones comerciales ha disminuido aunque sigue siendo utilizada en la esfera de la investigación debido al número de aplicaciones de software gratuitas (Dutoit, y otros, 1996).

***Tipo 3. Síntesis de dominio específico***

La síntesis de dominio específico concatena palabras y frases pre-grabadas para crear enunciados completos. Es usada en aplicaciones donde la variedad de los textos del sistema está limitada a una salida de audio en un dominio particular, como los anuncios en un calendario de tránsito o reportes del clima (Lamel, y otros, 1993). La tecnología es muy simple de implementar y ha sido empleada de manera comercial por varios años en dispositivos como calculadoras o relojes parlantes.

El nivel de naturalidad de estos sistemas puede ser muy alto debido a que la variedad de los tipos de oraciones está limitada y logran estar muy cerca de la entonación de las grabaciones originales. Debido a que estos sistemas están limitados por las palabras y frases en sus bases de datos, no son empleados para propósitos generales, solo pueden sintetizar combinaciones de palabras y frases para los que han sido programados.

### 1.3.2. Síntesis de formantes

La síntesis de formantes no utiliza muestras de voz humana durante el tiempo de ejecución. En su lugar, el audio de salida es creado a partir de la síntesis aditiva y un modelo acústico (síntesis mediante modelado físico). Parámetros como la frecuencia fundamental, fonación y niveles de ruido son variados a través del tiempo para crear una forma de onda de una voz artificial. Algunas veces es llamado síntesis basada en reglas. Sin embargo, existen sistemas de concatenación que también tienen *plugins* basados en reglas.

Varios sistemas basados en la tecnología de síntesis de formantes generan una voz artificial con sonido robótico que no podría ser confundida con la voz humana. Sin embargo, la naturalidad máxima no es el objetivo de los sistemas de síntesis de voz, los sistemas de síntesis de formantes tienen ventajas sobre otros sistemas de concatenación. El habla a través de la síntesis de formantes puede ser inteligible, inclusive a grandes velocidades. El habla sintetizada a grandes velocidades es usada por personas con dificultades visuales para navegar de maneras más fluida en computadoras usando un lector de pantalla. Los sintetizadores de formantes son programas pequeños en comparación a los sistemas de concatenación debido a que no tienen una base de datos de muestras de voz.

La síntesis de formantes puede ser empleada en sistemas embebidos donde la memoria y el poder del microprocesador son limitados. Debido a que los sistemas basados en formantes tienen completo control sobre todos los aspectos del audio de salida, una amplia variedad de entonaciones pueden ser generadas para transmitir no solo preguntas o declaraciones, sino una variedad de emociones y entonaciones en la voz (Music and Computers, 1993).

Para la implementación del *plugin* se utilizó la tecnología de síntesis concatenativa, específicamente la síntesis de difonos que es la desarrollada por la biblioteca QtTextToSpeech, con la cual trabaja el *plugin*.

## 1.4. Análisis de soluciones similares

Actualmente existen varias herramientas de conversión de texto a voz, ya sea comerciales o de libre distribución, entre los que se encuentran:

### Assistive Context-Aware Toolkit (ACAT)

Kit de herramientas sensible al contexto asistencial (ACAT). Es una plataforma de código abierto desarrollado en los laboratorios de Intel. Permite a las personas con enfermedades de las neuronas motoras y otras discapacidades tener acceso completo a las capacidades y aplicaciones de sus ordenadores a través de interfaces adecuadas para su condición. Específicamente, la ACAT permite a los usuarios comunicarse fácilmente con otros a través de la simulación de teclado, la predicción de palabras y síntesis de voz. Los usuarios pueden realizar una serie de tareas tales como la edición, la gestión de documentos, navegar por la Web y acceder a mensajes de correo electrónico.

ACAT es útil para los desarrolladores de Microsoft Windows que estén interesados en el desarrollo de tecnologías de apoyo a las personas con esclerosis lateral amiotrófica (ELA) o discapacidades similares.

Es utilizada también por los investigadores que trabajan en nuevas interfaces de usuario y en nuevas modalidades de detección o predicción de palabras (Intel Corporation, 2016).

**Loquendo**

Loquendo TTS es un programa informático que hace la función de síntesis del habla. Proporciona voces reales para los datos dinámicos y funciona en diversas aplicaciones de voz, también ofrece voces naturales con capacidades para aplicaciones multimodales de voz. El TTS de Loquendo puede sintetizar idiomas y voces distintas simultáneamente. La existencia de un léxico de usuario asegura que vocabularios especializados, abreviaciones, acrónimos e incluso entonaciones regionales estén pronunciados correctamente. El diccionario usado asegura que los términos especializados de vocabulario, las abreviaciones, las siglas y también las diferencias regionales en la pronunciación suenen en el justo modo y en el momento en el que el desarrollador las crea. Los sistemas operativos soportados son: MS Windows (XP, Vista, 7, 8, 8,1, 10); Linux Red Hat Enterprise 3, 4, 5*; SUSE Linux 10 (Loquendo, 2012).

**IVONA TTS**

IVONA es un sistema de síntesis de voz en varios idiomas desarrollado en Polish IT, compañía de IVO software. IVONA utiliza la técnica de selección de unidades, este tipo de síntesis utiliza grandes bases de datos de voz grabada. Durante la creación de las bases de datos cada frase grabada se segmenta en otras como tonos individuales, sílabas, morfemas, palabras, frases y oraciones (IVONA, 2012).

La división en segmentos se realiza usando un reconocedor de voz especialmente modificado. Un índice de las unidades de voz en la base de datos se crea sobre la base de la segmentación y los parámetros acústicos como la frecuencia fundamental (tono) o la duración. En tiempo de ejecución, la emisión objetivo deseada se crea mediante la determinación de la mejor cadena de unidades candidato de la base de datos (selección de unidades). IVONA es compatible con los sistemas basados en Windows, Unix, Android, Tizen, OS (Amazon Company, 2016).

## 1.5. Observación de las soluciones existentes

Los sistemas como ACAT, Loquendo e IVONA TTS guardan estrecha relación con el objetivo de la investigación, permitiendo identificar un conjunto de funcionalidades básicas que debe poseer la aplicación que se desea implementar. Dichos sistemas solo podrían servir de base para la identificación del objetivo antes mencionado, no para su utilización, ya que estos sistemas no cumplen los requerimientos y especificaciones necesarias para la utilización en la plataforma PRIMICIA. Un ejemplo

claro de por qué no se pueden utilizar dichos sintetizadores es que no poseen la funcionalidad de cargar los textos de una base de datos. En el caso del conjunto de herramientas ACAT fue desarrollado en C # utilizando Microsoft Visual Studio * 2012 y .NET 4.5, lo que lo hace incompatible con la plataforma, teniendo en cuenta que el lenguaje en que se desarrolla PRIMICIA es C++. Por otra parte, posee código abierto sirviendo de apoyo para el desarrollo de la solución. Después del análisis realizado sobre los sintetizadores estudiados, se puede decir que no existe ninguno dedicado al tema específico en cuestión, aunque constituyen buenos ejemplos para el desarrollador.

## 1.6. Conclusiones parciales

- El análisis del funcionamiento de un sintetizador y de los tipos de sintetizadores sirvió de base para la conceptualización y posterior desarrollo del sintetizador de voz para PRIMICIA.

-El análisis de los principales conceptos asociados al dominio del problema permitió tener un mejor entendimiento y comprensión de las temáticas que conforman la investigación así como de los factores que influyen en la situación problemática actual.

-El estudio de las soluciones existentes contribuyó a concluir que no existe un sintetizador de voz a nivel nacional ni internacional que se adecue a las necesidades del proyecto PRIMICIA.

# Capítulo 2
# Herramientas y tecnologías

## Introducción

El crecimiento alcanzado en el desarrollo de software es inmenso, todo el personal involucrado en este proceso se ha visto en la obligación de estandarizar el trabajo de las aplicaciones que se crean. Se han desplegado una serie de metodologías para el desarrollo de software lo que posibilita que todos los miembros del proyecto puedan entenderse utilizando el mismo estándar. Además, se han desarrollado diferentes lenguajes de modelación orientado a objetos como es el UML. Las herramientas CASE (Computer Aided/Assisted Software/System Engineering) o (Ingeniería de Software Asistida por Computadora) no se han quedado atrás y al igual que las metodologías y lenguajes de modelado han tenido un amplio desarrollo.

En este capítulo se hace un estudio de metodologías de desarrollo de software, lenguajes de modelado y herramientas para la implementación de software y se proponen las más adecuadas en la elaboración de la solución propuesta.

## 2.1. Metodología de Desarrollo de Software

Las metodologías y estándares utilizados en el desarrollo de software proporcionan las guías para poder conocer todo el camino a recorrer desde antes de empezar la implementación, con lo cual se asegura la calidad del producto final, así como también el cumplimiento en la entrega del mismo en un tiempo estipulado. Cada una de las metodologías que existen tiene características específicas que las hacen singulares, por lo que clasificarlas sería una tarea engorrosa. A partir de su filosofía de desarrollo se pueden separar en pesadas o tradicionales y ligeras o ágiles que a continuación se describen. Entre las metodologías más utilizadas se encuentran Rational Unified Process (RUP) y Programación Extrema (XP), de las que se abordan algunas características en este epígrafe para poder argumentar así la selección realizada.

### 2.1.1. Metodologías Tradicionales de Desarrollo

Las metodologías tradicionales o pesadas de desarrollo de software se guían expresamente a través de un plan durante el proceso de desarrollo, en el cual se generan gran cantidad de artefactos y roles mediante un modelado y documentación detallada, apoyándose en poderosas herramientas de trabajo y notaciones definidas en la arquitectura del software. Las metodologías tradicionales se caracterizan por estar enfocadas principalmente al proceso definido por la metodología y no a las características del proyecto que se desarrolla (Centers for medicare & medicaid services, 2008).

A continuación se exponen algunas de las características fundamentales de una de las metodologías tradicionales que cuenta con un gran prestigio en su clasificación.

### Rational Unified Process (RUP)

La metodología RUP, llamada así por sus siglas en inglés *Rational Unified Process*, es una metodología cuyo fin es entregar un producto de software. Permite estructurar todos los procesos y medir la eficiencia de la organización. Es un proceso de desarrollo de software que utiliza el lenguaje unificado de modelado UML. Constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Proporciona disciplinas en las cuales se encuentran artefactos con los que se puede contar como guías para poder documentar e implementar de una manera fácil y eficiente (Molpeceres, 2002).

Según RUP el desarrollo del software se divide en cuatro fases: Inicio, Elaboración, Construcción y Transición; cada una de las cuales es desarrollada mediante ciclos de iteraciones.

-Inicio: El objetivo en esta etapa es determinar la visión del proyecto.

-Elaboración: En esta etapa el objetivo es determinar la línea base de la arquitectura.

-Construcción: En esta etapa el objetivo es obtener la capacidad operacional inicial.

-Transición: En esta etapa el objetivo es llegar a obtener el realce del producto.

Cada una de estas etapas es desarrollada mediante ciclos de iteraciones, lo cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

En RUP se han agrupado las actividades en grupos lógicos definiéndose nueve flujos de trabajo principales. Los seis primeros son conocidos como flujos de ingeniería o proceso y los tres últimos como de apoyo o soporte.

Estos se observan en la figura que se muestra a continuación:

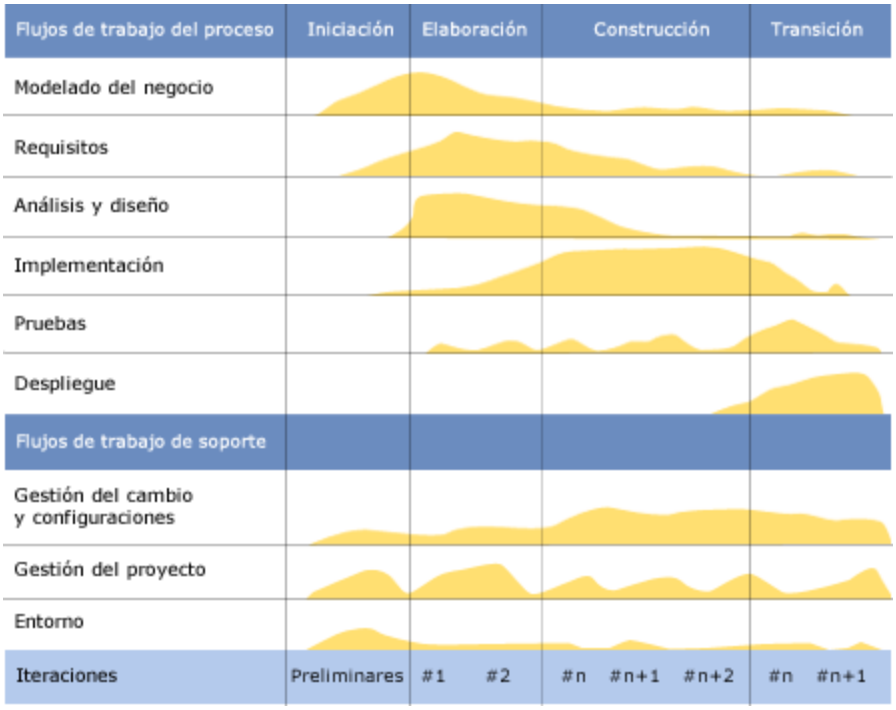



Figura 3. Disciplinas, Fases e Iteraciones de la Metodología RUP (Modelos de software, 2011)

RUP es un proceso que está definido por tres características esenciales: dirigido por los casos de uso, iterativo e incremental y centrado en la arquitectura.

**Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requisitos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

**Centrado en la arquitectura:** La arquitectura involucra los elementos más significativos del sistema y está influenciada, entre otros, por sistemas operativos, gestores de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados y requisitos no funcionales.

La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso (CU) relevantes desde el punto de vista de la arquitectura. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

Todas las vistas juntas forman el llamado modelo 4+1 de la arquitectura, recibe este nombre porque lo forman las vistas lógica, de implementación, proceso y despliegue, más la de casos de uso que es la que da cohesión a todas.

**Iterativo e Incremental:** Para hacer más manejable un proyecto se recomienda dividirlo en ciclos. RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Por ejemplo, una iteración de elaboración centra su atención en el análisis y diseño, aunque refina los requisitos y obtiene un producto con un determinado nivel, pero que irá creciendo incrementalmente en cada iteración.

Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada, es por eso que se dice que son miniproyectos" ( Pressman, 2010).

RUP está basado en 5 principios:

**Adaptar el proceso:** El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico, aunque se debe tener en cuenta el alcance del proyecto.

**Balancear prioridades:** Debe encontrarse un balance que satisfaga los deseos de todos.

**Demostrar valor iterativamente:** Los proyectos se entregan en etapas iteradas. En cada iteración se analiza la opinión, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

**Elevar el nivel de abstracción:** Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes de cuarta generación (SQL, lenguajes de consulta) o esquemas (frameworks). Esto facilita a los ingenieros de software dirigirse directamente de los requisitos a la

codificación de software, en dependencia de las necesidades del cliente. Un nivel alto de abstracción también permite discusiones sobre diversos niveles arquitectónicos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.

**Enfocarse en la calidad:** El control de calidad debe estar presente en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

Las actividades de RUP se centran en crear y mantener modelos utilizando UML. Como no existe un único proceso que sea apropiado para todos los desarrollos, RUP es un proceso configurable. Se adapta tanto a grupos pequeños de desarrollo como a grandes organizaciones. Basándose en lo que se consideran las mejores prácticas de desarrollo de software, RUP resulta apropiado para una amplia gama de proyectos y organizaciones.

RUP aplica 6 prácticas principales para el desarrollo de sistemas. Estas prácticas son:

**Desarrollo de software en forma iterativa:** permite ir creciendo en el entendimiento del problema a través de refinamientos sucesivos. Esto también permite introducir cambios tácticos en los requisitos, características del sistema o en los tiempos.

**Gestión de requisitos:** es necesario para garantizar que al final el software tenga la calidad requerida.

**Uso de arquitecturas basadas en *plugin*s:** el uso de una arquitectura basada en *plugin*s hace que el sistema pueda reutilizar *plugin*s desarrollados con antelación y facilitar el trabajo que se realiza.

**Modelización visual del software:** El proceso le demuestra cómo modelar visualmente software para capturar la estructura y el comportamiento de arquitecturas y de *plugin*s.

**Verificación de calidad del software:** RUP le asiste en el planeamiento, el diseño, la puesta en marcha, la ejecución, y la evaluación de las pruebas de confiabilidad, funcionalidad, performance de la aplicación y el sistema.

**Control de cambios:** La capacidad de manejar los cambios, asegurándose que cada cambio sea aceptable, y pudiendo continuar con los mismos, es esencial en un ambiente en el cual el cambio es inevitable. (Jacobson, y otros, 2000).

### 2.1.2. Metodologías Ágiles de Desarrollo

*"Las metodologías ágiles se enfocan generalmente en intentar evadir las burocráticas guías de las metodologías tradicionales"* (Martínez, 2005) .Centran sus procesos de desarrollo a los individuos que interactúan con el proyecto así como con el equipo de trabajo, obteniendo buenos resultados, es decir, se desarrolla un buen software en cuanto a su funcionamiento pero no genera una documentación consistente. Las metodologías ágiles están basadas en heurísticas provenientes de prácticas de producción de código y el cliente es parte del equipo de desarrollo, generan pocos artefactos y poseen pocos roles. A continuación se exponen algunas de las características de una de estas metodologías.

### Programación Extrema (XP)

Extreme Programming (XP), pertenece al grupo de metodologías ágiles, donde lo principal es la comunicación entre el personal implicado y la obtención de un resultado deseado en el menor tiempo posible.

Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizada para proyectos de corto plazo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

En esta metodología se realizan pruebas unitarias, que sirven como método para adelantarse a obtener los posibles errores, se utiliza la refabricación, lo cual se basa en la reutilización de código y la programación en pares, es decir, dos desarrolladores trabajando en una misma estación de trabajo, lo cual constituye una peculiaridad de XP.

Esta metodología le da el derecho al usuario de decidir qué se implementa, saber el estado real y el progreso del proyecto, añadir, cambiar o quitar requisitos en cualquier momento. La metodología le permite a los desarrolladores decidir cómo se implementan los procesos, pedir al cliente en cualquier momento aclaraciones de los requisitos, estimar el esfuerzo para implementar el sistema, y cambiar los requisitos en base a nuevos descubrimientos.

Las principales características de esta metodología son: la comunicación entre los usuarios y los desarrolladores, la simplicidad al desarrollar y codificar los módulos del sistema y la retroalimentación concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales. Esta metodología empieza en pequeño y añade funcionalidad con retroalimentación continua, donde el manejo del cambio se

convierte en parte sustantiva del proceso y el costo del cambio no depende de la fase o etapa. No introduce funcionalidades antes que sean necesarias además que el cliente o el usuario se convierte en miembro del equipo" (Molpeceres, 2002).

## Agile Unified Process (AUP) - UCI

El Proceso Unificado Ágil (AUP) es una versión simplificada de RUP. Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. En la UCI se decide hacer una variación de esta metodología, de forma tal que se adapte al ciclo de vida definido para la actividad productiva en la universidad, logrando estandarizar el proceso de desarrollo de software. De las cuatro fases que propone AUP ( Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos en la UCI mantener la fase de Inicio pero modificando su objetivo, se unifican las restantes 3 fases de AUP en una sola titulada Ejecución y se agrega la fase de Cierre (Sánchez, 2015).

A continuación se muestra la descripción de las fases y algunas técnicas.

Tabla 2. Descripción de las fases de AUP (Sánchez, 2015)

| Fases de AUP | Fases Variación AUP-UCI | Objetivos de las fases (Variación AUP-UCI) |
|---|---|---|
| Inicio | Inicio | En esta fase se realiza un estudio inicial de la organización cliente que permite obtener una información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto. |
| Elaboración | Ejecución | En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. |
| Construcción | | |
| Transición | | |

| | | |
|---|---|---|
| | Cierre | En esta fase se analizaron tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto. |

### Metodología seleccionada

El plugin de síntesis de voz a desarrollar es sencillo en cuanto a su volumen y su implementación, se debe realizar en un tiempo relativamente corto y limitado. Teniendo en cuenta estas características y la definición de metodología ágil y robusta que se detalla anteriormente, lo más indicado es utilizar una metodología ágil.

*Extreme Programming* (XP) es una metodología ágil, pero para trabajar con ella se recomienda tener experiencia. Se basa en la capacidad y madurez del personal, característica que descalifica su uso para guiar el desarrollo del *plugin* de síntesis de voz, porque el personal tiene poca experiencia en cuanto al trabajo con ella. De ahí que es necesario seleccionar una metodología ágil con la que el personal tenga experiencia o que se asemeje a otras metodologías que el personal domine.

Teniendo en cuenta lo que se expresa anteriormente, se selecciona para guiar el proceso de desarrollo del *plugin* de síntesis de voz la versión ágil de la metodología RUP denominada AUP en su variación para la UCI. Esta metodología se utilizó por las características que presenta y por ser la utilizada para guiar el proceso de desarrollo del software en el proyecto, permitiendo una mejor compatibilidad en la documentación.

## 2.2. Lenguaje Unificado de Modelado (UML)

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar parte de un diseño de software orientado a objetos. Muchas organizaciones los usan en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores. El uso de un lenguaje de modelado es más sencillo que la auténtica programación, pues existen menos medios para verificar efectivamente el funcionamiento adecuado del modelo.

UML es considerado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante este es posible establecer las estructuras necesarias para plasmar un sistema de software previo al

proceso intensivo de escribir código. UML se ha convertido en uno de los más utilizados por lo expresivo, claro y uniforme que resulta para el diseño orientado a objetos. Permite una fuerte integración entre las herramientas, los procesos y los dominios. Posibilita el intercambio de modelos entre las distintas herramientas CASE y mejora el tiempo total de desarrollo, además presenta una alta reutilización y disminución de costos (Fowler, y otros, 1999).

El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos y describe la semántica esencial de lo que estos diagramas y símbolos significan. Sus principales funciones son las de Visualizar, Especificar, Construir y Documentar. Un modelo UML está compuesto por tres clases de bloques de construcción: Elementos, Relaciones y Diagramas (Fowler, y otros, 1999).

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

- Los Diagramas de Estructura enfatizan en los elementos que deben existir en el sistema modelado como son los Diagrama de clases, Diagrama de *plugins*, Diagrama de objetos, Diagrama de estructura compuesta, Diagrama de despliegue y Diagrama de paquetes.

-Los Diagramas de Comportamiento enfatizan en lo que debe suceder en el sistema modelado, por ejemplo: Diagrama de actividades, Diagrama de casos de uso y Diagrama de estados.

-Los Diagramas de Interacción son un subtipo de diagramas de comportamiento que enfatizan sobre el flujo de control y de datos entre los elementos del sistema modelado como es el Diagrama de secuencia, Diagrama de comunicación que es una versión simplificada del Diagrama de colaboración, Diagrama de tiempos y Diagrama de vista de interacción (Zuluaga Giraldo, 2011).

UML permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos. También documenta todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, entre otros.). Este lenguaje de modelado es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental (Stevens, y otros, 2012).

## 2.3. Herramientas CASE

Para superar las dificultades existentes en el uso de las diversas tecnologías, la industria de computadoras ha desarrollado un soporte automatizado para el desarrollo y mantenimiento de software. Este es llamado Computer Aided Software Engineering (CASE) (Guerra, 2009).

Las herramientas CASE son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores durante todos los pasos del ciclo de vida de un software. Los estados en el ciclo de vida de desarrollo de un software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación.

Una herramienta CASE se utiliza para ayudar a las actividades del proceso de software que es utilizado para diseñar e implementar otro software. Las herramientas Case facilitan mejoras en la calidad y productividad del diseño y desarrollo. Estas herramientas proporcionan a los desarrolladores de software grandes ventajas: apoyan a las metodologías y métodos, mejoran la comunicación entre las personas que interactúan con el sistema permitiéndoles compartir mejor su trabajo, establecen métodos eficientes para almacenar y utilizar los datos, hacen más eficaz el mantenimiento y automatizan fragmentos del análisis y diseño pesados y vulnerables a errores (Guerra, 2009).

A continuación se abordan las características de dos de estas herramientas para así seleccionar una de ellas que garantice una mayor eficiencia en el desarrollo del software.

### Rational Rose

Es una herramienta software para el modelado visual mediante UML de sistemas software. Permite especificar, analizar y diseñar el sistema antes de codificarlo. Es un agrupamiento de metodologías y herramientas que abarca todos los aspectos del desarrollo de software, desde su concepción hasta la elaboración del producto.

Entre sus principales características se encuentran: Soporte para análisis de patrones ANSI C++, Rose J y Visual C++. Cuenta con características de control por separado de *plugins* que permite una administración más granular y el uso de modelos. Permite la generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables. Soporta Enterprise Java Beans 2.0. Tiene capacidad de análisis de calidad de código. Cuenta además con modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requisitos de aplicación a través de diseños lógicos y físicos (Murillo Alfaro, 1999).

Rational Rose Enterprise Edition es una herramienta que se puede encuadrar dentro del grupo de herramientas más técnicas debido a que se encarga de llevar a cabo tanto la automatización de los sistemas para la posterior generación de código, como para labores de ingeniería inversa. Sin duda Rational Rose Enterprise Edition es una forma de ayuda para la comprensión del sistema y de sus distintos *plugins*. Lo mejor es que se puede aplicar ingeniería inversa a una multitud de códigos distintos, siempre que estén orientados a objetos. La clave está en la creación de *plugins*, los cuales van a contener una serie de archivos dentro de los cuales se encuentran las distintas clases que pertenecen a dicho plugin. Mediante la especificación de la sintaxis que presentan dichos ficheros, se realiza de forma automática la ingeniería inversa. Rational Rose presenta una pequeña desventaja, y es que necesita de mucha memoria para poder de alguna forma ser manejado de forma rápida y eficiente.

### Visual Paradigm

Visual Paradigm es una herramienta con un diseño centrado en casos de uso y enfocado al negocio que genera un software de calidad, tiene la particularidad de ser un lenguaje común para todo el equipo de desarrollo y eso facilita la comunicación. Posee capacidades de ingeniería directa e inversa, su modelo y el código se encuentran sincronizados en todo el ciclo de desarrollo y está disponible en múltiples versiones y plataformas. Es muy personalizable, tiene un generador de mapeo de objetos-relacionales para los lenguajes de programación Java, .NET y PHP. La versión gratuita no permite realizar ingeniería inversa, pero permite crear diagramas y generar código a partir de ellos.

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (Visual Paradigm, 2013).

Visual Paradigm Studio cuenta con las siguientes características: Soporte de UML versión 2.1, Diagramas de Procesos de Negocio, documento modelado colaborativo y Subversión, interoperabilidad con modelos UML 2.1, código a modelo, código a diagrama, C++, generación de código, editor de Detalles de Casos de Uso. Otras de sus características son: los diagramas de flujo de datos, generador de informes para generación de documentación, distribución automática de diagramas, soporte ORM que no es más que la generación de objetos Java desde la base de datos, generación de bases de datos, entre otras (Visual Paradigm, 2013).

## 2.4. Visual Paradigm 8.0 como base en el desarrollo de la solución

Visual Paradigm 8.0 es una herramienta que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Además de permitir diseñar todos los tipos de diagramas que se necesitan en esta investigación. Permite generar la documentación necesaria para el posterior diseño e implementación del sistema. Tiene también un diseño centrado en casos de uso y enfocado al negocio. Uno de los aspectos más importantes de la selección de esta herramienta es que en la Universidad de Ciencias Informáticas se está trabajando en función de utilizar tecnologías libres y Visual Paradigm 8.0 es una herramienta libre y multiplataforma.

## 2.5. Herramientas utilizadas

### Qt Creator

Qt Creator es un Entorno de Desarrollo Integrado (IDE) y se centra en aumentar la productividad de los desarrolladores que utilizan Qt con C++. Permite navegar rápidamente entre los archivos, métodos y clases con atajos de teclado rápidos. Su de *plugin* hace que sea abierto a la adición de nuevas herramientas y formatos de archivo (QT Creator, 2015).

### QT Framework

Qt es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con interfaz gráfica de usuario, así como para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos y consolas para servidores. Es un software libre y de código abierto donde participa tanto la comunidad de desarrolladores de QT como desarrolladores de Nokia y otras empresas.

Qt utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en otros lenguajes de programación. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales (Knoll, 2011).

### pgAdmin

La herramienta pgAdmin posee código abierto para la administración de bases de datos PostgreSQL y derivados (EnterpriseDB Postgre Plus Advanced Server y Greenplum Database). El pgAdmin se diseña para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL

hasta desarrollar bases de datos complejas. La interface gráfica soporta todas las características de PostgreSQL y hace simple la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Microsoft Windows, Linux, FreeBSD, Mac OSX y Solaris (pgAdmin, 2016).

### PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Sus características técnicas la hacen uno de los sistemas gestores de bases de datos más potentes y robustos del mercado. Su desarrollo comenzó hace más de 18 años y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

Las características más importantes y soportadas por PostgreSQL son: Es un sistema gestor de base de datos 100% ACID, integridad referencial, replicación asincrónica/sincrónica, copias de seguridad en caliente. Es Unicode y posee juegos de caracteres internacionales además de regionalización por columna. Tiene múltiples métodos de autentificación, acceso encriptado vía SSL, actualización in-situ integrada, completa documentación, licencia BSD. Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit (PosgreSQL-es, 2010).

### Symfony

Symfony es un completo *framework* diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo Vista Controlador. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Symfony fue diseñado para ajustarse a los siguientes requisitos: Fácil de instalar y configurar en la mayoría de plataformas y es independiente del sistema gestor de bases de datos. Su capa de abstracción y el uso de ORM (Doctrine 2, Propel) permiten cambiar con facilidad de Sistemas Gestores de Base de Datos (SGBD) en cualquier fase del proyecto. Utiliza programación orientada a objetos y características como los espacios de nombres. Sencillo de usar en la mayoría de casos, aunque es preferible para el desarrollo de grandes aplicaciones Web que para pequeños proyectos. Aunque utiliza MVC (Modelo Vista Controlador), tiene su propia forma de trabajo en este punto, con variantes del MVC clásico como la capa de abstracción de base de datos, el controlador frontal y las acciones. Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador sólo debe configurar aquello que no es convencional. Sigue la mayoría de mejores prácticas y patrones de diseño para la web (Symfony, 2016).

**JQuery**

JQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC (Q-Success, 2016).

JQuery es un software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. JQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio ( The jQuery Foundation, 2016).

Entre sus principales características se encuentran: La manipulación de la hoja de estilos CSS, efectos y animaciones. Cuenta además con animaciones personalizadas, AJAX, soporta extensiones. Tiene varias utilidades como obtener información del navegador, operar con objetos y vectores, funciones para rutinas comunes, etc. Compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+.

**Bootstrap**

Bootstrap es un framework o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene HTML y plantillas de diseño basadas en CSS para la tipografía, formas, botones, navegación y otros *plugins* de la interfaz, así como extensiones de JavaScript opcionales. Su objetivo es facilitar el desarrollo de sitios web dinámicos y aplicaciones web. Es un framework web de extremo frontal, es decir, una interfaz para el usuario, a diferencia del código de servidor que reside en el *back-end* o servidor. Bootstrap es el segundo proyecto más galardonado con una estrella en GitHub, con más de 95K estrellas y más de 40K horquillas (Bootstrap, 2016).

Bootstrap es compatible con las últimas versiones del Google Chrome, Firefox, Internet Explorer, Opera, Safari y navegadores, aunque algunos de estos navegadores no son compatibles con todas las plataformas. Desde la versión 2.0 también es compatible con el diseño web sensible. Esto significa que el diseño de las páginas web ajusta dinámicamente, teniendo en cuenta las características del dispositivo que se utiliza (escritorio, tableta, teléfono móvil).

Desde la versión 3.0, Bootstrap adoptó un móvil por primera filosofía de diseño, haciendo hincapié en el diseño de respuesta por defecto. Bootstrap posee código abierto.

### Servidor HTTP Apache

El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la *Apache Software Foundation* dentro del proyecto HTTP Server. Presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración (The Apache Software Foundation, 2016).

Apache tiene amplia aceptación en la red desde 1996. Es el servidor HTTP más usado. Jugó un papel fundamental en el desarrollo fundamental de la *World Wide Web* y alcanzó su cuota más alta de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. En 2009 se convirtió en el primer servidor web que alojó más de 100 millones de sitios web (Netcraft, 2014).

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache. Algunas de las características que posee son: Modular, de código abierto, multiplataforma, extensible, popular (fácil conseguir ayuda/soporte).

**JetBrains PhpStorm**

JetBrains PhpStorm es un IDE comercial, multiplataforma para PHP desarrollado sobre la plataforma JetBrains *IntelliJ IDEA*. Proporciona un editor para PHP, HTML y JavaScript con el análisis de código en la marcha, la prevención de errores y refactorizaciones automáticas para PHP y el código JavaScript. La terminación del código de PhpStorm soporta PHP 5.3, 5.4, 5.5, 5.6 y 7.0. Incluye además un editor de SQL con resultados de la consulta editables (Zukerman, 2012). Los usuarios pueden ampliar el IDE mediante la instalación de *plugin*s creados para la plataforma de *IntelliJ* o escribir sus propios *plugin*s (PhpStorm, 2016).

## 2.6 Conclusiones parciales del capítulo

-Tras realizar el análisis de las metodologías de desarrollo de software usadas frecuentemente, así como de las características y particularidades de cada una de ellas, se puede concluir que la metodología a utilizar será el AUP-UCI.

-La integración de las tecnologías seleccionadas garantizará que se puedan construir cada uno de los artefactos que servirán de base para el posterior diseño e implementación del producto que se desea.

- El uso de herramientas de distribución gratuita y multiplataforma permitirá que no existe divergencia con las políticas de utilización de software libre que sigue el país.

# Capítulo 3

# Descripción de la solución propuesta

## Introducción

En este capítulo se exponen los artefactos generados que corresponden al flujo de trabajo de Modelado. Estos artefactos ayudan a comprender mejor el desarrollo de la solución propuesta. Permiten ver desde varias perspectivas a través de diagramas y descripciones todo el diseño del *plugin* para la síntesis de voz de las noticias. De esta forma se garantiza una buena base para el siguiente flujo de trabajo que es la implementación.

## 3.1. Modelo de dominio

El Instituto de Ingenieros Eléctricos y Electrónicos, sostiene que los modelos de dominio son representaciones de un dominio de aplicación que se puede utilizar para una variedad de objetivos operacionales en apoyo de tareas o procesos de ingeniería de software específicos (IEEE, 1990).

Por su parte Craig Larman plantea que "*el modelo del dominio muestra (a los modeladores) clases conceptuales significativas en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. (…) es una representación de las clases conceptuales del mundo real, no de plugins de software. No se trata de un conjunto de diagramas que describen clases software, u objetos software con responsabilidades*" (Larman, 2003).

En otro momento, el mismo autor señala, que *"todos los modelos son aproximaciones del dominio que estamos intentando entender. Un buen modelo del dominio captura las abstracciones y la información esenciales necesarias para entender el dominio en el contexto de los requisitos actuales y ayuda a la gente a entender el dominio -sus conceptos, terminología y relaciones"* (Larman, 2003).

En síntesis, el modelo de dominio es la representación visual de los conceptos u objetos de interés para el negocio, sus características y las relaciones entre ellos. Es el mecanismo fundamental para comprender el dominio del problema y para establecer conceptos comunes. Se emplea principalmente cuando la información tiene múltiples orígenes, ya que facilita el análisis y la comprensión.

En el caso de la presente investigación, se decide realizar un modelo de dominio ya que no se tienen bien definidos los procesos del negocio para la aplicación a desarrollar. A continuación, se muestra el modelo de dominio creado para dar solución al problema de la presente investigación:

### 3.2. Diagrama de clases del dominio

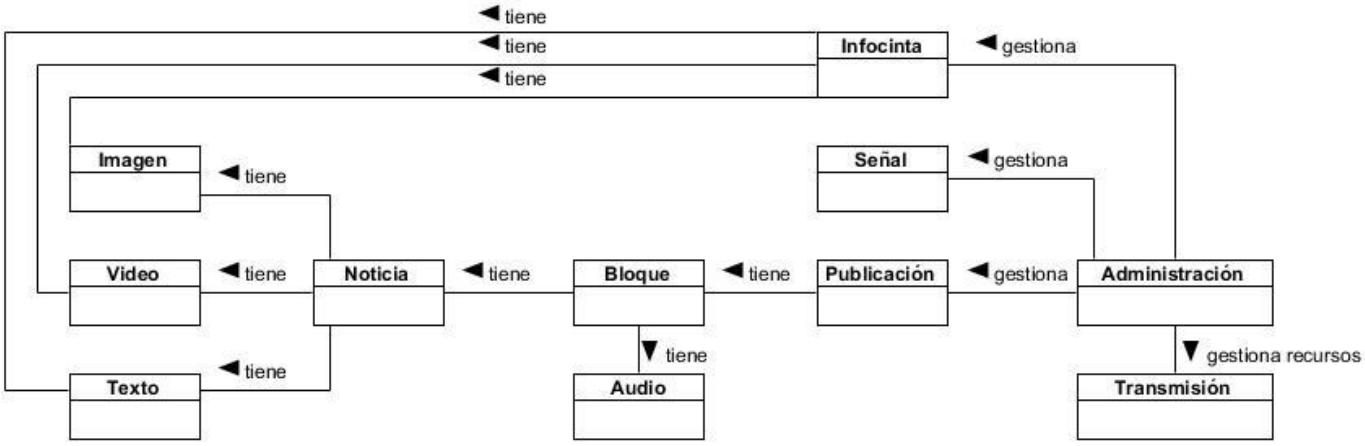



Figura 4. Diagrama de clases del dominio

### 3.3. Glosario de términos del dominio

**Transmisión:** Subsistema que transmite la información que se gestiona en el subsistema de administración por un canal de televisión.

**Administración:** Subsistema que gestiona los recursos que se transmiten.

**Publicación:** Se encarga de gestionar los bloques que van a ser transmitidos.

**Bloque:** Contiene el listado de noticias que se encuentran publicadas en el mismo.

**Noticia**: Conjunto de información que llega a las personas.

**Infocinta:** Elemento visual situado generalmente en el borde inferior de la pantalla, utilizado para transmitir una noticia importante sin necesidad de afectar la noticia principal que se esté transmitiendo.

**Imagen:** Archivo con formato de imagen que se almacena. Se utiliza para la confección de las noticias y las infocintas.

**Video:** Archivo con formato de video que se almacena. Se utiliza para la confección de las noticias e infocintas.

**Audio:** Archivo con formato de audio que se almacena. Se utiliza para la confección de los bloques.

**Texto:** Información que se utiliza para la confección de las noticias y las infocintas.

**Señal:** Señal de un canal externo que se enlaza y transmite, puede ser inmediata o programada.

### 3.4. Requisitos

Roger S. Pressman, considerado el padre de la Ingeniería de software, declaró que "*la ingeniería de requisitos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál es el impacto del software, qué es lo que el cliente final quiere y cómo interactuarán los usuarios finales con el software*" ( Pressman, 2010).

Dicho de otro modo, la ingeniería de requisitos permite visualizar de un modo más claro lo que el cliente desea del software, permitiendo comprender las necesidades, analizar la factibilidad y negociar soluciones razonables. Para una mayor organización, los requisitos son divididos en dos grandes grupos: requisitos funcionales y requisitos no funcionales.

La obtención y análisis de los requisitos es un proceso de gran importancia para el correcto desarrollo de un sistema. Este proceso involucra el uso de diversos métodos que facilitan la comunicación con clientes, usuarios finales o potenciales y demás personas legales o jurídicas que facilitan la comprensión del Sistema (Sommerville, y otros, 2002). Uno de los métodos que se emplea para la obtención y análisis de los requisitos es la tormenta de ideas.

La entrevista es una conversación planificada entre el investigador y el entrevistado para obtener Información. Su uso constituye un medio para el conocimiento cualitativo de los fenómenos o sobre características personales del entrevistado. Suele ser empleada cuando el problema de estudio no se puede observar o es muy difícil hacerlo por ética o complejidad. Debido a que el desarrollador no guarda estrecha relación con el manejo de los datos de PRIMICIA, se decide entrevistar al líder de Proyecto, el cual contribuye significativamente al proceso de obtención y análisis de requisitos. Para complementar la información recopilada, se decide emplear la tormenta de ideas.

La tormenta de ideas consiste en realizar reuniones grupales encaminadas a generar ideas en un ambiente libre de críticas o juicios. En ella las propuestas de cada participante impulsan a los demás a intentar formular alguna otra. Esto facilita la obtención de mejores resultados y la toma de decisiones más acertadas, enriqueciendo la visión y brindando una perspectiva más amplia de la tarea a realizar. Estimula

tanto los aportes creativos como las discusiones sobre las fallas o los errores en que podría incurrirse. Para el uso de este método se selecciona un facilitador que posee dominio del tema de la investigación.

Como fruto de los métodos empleados, se obtuvo un conjunto de RF y RNF con los cuales debe cumplir el plugin de síntesis de voz de las noticias.

## 3.5. Requisitos Funcionales

Según Ian Sommerville, los requisitos funcionales (RF) "*son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar ante entradas particulares y de cómo se debe comportar en situaciones particulares*" (Sommerville, y otros, 2002). A continuación, se identifican los RF de la aplicación a desarrollar.

**RF 1. Activar la síntesis de voz.**

El jefe de canal debe tener la opción de activar o no la síntesis de voz generalizada en la plataforma PRIMICIA, independientemente si se publican noticias con síntesis de voz activada o desactivada. Así se logra un control general de la activación o desactivación de la síntesis de voz.

**RF 2. Permitir que el redactor decida qué noticia será sintetizada.**

El sistema debe permitir que el redactor de la noticia active la síntesis de voz para cada noticia que él redacte o edite.

**RF 3. Determinar el tiempo de duración de las noticias que tengan la síntesis de voz activada.**

El sistema debe guardar el tiempo de duración de cada noticia que tenga activada la síntesis de voz.

**RF 4. Realizar la síntesis de voz.**

El sistema debe ser capaz de realizar la síntesis de voz para cada noticia que tenga la opción activada.

**RF5: Establecer el lenguaje de la entonación para la síntesis de voz.**

El sistema debe permitir establecer el lenguaje de la entonación con la que se realiza la lectura de contenido.

**RF 6. Establecer la velocidad de la síntesis de voz.**

El sistema debe permitir establecer la velocidad de la síntesis de voz.

**RF 7. Establecer el tono de la síntesis de voz.**

El sistema debe permitir establecer el tono de la síntesis de voz.

**RF 8. Establecer volumen de la síntesis de voz.**

El sistema debe permitir establecer el volumen para realizar la síntesis de voz.

## 3.6. Requisitos No Funcionales

Ian Sommerville afirma que los requisitos no funcionales (RNF) "*como su nombre lo sugiere, son aquellos requerimientos que no se refieren a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento*" (Sommerville, y otros, 2002). En resumen, son un conjunto de reglas que hacen del sistema un producto agradable, usable, rápido o confiable. Estos requisitos se encargan de especificar las propiedades que debe tener el producto, así como las restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad, fiabilidad, entre otras. Seguidamente se definen los RNF pertenecientes a la aplicación a desarrollar:

**RNF 1. Fiabilidad**:

Los usuarios tendrán acceso a la información manejada por el sistema en dependencia del rol que se les asigne en el proyecto PRIMICIA. El sistema debe estar disponible todo el tiempo para sus usuarios, descontando el tiempo que se encuentre en mantenimiento.

**RNF 2. Eficiencia:**

El tiempo de respuesta estará dado por la cantidad de información a procesar, entre mayor cantidad de información mayor será el tiempo de procesamiento. De igual modo, la velocidad de procesamiento de la información, la actualización y la recuperación dependerán de la cantidad de información que tenga que procesar la aplicación.

**RNF 3. Requisitos de hardware.**

Los requisitos de hardware fueron definidos por el arquitecto del proyecto PRIMICIA y como el procesamiento más complejo de la solución propuesta es el que realiza la biblioteca y esta tiene menores requerimientos que los definidos (Developer Section on Free Accessibility Solutions, 2016), por tanto se mantienen los siguientes:

Para la computadora cliente se necesita: 1GB de RAM, microprocesador Core 2 Duo a 3.3 GHz, espacio en disco: 40 GB.

Para el Servidor Administración se necesita: microprocesador Core 2 Duo a 3.3 GHz, 4 GB de RAM, espacio en disco: 500 GB.

Para el Servidor de Transmisión: Microprocesador: Core 2 Duo a 3.3 GHz.

**RNF 4. Requisitos de software:**

Para la computadora cliente es necesario un navegador Mozilla Firefox 35 o Chrome 32.-Sistema operativo: Ubuntu 14.04 o Windows 7 o superior.

Para el Servidor de Administración se necesita sistema operativo: Ubuntu 14.04, Sistema Gestor de Base de Datos: PostgreSQL 9.2, un servidor Web: Apache 2.2.

Para el Servidor de Transmisión se necesita sistema operativo: Ubuntu 14.04, Qt 5.

**RNF 5. Estándares aplicables:**

El sistema será desarrollado bajo los patrones de las normativas ISO.

## 3.7. Descripción de los actores del sistema

A continuación, se especifican los actores con los que cuenta el sistema:

Tabla 3. Descripción de los actores del sistema

| Actor | Descripción |
|---|---|
| **Jefe de canal** | Usuario que a su vez es un redactor y que podrá realizar las funcionalidades de administración (activar o desactivar la síntesis de voz general). Además de tener acceso a la activación o desactivación de la síntesis de voz de cada noticia. |
| **Corrector** | Usuario que a su vez es un redactor y podrá realizar las funcionalidades de activar o desactivar la síntesis de voz en cada noticia que gestione |
| **Editor** | Usuario que a su vez es un redactor y que podrá realizar las funcionalidades de |

| | |
|---|---|
| | activar o desactivar la síntesis de voz en cada noticia que gestione. |
| **Redactor** | Usuario que podrá realizar las funcionalidades de activar o desactivar la síntesis de voz en cada noticia que gestione |

## 3.8. Descripción de los Casos de Uso del Sistema

Los casos de uso son la entrada esencial para realizar el análisis, diseño, implementación y pruebas del sistema. Es por ello que realizar la descripción de los mismos es de gran importancia. El jefe de canal será el único que tendrá acceso a la activación o desactivación del *plugin* de síntesis de voz, en dependencia de las necesidades del cliente o de las necesidades específicas en el momento de la transmisión del canal de noticias PRIMICIA.

## 3.9. Diagrama de Caso de Uso del Sistema

Un diagrama de casos de uso es una representación gráfica de un conjunto de casos de uso del sistema, actores y su interacción. En él, cada caso de uso debe comunicarse con al menos un actor. Estos diagramas son de gran importancia ya que permiten modelar el comportamiento de un sistema, un subsistema o una clase. A continuación, se muestra el diagrama de casos de uso del sistema correspondiente a esta investigación.

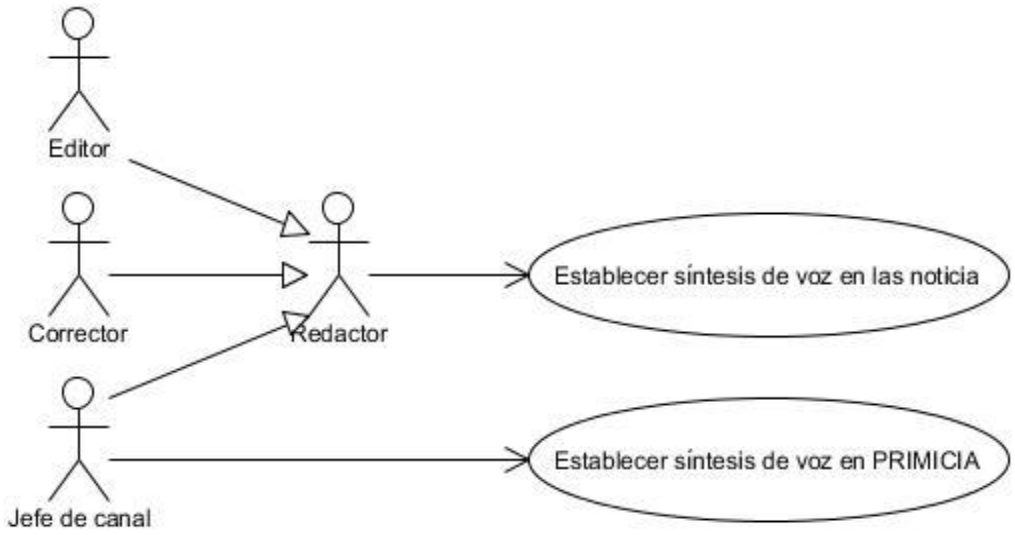



Figura 5. Diagrama de casos de uso del Sistema para el subsistema de Administración.

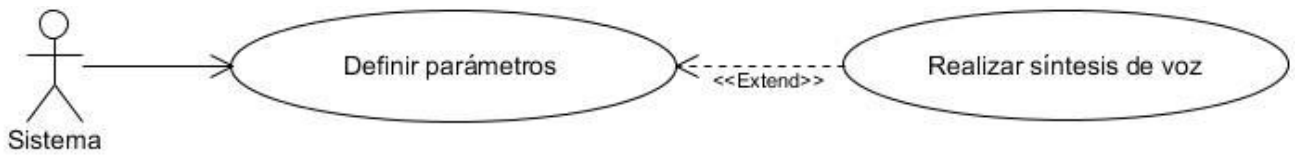



Figura 6. Diagrama de casos de uso del Sistema para el subsistema de Transmisión.

Tabla 4. Descripción del caso de uso Realizar síntesis de voz

| | |
|---|---|
| **Caso de Uso:** | Realizar síntesis de voz. |
| **Actores:** | Sistema. |
| **Propósito:** | Este caso de uso se lleva a cabo con el objetivo de realizar la síntesis de voz. |
| **Resumen:** | El caso de uso inicia cuando el sistema indique que se va a realizar la síntesis de voz o determinar la duración de la misma y termina con la realización de la síntesis de voz o con la actualización del valor de la duración de la noticia en pantalla para cuando sea sintetizada. |
| **Precondiciones:** | La síntesis de voz en PRIMICIA debe estar activada.<br>Debe existir al menos una noticia con el sintetizador activado.<br>Se deben haber definido los parámetros de configuración. |
| **Referencias:** | RF 3, RF 4. |
| **Prioridad:** | Crítico |
| **Flujo Normal de Eventos** | |
| **Realizar síntesis de voz** | |
| **Acción del Actor** | **Respuesta del Sistema** |

| | |
|---|---|
| 1. Decide realizar una de las siguientes opciones:<br>- Determinar duración de la síntesis de voz (sección 1).<br>- Realizar síntesis de voz (sección 2). | |
| 2. Envía el identificador de la noticia a la que se le realiza la acción. | |
| | 3. Busca en la base de datos la noticia. |
| | 4. Decodifica el json de la noticia, que se encuentra encriptado en base 64. |
| | 5. Obtiene el html de la noticia. |
| | 6. Elimina las etiquetas del html. |
| | 7. Obtiene un texto claro y legible. |
| | 8. Crea un objeto por cada pantalla de tipo texto o texto imagen que es relacionado con la noticia. |

| **Sección 1 “Determinar duración de la síntesis de voz”** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| | 9. Determina la duración de la síntesis de voz en la noticia. |
| | 10. Actualiza el json de la noticia con el tiempo de duración de la lectura en cada pantalla y la duración total de cada noticia. |
| | 11. Termina el caso de uso. |

| **Sección 2 “Realizar síntesis de voz”** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| | 7. Realiza la síntesis de voz con los parámetros de configuración definidos anteriormente. |
| | 8. Termina el caso de uso. |

Tabla 5. Descripción del caso de uso Establecer síntesis de voz en PRIMICIA

| **Caso de Uso:** | Establecer síntesis de voz en PRIMICIA. |
|---|---|
| **Actores:** | Jefe de canal. |
| **Propósito:** | Este caso de uso se lleva a cabo con el objetivo de establecer la síntesis de voz en PRIMICIA. |
| **Resumen:** | El caso de uso inicia cuando el jefe de canal decide activar la síntesis de voz en PRIMICIA y termina cuando el sistema registra la petición del usuario. |
| **Precondiciones:** | Los actores tienen que estar autenticados en la aplicación con permiso de jefe de canal. |
| **Referencias:** | RF 1. |
| **Prioridad:** | Medio |
| **Flujo Normal de Eventos** | |
| **Establecer síntesis de voz en la noticia.** | |
| **Acción del Actor** | **Respuesta del Sistema** |

| | |
|---|---|
| 9. Presiona el botón para activar la síntesis de voz en PRIMICIA. | |
| | 10. Verifica que la síntesis de voz está desactivada. |
| | 11. Establece en verdadero el atributo voz en off de PRIMICIA. |
| | 12. Termina el caso de uso. |
| **Flujos alternos** | |
| **2a. La voz en off ya está activada.** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | 1. Establece en falso el atributo voz en off de PRIMICIA. |
| | 2. Termina el caso de uso. |

Tabla 6. Descripción del caso de uso Establecer síntesis de voz en la noticia

| | |
|---|---|
| **Caso de Uso:** | Establecer síntesis de voz en la noticia. |
| **Actores:** | Jefe de canal, Redactor, Corrector, Editor. |
| **Propósito:** | Este caso de uso se lleva a cabo con el objetivo de establecer la síntesis de voz en la noticia. |
| **Resumen:** | El caso de uso inicia cuando el redactor decide activar la síntesis de voz en la noticia y termina cuando el sistema registra la petición del usuario. |
| **Precondiciones:** | Los actores tienen que estar autenticados en la aplicación. |

| | |
|---|---|
| **Referencias:** | RF 2. |
| **Prioridad:** | Medio |
| **Flujo Normal de Eventos** | |
| **Establecer síntesis de voz en la noticia.** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 13. Presiona el botón para activar la síntesis de voz en la noticia. | |
| | 14. Verifica que la síntesis de voz está desactivada. |
| | 15. Establece en verdadero el atributo voz en off de la noticia. |
| | 16. Termina el caso de uso. |
| **Flujos alternos** | |
| **2a. La voz en off ya está activada.** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | 3. Establece en falso el atributo voz en off de la noticia. |
| | 4. Termina el caso de uso. |

Tabla 7. Descripción del caso de uso Establecer parámetros.

| | |
|---|---|
| **Caso de Uso:** | Establecer parámetros. |
| **Actores:** | Sistema. |

| | |
|---|---|
| **Propósito:** | Este caso de uso se lleva a cabo con el objetivo de definir los parámetros de configuración con los que se va a realizar la síntesis de voz. |
| **Resumen:** | El caso de uso inicia cuando el actor desea establecer los parámetros de configuración para la síntesis de voz y termina con el establecimiento de los parámetros. |
| **Precondiciones:** | |
| **Referencias:** | RF 5, RF 6, RF 7, RF 8. |
| **Prioridad:** | Medio |
| **Flujo Normal de Eventos** | |
| **Establecer parámetros.** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Selecciona el motor de habla. | |
| | 2. Muestra los parámetros asociados al motor seleccionado (lenguaje de la entonación, velocidad de lectura, tono y volumen). |
| 3. Establece los parámetros asociados al motor (lenguaje de la entonación, velocidad de lectura, tono y volumen). | |
| | 4. Muestra las voces asociadas al lenguaje de entonación. |
| 5. Establece la voz. | |
| | 6. Quedan establecidos todos los parámetros para la |

| | |
|---|---|
| | síntesis de voz. |
| | 7. Termina el caso de uso. |

## 3.10. Estilo arquitectónico

*"Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una estructura para todos los componentes del sistema"* ( Pressman, 2010). Los estilos definen los patrones que posiblemente se utilicen en los sistemas, evaluando así la arquitectura que más se ajuste a los requisitos que tiene definido el software. Los estilos pueden apreciarse como una forma de organizar los distintos modelos de sistemas.

Para el desarrollo de la solución propuesta se utiliza el estilo arquitectónico Llamada y retorno. Este estilo tiene la ventaja de brindar al programa una estructura relativamente fácil de modificar lo que contribuye a la flexibilidad del *plugin* a desarrollar. En la solución se emplea este estilo ya que la misma cuenta con una aplicación principal que se comunica con el resto de las partes de la solución a través de llamadas y esperando un retorno, brindando un mejor desempeño interno de comunicación en el sistema.

### 3.10.1. Patrón arquitectónico

Los patrones arquitectónicos o patrones de arquitectura, ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor ( Pressman, 2010).

Según ( Pressman, 2010) un patrón arquitectónico en capas permite organizar el sistema en diferentes conjuntos de abstracción, lo cual garantiza entre varios aspectos proporcionar amplia reutilización y admitir refinamientos y optimizaciones.

Para la realización del *plugin* de síntesis de voz se utiliza el patrón en capas, específicamente en dos capas: Lógica y Acceso a Datos. La capa lógica se encarga del procesamiento que tiene lugar en la aplicación y la de acceso a datos es la que se encarga de garantizar la persistencia de los datos. Se escoge el patrón en capas debido a que distribuye el trabajo por capas donde cada una provee de

servicios a la otra. La arquitectura en dos capas permite hacer más sencillo el desarrollo de las aplicaciones para luego brindarle un mejor mantenimiento.

Para que los actores del sistema puedan activar el *plugin*, es necesario agregarle al subsistema de administración una opción para el trabajo con el sintetizador. Debido que la plataforma PRIMICIA se desarrolla en Symfony, framework que está basado en el patrón arquitectónico Modelo Vista Controlador (MVC), se decide utilizarlo para poder generar los mismos artefactos. MVC es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Esta separación permite hacer modificaciones en un componente sin necesidad de modificar los otros. De manera genérica los componentes MVC se podrían definir como sigue ( Pressman, 2010):

**Modelo:** es la representación específica de la información con la que el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas.

**Vista:** presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

**Controlador:** responde a eventos, usualmente acciones del usuario e invoca peticiones al modelo y probablemente a la vista. Por tanto se podría decir que el controlador hace de intermediario entre la vista y el modelo.

### 3.10.2. Patrones de diseño

Los patrones de diseño mejoran la calidad de los sistemas además de brindar una solución probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Los patrones de diseño se dividen en dos grupos, los Patrones Generales de Software para Asignar Responsabilidades y los Patrones del Grupo de los Cuatro. A continuación se presentan los patrones de diseño que son utilizados para el desarrollo de la solución.

#### Patrones GRASP

Los patrones generales de software para asignar responsabilidades (por sus siglas en inglés GRASP) permiten asignar responsabilidades a objetos de forma tal que conozcan el rol que le corresponde. Son descritos los principios fundamentales expresados en forma de patrones de la asignación de responsabilidades (Larman, 2003). Los patrones que se utilizan son:

**Experto:** Asigna una responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Constituye un principio básico que frecuentemente se utiliza en el diseño orientado a objetos. Las ventajas que presenta este patrón son: la conservación del encapsulamiento debido a que los objetos para hacer lo que se les pide se valen de su propia información, además de posibilitar que el sistema sea robusto y de fácil mantenimiento. En la solución el patrón se evidencia en la clase Text de la capa Lógica.

**Creador:** Asigna la responsabilidad a una clase de crear objetos, práctica muy frecuente en los sistemas orientados a objetos. Entre sus beneficios se encuentra el de brindar soporte a un bajo acoplamiento lo que trae como consecuencia una menor dependencia respecto al mantenimiento y la reutilización en mayor medida. En la solución el patrón se evidencia en la clase New de la capa Lógica.

**Controlador:** Asigna la responsabilidad a una clase de administrar un mensaje de los eventos del sistema. *"Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del sistema"* (Larman, 2003). En la solución el patrón se evidencia en la case TextToSpeech de la capa Lógica.

**Alta cohesión:** Asigna las responsabilidades en función de mantener la complejidad dentro de los límites posibles. La cohesión es una medida que refleja cuán enfocadas y relacionadas se encuentran las responsabilidades en una clase. Las clases que presentan una alta cohesión se caracterizan por poseer responsabilidades estrechamente relacionadas que no hagan un enorme trabajo; sin embrago, una clase con baja cohesión realiza muchas cosas no afines a un trabajo excesivo. Entre los beneficios que presenta el patrón está la mejora de la facilidad y claridad para comprender el diseño. En la solución el patrón se evidencia en la clase DataAcces de la capa de Acceso a datos.

**Bajo acoplamiento:** Asigna la responsabilidad a las clases garantizando que las mismas se encuentren lo más independiente posibles, lo que trae consigo una dependencia escasa y un aumento en la reutilización. En la solución el patrón se refleja en la clase Connection de la capa de Acceso a datos.

**Patrones GOF**

Los patrones GOF son soluciones técnicas basadas en Programación Orientada a Objetos (POO). Están clasificados en patrones creacionales, estructurales y de comportamiento. Los creacionales son los encargados de la creación de instancias de los objetos, los estructurales plantean las relaciones entre clases y los de comportamiento plantean la interacción y cooperación entre las clases (Larman, 2003).

**Observador (Observer):** este patrón define una dependencia del tipo uno a muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar el cambio a todos los otros dependientes (Larman, 2003). Se utiliza para cuando el jefe de canal active la síntesis de voz inmediatamente la transmisión comience a realizar la síntesis. En la solución el patrón se evidencia en la clase TextToSpeech de la capa Lógica.

**Instancia única o Singleton:** es un patrón de creación y garantiza que una clase posea una única instancia, a la vez que provee un punto de acceso global a ella. Permite por su diseño restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. En la solución el patrón se evidencia en la clase Connection de la capa de Acceso a datos.

## 3.11. Diagrama de clases del diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, el modelo de diseño sirve de abstracción de la implementación del sistema y es de ese modo utilizada como una entrada fundamental de las actividades de implementación (Booch, y otros, 2000).

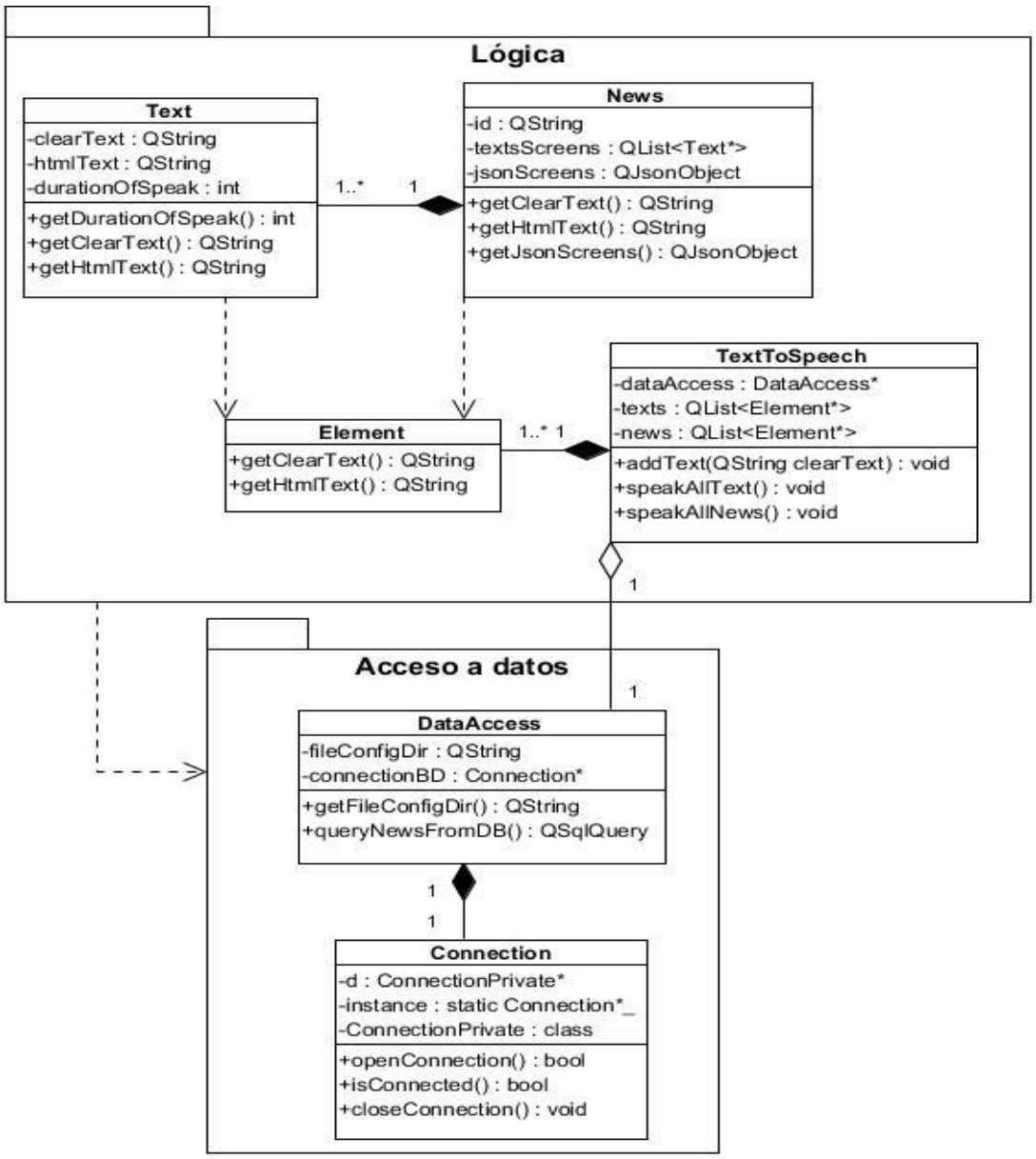



Figura 7. Diagrama de Clases del diseño

## Descripción del diagrama de clases del diseño

La capa Lógica agrupa las siguientes clases:

**TextToSpeech:** Tiene la responsabilidad de controlar toda la lógica a través de sus funcionalidades: por ejemplo agregar noticias o textos, realizar las síntesis de voz, determinar el tiempo de lectura de un texto o de una noticia y establecer los parámetros de lectura.

**Element:** Es una abstracción que agrupa funcionalidades comunes entre las clases Text y New para facilitar su acceso desde la clase TextToSpeech.

**News:** Agrupa todas las funcionalidades y atributos asociados al concepto noticia de PRIMICIA; por ejemplo id de la noticia, título, json que contiene las pantallas de la noticia y una lista de objetos de la clase Text para los textos de las pantallas.

**Text:** Entidad que contiene un texto en formato html o de forma clara y la funcionalidad de hacer el cambio entre texto claro y texto en formato html, además de la duración de la lectura del texto claro.

La capa de Acceso a datos agrupa las siguientes clases:

**DataAccess:** Tiene como objetivo principal el acceso y persistencia de los datos ya sea en ficheros como en la base de datos.

**Connection:** Se encarga de establecer la conexión con la base de datos y realizar las consultas que se determinen en la clase DataAccess.

### 3.12. Resultados de la investigación

Como resultado de la investigación se obtuvo un *plugin* para la síntesis de voz en las noticias de la plataforma PRIMICIA, el cual a su vez es una biblioteca compartida que utiliza el módulo QtTextToSpeech. Por ser una biblioteca compartida el *plugin* puede utilizarse en cualquier aplicación que sea implementada con el lenguaje C++. La biblioteca cuenta con varias funcionalidades como: establecer la velocidad de lectura, ajustar el tono, escoger el idioma de pronunciación y escoger las voces. Permite además determinar el tiempo de duración de la lectura de un texto y de una noticia, ayudando así a establecer el tiempo de duración de cada pantalla. Al obtener el tiempo de duración el subsistema de transmisión puede ajustar la duración de la pantalla con el tiempo que se demora el proceso de lectura.

El módulo QtTextToSpeech tiene implementado varios motores de habla en dependencia del sistema operativo, lo cual dota a la solución de la característica multiplataforma. Los motores de habla que implementa la solución son: osx para mac, qttospeech.java para android, sapi y winrt para windows, flite y speechdispatcher para GNU/ linux. Debido a que el subsistema de transmisión se desarrolla en GNU/Linux se seleccionó el speechdispatcher, el cual presenta las mejores voces y la mayor cantidad de lenguajes entre las distribuciones de GNU/Linux.

## 3.13. Conclusiones parciales del capítulo

- La modelación del dominio y del sistema, a partir de sus correspondientes diagramas proporciona una visión común del negocio y una guía certera en función de obtener un producto de interés para el cliente.

- La obtención de requisitos y la descripción textual de los casos de uso permitió comprender las funcionalidades que intervienen en la solución.

-Los estilos y patrones utilizados proporcionan uniformidad, comprensión y escalabilidad a la solución desarrollada, al mismo tiempo que favorecen la rigurosidad en su diseño.

# Capítulo 4

# Validación de requisitos

## Introducción

Una vez concluido el modelo del diseño, se dispone de las condiciones necesarias para proceder a la construcción del sistema. En el presente capítulo se abordarán aspectos referentes a las disciplinas de implementación y prueba definidas por la metodología de software. Se realiza el diagrama de componentes el cual permite tener una visión más amplia de la organización y dependencias lógicas de los componentes del sistema. También se modela el diagrama de despliegue que representa la distribución física de las funcionalidades en sus respectivos nodos.

## 4.1. Diagramas de Componentes

*"Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño"* (Jacobson, y otros, 2000). Entre los estereotipos estándar de componentes se encuentran: *executable* (programa que puede ser ejecutado en un nodo), *file* (fichero que contiene datos o código fuente), *library* (librería estática o dinámica) y *document* (documento). Los estereotipos pueden ser modificados durante la creación de componentes en un entorno de implementación particular. Un diagrama de componentes describe la estructura física del sistema y posibilita la representación de las dependencias de los componentes tanto en tiempo de ejecución como de compilación.

A continuación se muestra el diagrama de componentes donde se aprecia la división del sistema en componentes y las dependencias que se generan entre los mismos.

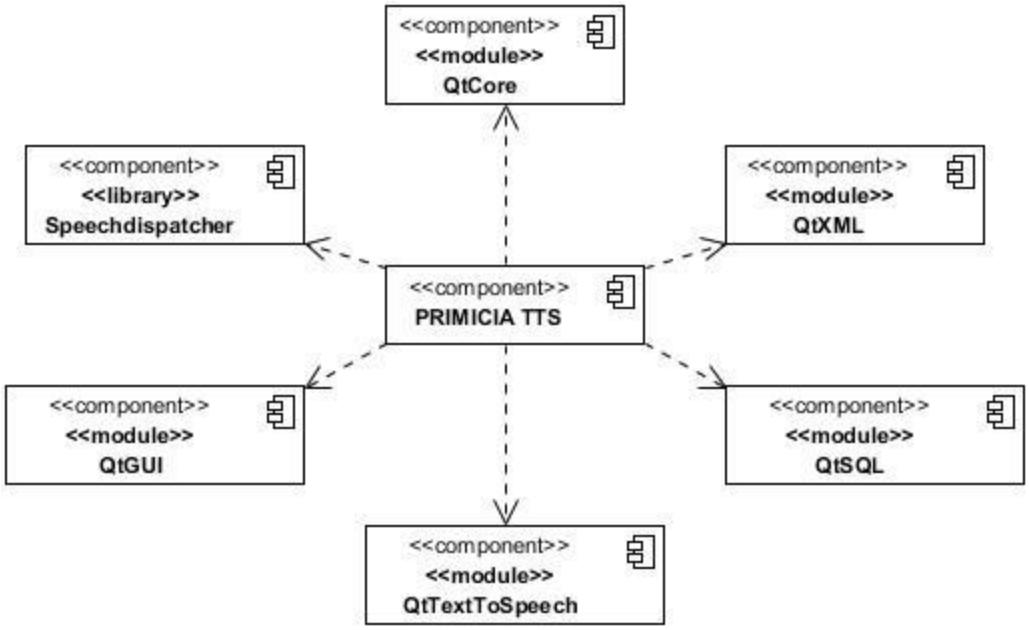



Figura 8. Diagrama de Componentes

El diagrama de componentes que se muestra contiene los siguientes módulos: QtCore que se utiliza para la mayoría de los objetos como lo son las clases QObject y QString. Otro módulo es QtXML, utilizado para el trabajo con ficheros xml, un ejemplo es la clase QDomDocument. El módulo QtSQL es el que realiza el trabajo con la base de datos, un ejemplo es la clase QSqlQuery y el QSqlDatabase. El módulo QtTextToSpeech es el intermediario entre la biblioteca desarrollada y el motor de habla QTextToSpeech. El módulo QtGUI es utilizado para extraer un texto claro de un texto en formato HTML, un ejemplo de ello es la clase QTextDocument.

## 4.2. Estándar de codificación

Los estándares de codificación son reglas y descripciones que son establecidas para facilitar la comprensión y el mantenimiento del código. El estándar de codificación debe seguirse desde el comienzo del *software*. El empleo de los estándares permite obtener un código legible, reduce la posibilidad de cometer errores, mejora la comunicación entre los programadores y brinda una guía bien documentada para el mantenimiento del *software*. En la presente investigación se propone el uso del siguiente estándar de codificación para cumplir con las políticas establecidas por el departamento de Señales Digitales.

**Apariencia de atributos:**

- Se escriben en minúscula y su nombre debe tener relación con el valor que almacenan.

**Apariencia de funciones:**

-Se escriben en mayúscula y su nombre debe tener relación con la operación que realizan.

-En caso de que el nombre sea compuesto se utiliza la notación *CamelCase*.

**Apariencia de clases:**

-El nombre debe indicar con un sufijo la capa arquitectónica a la que pertenece y tener relación con el objetivo de la clase.

**Separadores:**

-Se utiliza el separador "_" para nombrar las clases.

**Líneas y espacios en blanco:**

-No contener más de una instrucción por línea.

-Dejar una línea en blanco entre las declaraciones de cada función.

-Las declaraciones de datos dentro de una función deben ir al inicio y separadas de las instrucciones ejecutables de la función a través de una línea en blanco.

### 4.3. Modelo de despliegue

El diagrama de despliegue se utiliza para modelar las relaciones entre los componentes del sistema implementado y su distribución en el o los hardware utilizados. Estos diagramas permiten describir la distribución física del sistema, así como la distribución de las funcionalidades de la solución en cada nodo. Cada nodo representa un recurso de cómputo (exceptuando el nodo Televisor) y poseen relaciones que representan medios de comunicación entre ellos.

A continuación se representa el modelo de despliegue correspondiente a la síntesis de voz en la Plataforma de Televisión Informativa PRIMICIA.

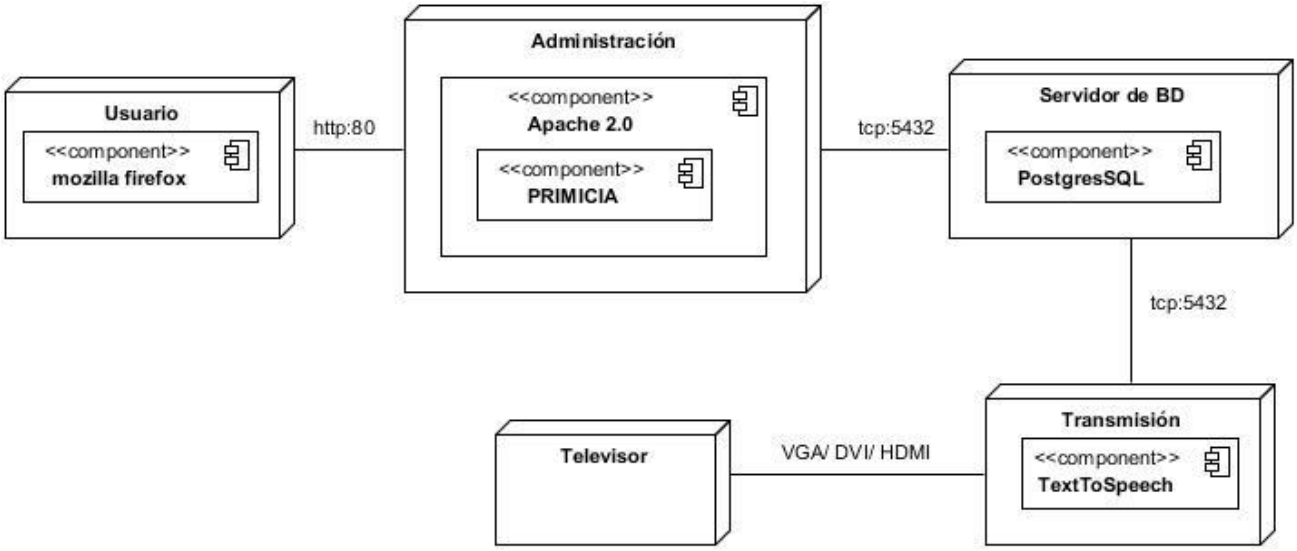



Figura 9. Diagrama de despliegue

Este modelo está conformado por las PC donde los usuarios de la plataforma van a acceder al subsistema de administración para gestionar las noticias que posteriormente serán vistas por los televidentes. Estas PC se comunican con el servidor web mediante el protocolo HTTP (Hypertext Transfer Protocol) por el puerto 80. El servidor web, a su vez, se comunica con el servidor de base de datos a través del protocolo TCP/IP (Transmission Control Protocol/Internet Protocol) por el puerto 5432. Este último es el encargado de manejar toda la información referente a las noticias. El servidor de base de datos también se comunica con el servidor de transmisión, quien contiene el *plugin* para la síntesis de voz. El servidor de transmisión transfiere la programación que se gestiona hacia los televisores, por donde los televidentes pueden observar las noticias publicadas.

### 4.4. Pruebas al sistema

El proceso de pruebas no es más que el proceso de ejecución de un programa con la intención de descubrir un error, una prueba tiene éxito si descubre un error no detectado hasta entonces. Las pruebas demuestran hasta qué punto las funciones del software parecen funcionar de acuerdo con las especificaciones. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad ( Pressman, 2010).

La estrategia que se ha de seguir a la hora de evaluar dinámicamente un sistema software debe permitir comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar el software en su conjunto. Las pruebas unitarias están dirigidas a probar cada componente

individualmente para asegurar que funcione de manera apropiada como unidad ( Pressman, 2010). Estas pruebas son aplicables a los componentes representados en el modelo de implementación para verificar que los flujos de control y de datos estén cubiertos, además de que funcionen como se espera.

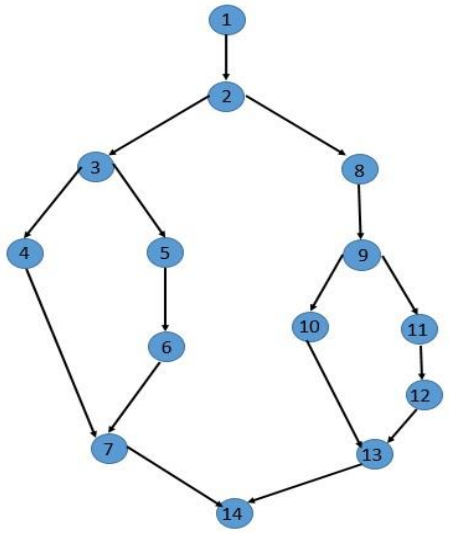
### 4.4.1. Pruebas de caja blanca

Las pruebas de caja blanca se basan en un minucioso examen de los detalles procedimentales. Mediante las pruebas de caja blanca se obtienen casos de pruebas que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo, ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas, ejecuten todos los bucles en sus límites y con sus límites operacionales y ejerciten las estructuras internas de datos para asegurar su validez ( Pressman, 2010). Para el *plugin* se realizó la técnica del camino básico escogiendo un caso de prueba.

```
164 void TextToSpeech::speakAll(QString type)
165 {
166     speech->stop();
167     posOfSay=0;
168     if(type=="text")
169     {
170         if(texts.size()>posOfSay)
171         {
172              speech->say(texts.at(posOfSay)->getClearText());
173         }
174         else
175         {
176             posOfSay=-1;
177         }
178     }
179     else
180     {
181         if(news.size()>posOfSay)
182         {
183             qobject_cast<News*>(news.at(posOfSay))->setPosOfRead(0);
184             speech->say( qobject_cast<News*>(news.at(posOfSay))->getClearTextToSpeak());
185         }
186         else
187         {
188             posOfSay=-1;
189         }
190     }
191 }
```

Figura 10. Ejemplo de código del plugin

**Paso número 1:**




**Paso número 2:**

Complejidad ciclomática = Aristas – Nodos + 2 = 16-14+2 = 4

**Paso número 3:**

Cantidad de caminos independientes:

1-2-3-4-7-14

1-2-3-5-6-7-14

1-2-8-9-10-13-14

1-2-8-9-11-12-13-14

**Paso número 4:**

Caso de prueba para: 1-2-3-4-7-14

Entrada: texto (tipo).

Resultado esperado: El sintetizador lee todos los textos.

Resultado de la prueba: durante la primera iteración los resultados obtenidos fueron satisfactorios.

### 4.4.2. Pruebas de integración

Las pruebas de integración implican una progresión ordenada de pruebas que van desde los componentes y culminan en el sistema completo. Se clasifican en integración incremental (ascendente o descendente) e integración no incremental. La primera se combina el módulo que se debe probar con el conjunto de módulos que ya han sido probados, mientras que la segunda se prueba cada módulo por separado y luego se integran todos a la vez probando al sistema completo (Fernández Peña, 2011).

Para probar la correcta integración del *plugin* con la plataforma PRIMICIA se aplicaron las pruebas de integración incremental ascendente, pues a medida en que se implementaba o modificaba algún componente se iba probando. Para la realización de estas pruebas se definieron dos niveles, los cuales se explican a continuación:

Primer nivel: Se comprueba el *plugin* donde se realiza la síntesis de voz. Cuando se terminó de implementar el *plugin* para la síntesis de voz se comprobó su funcionamiento, determinando si cumplía su funcionalidad correctamente. Se detectaron algunos errores los que fueron corregidos de forma satisfactoria, por ejemplo que no cargaba correctamente el json de las noticias.

Segundo nivel: Luego de la implementación del *plugin* se comprobó su integración con la plataforma PRIMICIA. Por último se verificó que cuando el usuario activara la síntesis de voz en el subsistema de administración el *plugin* se activara y comenzara a realizar la lectura de las noticias. En este proceso se detectaron algunos errores, por ejemplo cuando la noticia tenía pantallas de tipo texto imagen no se leía la pantalla. Otro error que se obtuvo es que cuando el sintetizador encontraba una noticia que solo tenía video o imagen daba error. Todos los errores encontrados fueron solucionados satisfactoriamente.

En la última etapa de pruebas se activó el sintetizador en 5 noticias de las 10 creadas para probar que el *plugin* solo sintetizara las noticias que el usuario seleccionó. En este proceso se obtuvo respuestas satisfactorias demostrando que todas las funcionalidades del sistema se realizan correctamente.

## 4.5. Técnicas para evaluar los sintetizadores de voz

Hasta la fecha se han realizado pocos estudios para tratar de evaluar los sistemas de síntesis de voz de forma cuantitativa, por lo que la evaluación ha de ser principalmente subjetiva. Para ello se han ideado varios métodos.

Uno de los procedimientos más utilizados es el MOS (Mean Opinion Score). Se presenta la voz sintética a distintos oyentes, los cuales en función de su criterio evalúan la calidad dando una puntuación entre 1 y 5. Este método permite evaluar la inteligibilidad, siendo muy utilizado para evaluar voz codificada (ScanSoft, 2012). Para la realización de esta prueba se escogieron 5 personas con el objetivo de evaluar solamente la inteligibilidad del sintetizador de voz. Se sintetizaron 6 noticias como pruebas. El siguiente gráfico muestra los resultados obtenidos.

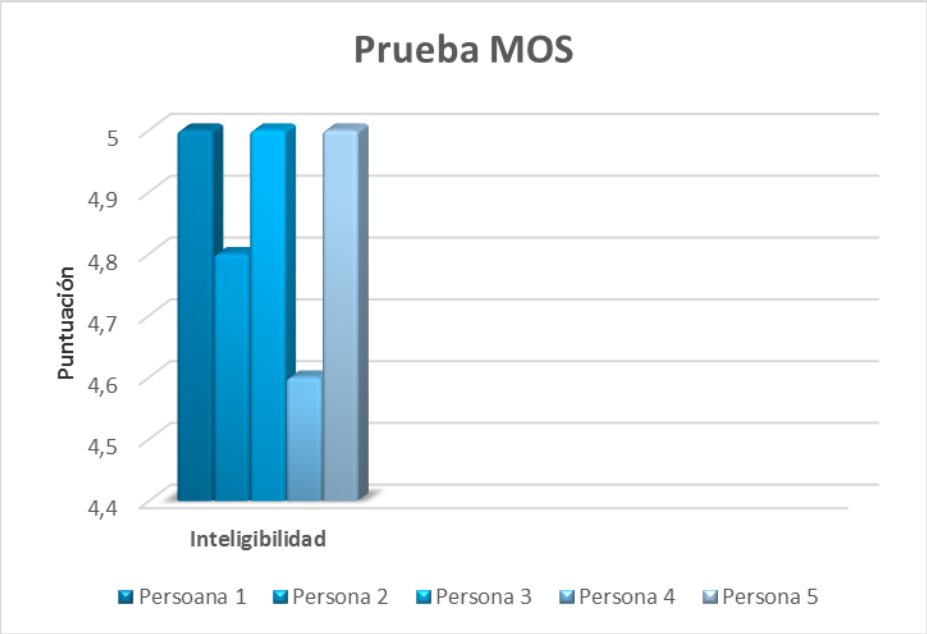



Figura 11. Gráfica de la prueba MOS

El método MRT (Modified Rhyme Test) trata de evaluar únicamente la inteligibilidad de la voz sintética eliminando la gran redundancia existente en los lenguajes naturales. Se presentan a los oyentes 6 construcciones del tipo consonante-vocal-consonante, de las cuales deben elegir una. Este método tiene el inconveniente de que no sirve para evaluar un sintetizador de forma global, ya que representa situaciones que no se corresponden con la realidad (ScanSoft, 2012). Para la realización de esta prueba se seleccionaron 5 personas para poder percibir si podían entender las construcciones. El siguiente gráfico muestran los resultados obtenidos.

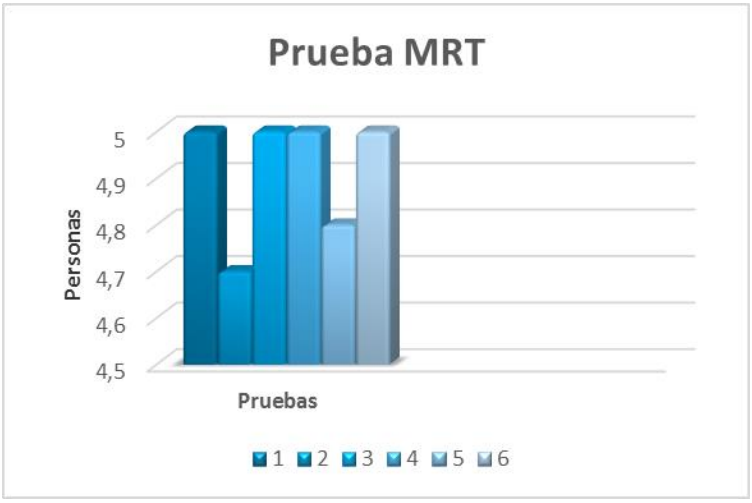



Figura 12. Gráfica de la prueba MRT

Otros de los métodos empleados para medir la inteligibilidad es el método SUS (Semantically Unpredictable Sentences). En este caso se emplean frases completas y sintácticamente correctas pero carentes de significado. El oyente se encuentra en un entorno de utilización real de un sintetizador de voz y además no puede apoyarse en la redundancia del lenguaje natural para reconocer las frases (ScanSoft, 2012). Para la realización de esta prueba se escogieron 5 personas para percibir si podían entender las 6 frases escogidas como pruebas. El siguiente gráfico muestra los resultados obtenidos.

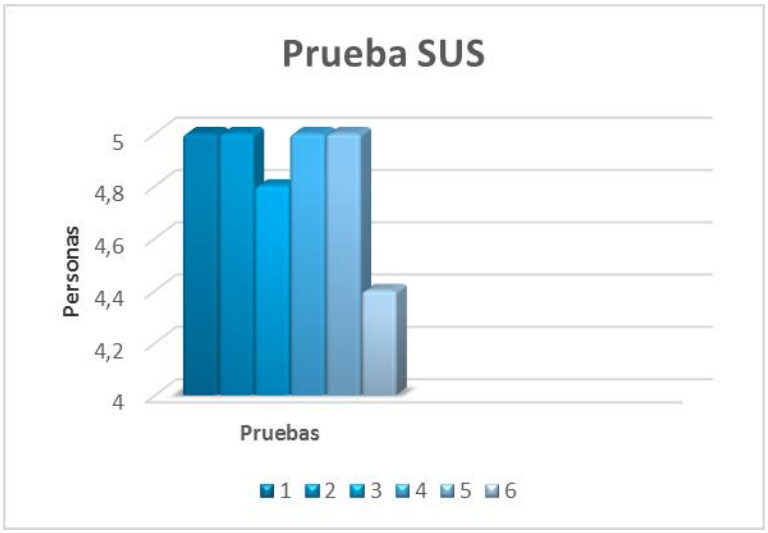



Figura 13. Gráfica de la prueba SUS

### Resultados obtenidos de las pruebas al sintetizador

La prueba MOS arrojó como resultado una inteligibilidad de un 97.6%. Por otra parte la prueba MRT dio como resultado un 98,3% y la prueba SUS un 97.3%. Teniendo en cuenta los resultados obtenidos se detectó que la inteligibilidad es alta en el sintetizador desarrollado.

## 4.5. Conclusiones parciales

-El estándar de codificación definido siguiendo las políticas del departamento Señales Digitales, permite tener un mejor entendimiento y mantenimiento de la solución, posibilitando que otros desarrolladores comprendan el código para una futura utilización o modificación.

-Con la realización de pruebas al sistema queda comprobado que la solución propuesta se ejecuta correctamente y sin errores.

-El uso de técnicas para evaluar sintetizadores de voz como MOS, MRT y SUS permitió comprobar que el *plugin* para la síntesis de voz desarrollado posee una alta inteligibilidad.

## Conclusiones generales

Al concluir la presente investigación se obtienen resultados que dan cumplimiento satisfactorio al objetivo general planteado inicialmente.

- La caracterización del proceso de desarrollo de los sintetizadores de voz permitió que se tuvieran los elementos fundamentales a tener en cuenta para la construcción de un sistema de este tipo. Por su parte, contribuyó a identificar las funcionalidades comunes que deben tener los sintetizadores de voz.
- Con la realización del *plugin* para la síntesis de voz se logra proveer a la Plataforma de Televisión Informativa PRIMICIA de una herramienta capaz de generar la lectura de noticias.
- La herramienta, metodología y lenguaje de modelado identificado a partir de la investigación de las herramientas y tecnologías actuales para el desarrollo de software garantizó la realización del correcto modelado de un conjunto de artefactos necesarios para el posterior diseño e implementación de la aplicación.
- Las pruebas realizadas demuestran el correcto funcionamiento de la solución.

.

## Recomendaciones

Las recomendaciones del presente trabajo de diploma están encaminadas a:

- Compilar los *plugin* de motores de habla que implementa la solución como: osx para mac, qttospeech.java para android, sapi y winrt para Windows.

## Bibliografía


**Pressman, Roger S. 2010.** *Ingeniería de Software. Un enfoque práctico.* 2010.

**The jQuery Foundation. 2016.** License jQuery Foundation. [En línea] The jQuery Foundation, 2016. [Citado el: 18 de 2 de 2016.] https://jquery.org/license/.

**Allen , Jonathan, Hunnicutt, M. Sharon y Klatt, Dennis. 1987.** *From Text to Speech: The MITalk system.* s.l. : Cambridge University Press, 1987. 0-521-30641-8.

**Amazon Company. 2016.** IVONA Text to Speech. [En línea] Amazon Company, 2016. [Citado el: 18 de 2 de 2016.] https://www.ivona.com/.

**Birkholz, Peter y Kroger , Bernard . 2013.** *Simulation of vocal tract growth for.* Saarbrücken : Institute for Computer Science, 2013. pp. 377–380.

**Black, Alan W. 2002.** *Perfect synthesis for all of the people all of the time.* s.l. : IEEE TTS Workshop, 2002.

**Booch, Grady, Jacobson, Ivar y Rumbaugh, James. 2000.** *El Proceso Unificado de Desarrollo de Software.* 2000.

**Bootstrap. 2016.** Bootstrap 3.3.6 released · Bootstrap Blog. [En línea] 2016. [Citado el: 15 de 2 de 2016.] http://blog.getbootstrap.com/2015/11/24/bootstrap-3-3-6-released/.

**Centers for medicare & medicaid services. 2008.** Centers for medicare & medicaid services. *Centers for medicare & medicaid services.* [En línea] 2008. [Citado el: 12 de 2 de 2016.] https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf.

**Dürr, Michael y Schlobinski, Peter. 2006.** *Deskriptive Linguistik: Grundlagen und Methoden.* s.l. : Vandenhoeck & Ruprecht, 2006. p. 301.

**Dutoit, T., y otros. 1996.** *The MBROLA Project: Towards a set of high quality speech synthesizers of use for non commercial purposes.* s.l. : ICSLP Proceedings, 1996.

**Fowler, Martin y Sccott, Kendall. 1999.** *UML Gota a Gota.* 1999.

**Gahlawata, M., Malika, A. y Bansalb, P. 2014.** *Natural Speech Synthesizer for Blind Persons Using Hybrid Approach.* 2014.

**Guerra, Y. F. 2009.** *Diseño de la plataforma soberana para el desarrollo de Sistemas de Información Geográfica del Polo Geoinformática.* La Habana : s.n., 2009.

**Herrera, Valencia, Ignasi, Joan y Pereira, Cao. 2008.** *Infraestructura de datos para la localizacion y acceso inteleigente a la informacion disponible a traves de servicios web y catálogos.* 2008.

**I. R., Titze. 2006.** *The Myoelatic Aerodynamic Theory of Phonation.* Iowa City : s.n., 2006.

**IEEE. 1990.** *Standard Glossary of Software Engineering Terminology.* 1990. ISBN 155937067X.

**Intel Corporation. 2016.** Assistive Context-Aware Toolkit (ACAT). [En línea] Intel, 2016. [Citado el: 16 de 2 de 2016.] https://01.org/acat.

**IVONA. 2012.** IVONA Text To Speech. [En línea] 2012. [Citado el: 15 de 1 de 2016.] http://www.ivona.com/.

**Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

**Knoll, Lars. 2011.** *The Qt Project is live!* 2011.

**Kominek, John y Black, Alan W. 2003.** *CMU ARCTIC databases for speech synthesis.* s.l. : Carnegie Mellon University, 2003.

**Ladefoged, Peter . 2005.** *Vowels and Consonants.* s.l. : Wiley-Blackwell, 2005. ISBN 978-1405124591.

**Lamel, L.F. , y otros. 1993.** *Proceedings ESCA-NATO Workshop and Applications of Speech Technology.* 1993.

**Larman, Craig . 2003.** *UML y Patrones.* 2003.

**Lemmetty, Sami. 1999.** *Review of Speech Synthesis Technology.* s.l. : Helsinki University of Technology, 1999.

**Loquendo. 2012.** Loquendo TTS. [En línea] 2012. [Citado el: 20 de 12 de 2015.] http://www.loquendo.com/es/technology/TTS.htm.

**Martínez, E. 2005.** Metodología. Hágase la luz. [En línea] 2005. [Citado el: 23 de 1 de 2016.] http://www.crisol.cc/eamftec/metodologias/Metodologias.pps.

**Molpeceres, Alberto. 2002.** Procesos de desarrollo: RUP, XP y FDD. [En línea] 2002. [Citado el: 18 de 1 de 2015.] http://www.javaHispano.org.

**Moreno Azcona, Gabriel Alejandro. 2008.** *Nueva Voz Concatenativa de Difonemas para el Español Mexicano en Festival.* Puebla : Universidad de las Américas, 2008.

**Muralishankar, R, Ramakrishnan, A.G. y Prathibha, P. 2004.** *Speech Communication.* 2004.

**Murillo Alfaro, Félix. 1999.** *Herramientas Case.* 1999.

**Music and Computers. 1993.** *Music and Computers.* s.l. : Dartmouth College, 1993.

**Netcraft. 2014.** February 2009 Web Server Survey | Netcraft. [En línea] Netcraft, 2014. [Citado el: 15 de 2 de 2016.] http://news.netcraft.com/archives/2009/02/18/february_2009_web_server_survey.html.

**PhpStorm. 2016.** PhpStorm FAQUE- PhpStorm - Confluence. [En línea] 2016. [Citado el: 20 de 2 de 2016.] http://confluence.jetbrains.com/display/PhpStorm/PhpStorm+FAQ.

**PosgreSQL-es. 2010.** PostgreSQL-es. [En línea] 2 de 10 de 2010. [Citado el: 17 de 2 de 2016.] http://www.postgresql.org.es/sobre_postgresql.

**Q-Success. 2016.** Usage Statistics and Market Share of JavaScript Libraries for Websites. [En línea] Q-Success, 2016. [Citado el: 18 de 2 de 2016.] http://w3techs.com/technologies/overview/javascript_library/all.

**QT Creator. 2015.** QT Creator. [En línea] 2015. [Citado el: 26 de 1 de 2016.] http://Blog.qt.digia.com/.

**Sánchez, Tamara Rodríguez. 2015.** *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana : s.n., 2015.

**ScanSoft. 2012.** ScanSoft. [En línea] 2012. [Citado el: 20 de 4 de 2016.] www.ScanSoft.com.

**Schröder, M. 2001.** *Emotional Speech Synthesis: A Review.* s.l. : University of the Saarland, 2001.

**Sommerville, I y Kontonya, G. 2002.** *Requirements Engineering: Processes and Techniques.* s.l. : John Wiley and Sons, 2002.

**Stevens, Perdita, Pooley, Rob y Wesley, Addison. 2012.** *Utilización de UML en Ingeniería del Software con Objetos y Componentes.* 2012.

**The Apache Software Foundation. 2016.** About the Apache HTTP Server Project - The Apache HTTP Server Project. [En línea] The Apache Software Foundation, 2016. [Citado el: 20 de 2 de 2016.] http://httpd.apache.org/ABOUT_APACHE.html.

**Visual Paradigm. 2013.** Visual Paradigm. [En línea] 2013. [Citado el: 20 de 2 de 2016.] www.visual-paradigm.com/product/vpuml.

**Yang Wang, William y Georgila, Kallirroi . 2011.** *Automatic Detection of Unnatural Word-Level Segments in Unit-Selection Speech Synthesis.* s.l. : IEEE ASRU, 2011.

**Zhang, Julia. 2011.** *Language Generation and Speech Synthesis in Dialogues for Language Learning.* 2011.

**Zukerman, Erez. 2012.** Slick PhpStorm Makes Editing JavaScript and PHP Fun. [En línea] 2012. [Citado el: 20 de 2 de 2016.] http://www.pcworld.com/article/248117/slick_phpstorm_makes_editing_javascript_and_php_fun.html.

**Zuluaga Giraldo, Carlos Alexander. 2011.** *Enterprise Architect y UML.* 2011.