PROCESAMIENTO DE LA GRAMÁTICA DE CONTEXTO DE UNA APLICACIÓN MÓVIL BASADA EN VOZ PARA CLIENTES ANDROID USANDO ONTOLOGÍAS DE DOMINIO

Diana Marcela Folleco Idrobo
Eliana Andrea Concha Agredo

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación de Servicios Avanzados de Telecomunicaciones
Popayán, Marzo de 2012

PROCESAMIENTO DE LA GRAMÁTICA DE CONTEXTO DE UNA APLICACIÓN MÓVIL BASADA EN VOZ PARA CLIENTES ANDROID USANDO ONTOLOGÍAS DE DOMINIO

Diana Marcela Folleco Idrobo

Eliana Andrea Concha Agredo

ANEXOS

Director

Francisco Orlando Martínez Pabón

Magister en Ingeniería, área Ingeniería Telemática

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Telemática

Línea de Investigación de Servicios Avanzados de Telecomunicaciones

Popayán, Marzo de 2012

CONTENIDO

ANEXO A..... 1

ESQUEMA PARA EVALUAR ONTOLOGÍAS ÚNICAS PARA UN DOMINIO DE CONOCIMIENTO..... 1

ANEXO B..... 6

ANÁLISIS Y DISEÑO DEL PILOTO DE PRUEBA. 6

1. MODELO DE CASOS DE USO..... 6

2. DIAGRAMA DE CLASES..... 9

3. DIAGRAMAS DE SECUENCIA 10

3.1 Caso de uso: ConfigurarIP 10

3.2 Caso de uso: Realizar consulta por voz..... 11

3.3 Caso de uso: Obtener información detallada 12

ANEXO C..... 13

MANUALES DE INSTALACIÓN Y CONFIGURACIÓN DE LAS HERRAMIENTAS EMPLEADAS EN LA CONSTRUCCIÓN DEL MECANISMO..... 13

1. INSTALACION Y CONFIGURACION DE POCKETSPHINX EN ANDROID..... 13

2. INSTALACION Y CONFIGURACION DEL SERVIDOR WEB 16

REFERENCIAS..... 18

LISTA DE FIGURAS

Figura 1. Criterios de evaluación de Ontologías. Fuente[1]. 1
Figura 2. Fases del Esquema de evaluación de ontologías. Fuente[1]. 2
Figura 4. Diagrama de casos de uso..... 6
Figura 5. Diagrama de clases del Sistema 10
Figura 6. Diagrama de secuencia del caso de uso: Configurar IP.....10
Figura 7. Diagrama de secuencia del caso de uso: Realizar consulta por voz. 11
Figura 8. Diagrama de secuencia del caso de uso: Obtener información detallada. 12
Figura 9. Diagrama de componentes del sistema.....12

LISTA DE TABLAS

Tabla 1.Descripción detallada del caso de uso:Configurar IP y puerto 6
Tabla 2. Descripción detallada del caso de uso: Realizar consultas por voz 8
Tabla 3. Descripción detallada del caso de uso: Obtener información detallada. 9

ANEXO A

ESQUEMA PARA EVALUAR ONTOLOGÍAS ÚNICAS PARA UN DOMINIO DE CONOCIMIENTO.

Como la ontología utilizada para el presente trabajo de grado es la única descripción conocida del dominio seleccionado (El arte), la evaluación de la misma se hizo utilizando el esquema propuesto en [1], método que permitió evaluar la calidad de la ontología en estado de modelado, y como única en el dominio específico del conocimiento seleccionado.

En [1] se resumen cada uno de los criterios más importantes que deben ser evaluados para cada uno de los autores de la literatura que ahí se referencian. Estos se muestran en la tabla:

Criterio / Autor	Taxonomía	Lenguaje	Aplicación	Vocabulario	Arquitectura Requerimientos	Aceptación Social	Razonamiento Automático	Software
(1)	√	-	√	√	-	-	-	-
(2)	-	√	√	√	-	-	√	-
(3)	√	-	√	√	-	-	-	-
(4)	-	√	√	√	-	√	-	-
(5)	√	√	√	√	√	-	-	-
(6)	√	√	√	√	√	-	-	√

(1) Brewster y cols, 2004 (2) Obrst y cols, 2007 (3) Porzel y Malaka, 2004
(4) Burton-Jones y cols, 2005 (5) Brank y cols, 2005 (6) Lozano-Tello, 2002 √ Considera - No considera

Figura 1. Criterios de evaluación de Ontologías. Fuente[1].

La mayoría de las propuestas coinciden en la evaluación de los siguientes criterios: a) la rigurosidad taxonómica, b) el lenguaje utilizado para la codificación, c) el rendimiento de las aplicaciones o tareas que utilizan las ontologías y d) el vocabulario utilizado para representar los conceptos y relaciones del dominio modelado. Es importante destacar que muchos de estos criterios se evalúan sobre la base de la comparación con otras ontologías disponibles, las cuales se usan como referencia [1].

Es por esta razón, y con la finalidad de solventar este último inconveniente, en [1] se propone un esquema para evaluar la calidad de una ontología única en un dominio de conocimiento y en estado de modelado.

Algunos objetivos del desarrollo de ontologías son: a) proporcionar una estructura de conocimiento común en un dominio, b) facilitar la reutilización del conocimiento y c) analizar el conocimiento. Estos objetivos se alcanzarán sólo si la ontología es de calidad y para garantizarlo es necesario que en cada fase del ciclo de vida de desarrollo, los Desarrolladores de Ontologías (DO) evalúen los resultados parciales.

Es posible valorar la calidad de una ontología sin la necesidad de recurrir a referencias de desarrollos previos, examinando un conjunto mínimo de criterios como: el vocabulario utilizado para representar

el conocimiento debe tener cobertura suficiente del corpus (conocimiento experto, textos y otras fuentes), la ontología debe estar escrita de manera correcta, sin errores y conforme a las reglas del lenguaje utilizado, la estructura taxonómica que organiza los conceptos y términos del dominio debe ser completa, sin redundancias y consistente, para que satisfaga los requerimientos para los cuales fue creada y, de manera particular, las preguntas de competencia deben ser respondidas adecuadamente.

El esquema de evaluación propuesto examina precisamente estos criterios de la manera siguiente:

1. *Uso correcto del lenguaje*: evaluar, con base en las características y reglas de construcción del lenguaje usado, la codificación de la ontología.
2. *Exactitud de la estructura taxonómica:* examinar la taxonomía considerando la consistencia, completitud y no redundancia de los conceptos y términos codificados.
3. *Validez del vocabulario:* evaluar el significado de los términos y conceptos a partir del conocimiento de expertos, recopilaciones de textos o cualquier otra fuente de conocimiento disponible del dominio.
4. *Adecuación a requerimientos:* validar si la ontología implanta los requerimientos preestablecidos y si responde a las preguntas de competencia.

El esquema está constituido por cuatro fases, una para cada criterio a evaluar, los cuales pueden aplicarse sin un orden rígido, dependiendo del ámbito del conocimiento. A continuación se describen las fases y las actividades que las constituyen.

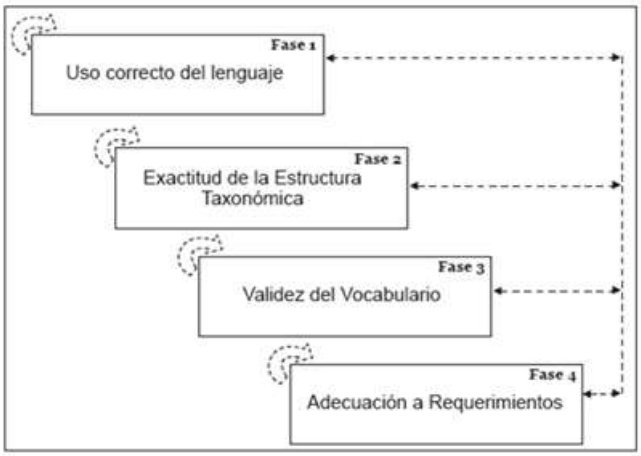


Figura 2. Fases del Esquema de evaluación de ontologías. Fuente[1].

Fase 1. Uso correcto del lenguaje

Es recomendable que el lenguaje seleccionado sea sólido (cualquier expresión pueda ser derivada a partir del conocimiento codificado) y completo (cualquier expresión que esté lógicamente implícita en la base de conocimiento pueda ser derivada). De esta forma, se pueden aplicar métodos de razonamiento sobre la ontología de manera satisfactoria [2]. También es importante que la escritura esté libre de errores o defectos, para garantizar su futura utilización de manera exitosa.

Las actividades que permitirán evaluar este criterio son:

- ✓ Validar que el lenguaje cumpla con estándares para desarrollos ontológicos, como: OWL (**O**ntology **W**eb **L**anguage),RDF(**R**esource **D**escription **F**ramework), DAML (**D**ARPA**A**gent **M**arkup **L**anguage), etc.
- ✓ Evaluar sintácticamente la ontología en cada fase del desarrollo. Se sugiere usar: las facilidades de los *test case (*piezas de código que especifican el correcto uso del lenguaje*)* de OWL[3] y RDF, el marco de prueba que provee el editor de ontologías Protégé-OWL y analizadores sintácticos como el de DAML o el de OWL, desarrollado en el proyecto WonderWeb [4].

Fase 2. Exactitud de la estructura taxonómica

Para examinar la rigurosidad de la estructura taxonómica que representa los conceptos, términos y clases del dominio, así como la naturaleza de las diferentes relaciones jerárquicas y semánticas, es necesario, y en algunas ocasiones, imprescindible, el conocimiento que sólo los expertos humanos pueden proporcionar.

La evaluación taxonómica considera el chequeo de inconsistencias, completitud y redundancia de los términos de la taxonomía [5]. Los errores más comunes son: clasificaciones semánticas incorrectas (clasificación de conceptos como subclase de una clase a la que no pertenecen), clases e instancias con diferentes nombres pero definiciones similares, omisión de conocimiento disjunto entre clases, ausencia de conceptos, redundancia de relaciones (clases con más de una relación de subclase), clases definidas como generalización o especializaciones de sí misma, entre otros.

Las actividades que se proponen en [4] a realizar en esta fase para identificar inconsistencias son las siguientes:

- Clases definidas como generalizaciones o especializaciones de sí mismas.
- Conceptos que no pertenecen a una clase en particular.
- Evaluar la completitud de los conceptos codificados en la ontología
- Ausencia de conceptos relevantes del dominio.
- Omisión de conocimiento disjunto entre clases de la estructura.
- Evaluar la existencia de redundancias en clases, instancias y relaciones.
- Clases e instancias con diferentes nombres, pero definiciones similares.
- Clases que tienen más de una relación de subclase.

Fase 3. Validez del vocabulario

Chequear que los términos codificados en la ontología existan y sean significativos en otras fuentes de conocimiento independientes, como por ejemplo, el conocimiento contenido en el *corpus* del dominio, entendiéndose por *corpus*, al conjunto más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación [6]

Las actividades de esta fase son las siguientes:

- Analizar el *corpus* del dominio: Identificar, extraer y organizar (en una tabla) los términos significativos del dominio a partir de los documentos. Es muy posible que en esta actividad sea necesaria la colaboración de los expertos.

- Evaluar el vocabulario considerando medidas de calidad de resultados usadas en escenarios de recuperación de información (búsqueda de documentos), tales como la precisión y el *recall* (exhaustividad) [3]. Los insumos para esta actividad serán la tabla construida en la primera actividad y el glosario de términos de la ontología.

Calcular Precisión: porcentaje de los términos de la ontología que aparecen en el *corpus* con relación a la cantidad total de términos de la ontología, utilizando la siguiente expresión:

Precisión = CO-C / COnto (1)

CO-C = Cantidad de términos que se solapan entre la ontología y el *corpus*.

COnto = Cantidad total de términos de la ontología.

Calcular *Recall*: porcentaje de términos del corpus que aparecen en la ontología con relación al total de términos en el corpus, utilizando la siguiente expresión:

Recall = CO-C / CCorp (2)

CCorp = Cantidad total de términos del *corpus*

En función de los valores obtenidos para la Precisión y el *Recall*, se establece una valoración cualitativa acerca de lo adecuado del vocabulario.

Fase 4. Adecuación a requerimientos

La principal actividad al inicio del desarrollo de una ontología, tal como sugieren los métodos de *Methontology* [7], consiste en la elaboración de un documento de requerimientos que especificará para qué se construye la ontología, cuál es la meta a alcanzar, descripción del dominio, posibles aplicaciones que la utilizarán, nivel de formalidad, fuentes de conocimiento disponibles, usuarios potenciales y escenarios de uso. Además, incluirá preguntas de competencia (posibles preguntas que la ontología deberá responder), las cuales serán utilizadas como un indicador del alcance y contenido del dominio representado.

Las actividades para evaluar la adecuación a los requerimientos son:

- Verificar que las especificaciones del documento de requerimientos se cumplan.
- Verificar que las respuestas proporcionadas por la ontología a las preguntas de competencias sean correctas y pertinentes.

Una vez realizada la evaluación, el desarrollador deberá chequear y analizar la validez de los resultados. En caso de no ser satisfactorios, se regresará a fases previas hasta alcanzar los resultados deseados, así, de manera incremental, se verificará la calidad de la ontología.

ANEXO B

ANÁLISIS Y DISEÑO DEL PILOTO DE PRUEBA.

1. MODELO DE CASOS DE USO

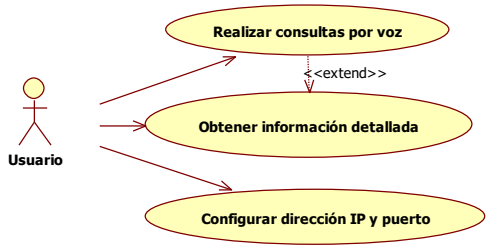


Figura 3. Diagrama de casos de uso

1.1 Descripción detallada Casos de Uso

Caso de uso	**Configurar Dirección IP y puerto**	
Objetivo	Fijar la dirección IP y el puerto al que va a conectarse la aplicación móvil y sobre los cuales se encuentra corriendo el Servidor Web.	
Actor /es	Usuario	
Descripción	Este caso de uso inicia después de que el usuario selecciona la opción "Configurar IP" del menú principal, en seguida se presentará una interfaz con dos campos editables de texto, en los cuales el usuario debe ingresar la dirección IP y el número de puerto del servidor Web.	
Pre-condiciones	Ninguna	
Pos-condiciones	Ir a la interfaz que permite realizar el reconocimiento de las consultas realizadas por voz.	
Flujo normal de eventos	Usuario	Dispositivo Móvil
	1. Oprime botón "Configurar IP" del menú principal.	
		2. Despliega en pantalla la interfaz de configuración.
	3. Digita la dirección IP y el número de puerto, finaliza guardando la configuración y retornando al menú principal.	
Flujos alternativos	Ninguno.	
Flujos de excepción	Ninguno.	

Tabla 1. Descripción detallada del caso de uso:Configurar IP y puerto

<table>
<tr><td>Caso de uso</td><td colspan="3">Realizar consultas por voz</td></tr>
<tr><td>Objetivo</td><td colspan="3">Realizar consultas a la aplicación, relacionadas con las Obras de Arte u Objetos de Arte, consultando todas las existentes por un Nombre, Periodo, Autor o Técnica Artística específicos.</td></tr>
<tr><td>Actor /es</td><td colspan="3">Usuario</td></tr>
<tr><td>Descripción</td><td colspan="3">Este caso de uso inicia después de que el usuario selecciona la opción “Consultar por voz” del menú principal, en seguida se presentará la interfaz que permite realizar la consulta a la aplicación, y por medio del botón “Presione para hablar” se realiza el proceso de reconocimiento, mostrando en pantalla la frase que finalmente fue reconocida en el cuadro de texto de dicha interfaz.</td></tr>
<tr><td>Pre-condiciones</td><td colspan="3">Que el usuario haya configurado adecuadamente la dirección IP y puerto sobre el que está corriendo el servidor.</td></tr>
<tr><td>Pos-condiciones</td><td colspan="3">Ir a la interfaz que despliega la lista de resultados de respuesta a las consultas realizadas.</td></tr>
<tr><td>Flujo normal de eventos</td><td colspan="3"><table>
<tr><td>Usuario</td><td>Dispositivo Móvil</td><td>Servidor</td></tr>
<tr><td>1. Oprime botón “Consultar por Voz” del menú principal.</td><td></td><td></td></tr>
<tr><td></td><td>2. Despliega en pantalla la interfaz de reconocimiento de voz.</td><td></td></tr>
<tr><td>3. Pronuncia la consulta a realizar mientras mantiene presionado el botón “Presione para hablar”.</td><td></td><td></td></tr>
<tr><td></td><td>4. Muestra en pantalla la frase reconocida y envía la consulta al servidor.</td><td></td></tr>
<tr><td></td><td></td><td>5. Se ha iniciado el servidor previamente y ha razonado y aplicado las reglas a la ontología, para almacenar dicho conocimiento inferido de nuevo en la misma. Recibe la consulta emitida desde el dispositivo móvil y realizar la respectiva consulta SQWRL a la ontología, para devolver los resultados obtenidos.</td></tr>
<tr><td></td><td>6. Despliega en</td><td></td></tr>
</table></td></tr>
</table>

		pantalla la lista de Obras u Objetos de arte respectiva que trae consigo el servidor como respuesta y la almacena en la base de datos interna del dispositivo "piloto1.db".	
Flujos alternativos	Que el usuario no desee realizar consultas por voz y cierre la aplicación.		
Flujos de excepción	Fallo en la comunicación con el servidor, que no permita realizar el procesamiento de la consulta emitida.		

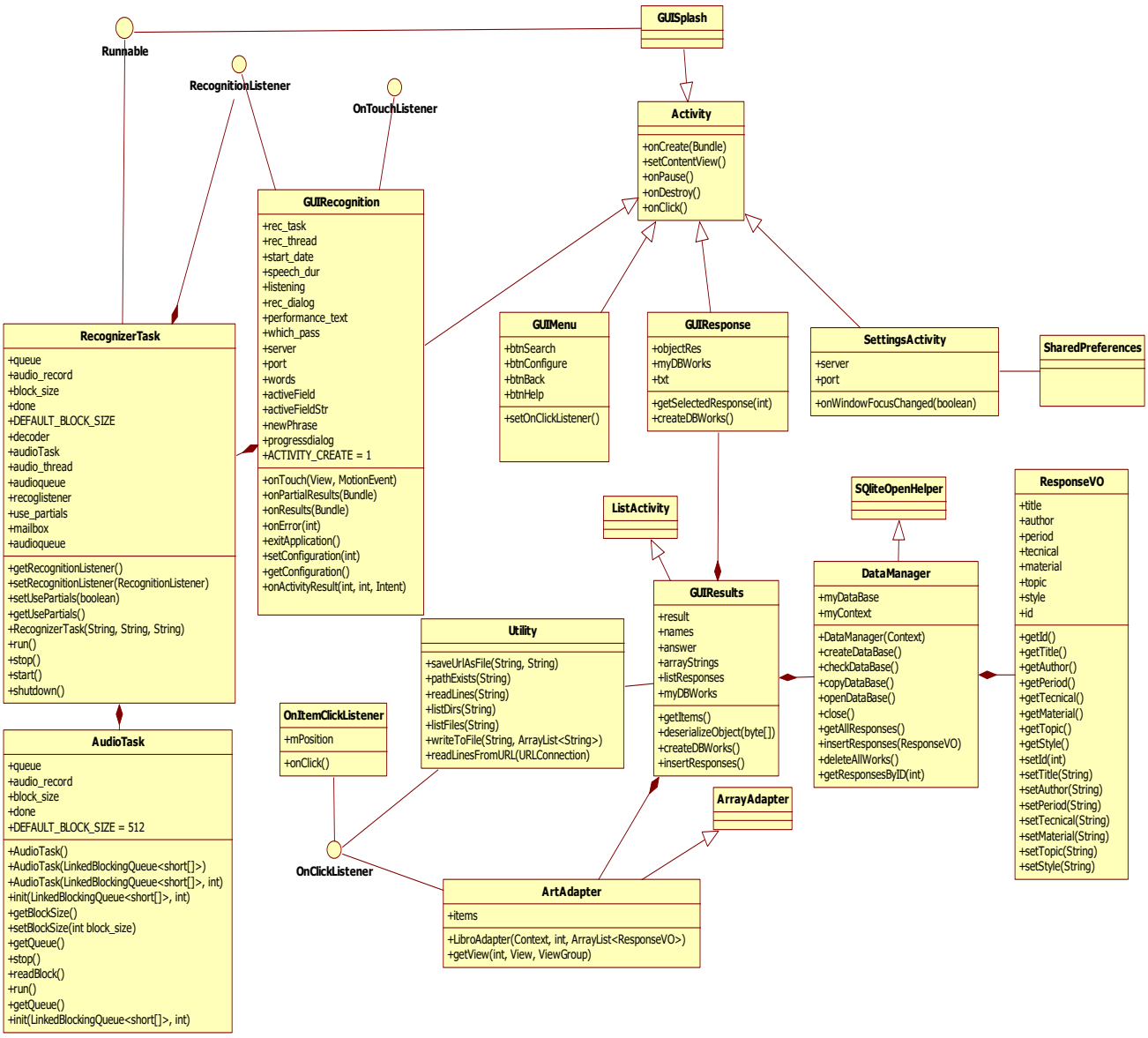
Tabla 2. Descripción detallada del caso de uso: Realizar consultas por voz

Caso de uso	**Obtener información detallada**		
Objetivo	Acceder a la información detallada de una Obra u Objeto de arte seleccionado, que se encuentre dentro de la lista respectiva que trae consigo el servidor como respuesta a las consultas que han sido emitidas desde el cliente.		
Actor /es	Usuario		
Descripción	Este caso de uso inicia después de que el usuario selecciona una opción de la lista obtenida en el anterior caso de uso, para obtener mayor información, la cual será desplegada en pantalla mostrando detalles más específicos de las Obras u Objetos de arte consultados.		
Pre-condiciones	Que el usuario haya realizado la consulta por voz y la lista de resultados se encuentre desplegada en pantalla.		
Pos-condiciones	Ir a la interfaz que despliega los detalles de la Obra u Objeto de arte seleccionado		
Flujo normal de eventos	Usuario	Dispositivo Móvil	Motor de base de datos SQLite
	1. Selecciona la obra u Objeto de arte de la lista desplegada en pantalla.		
		2. Obtiene el ítem seleccionado.	
			3. Realiza la consulta SQL a la base de datos "piloto1.db" para obtener la información detallada.
		3. Despliega en pantalla una interfaz con los detalles del ítem seleccionado, entre ellos: Autor, Periodo, Técnica, Material, Tema y	

		Estilo.	
Flujos alternativos	Ninguno.		
Flujos de excepción	Fallo en el almacenamiento y consulta de información a la base de datos.		

Tabla 3. Descripción detallada del caso de uso: Obtener información detallada.

2. DIAGRAMA DE CLASES



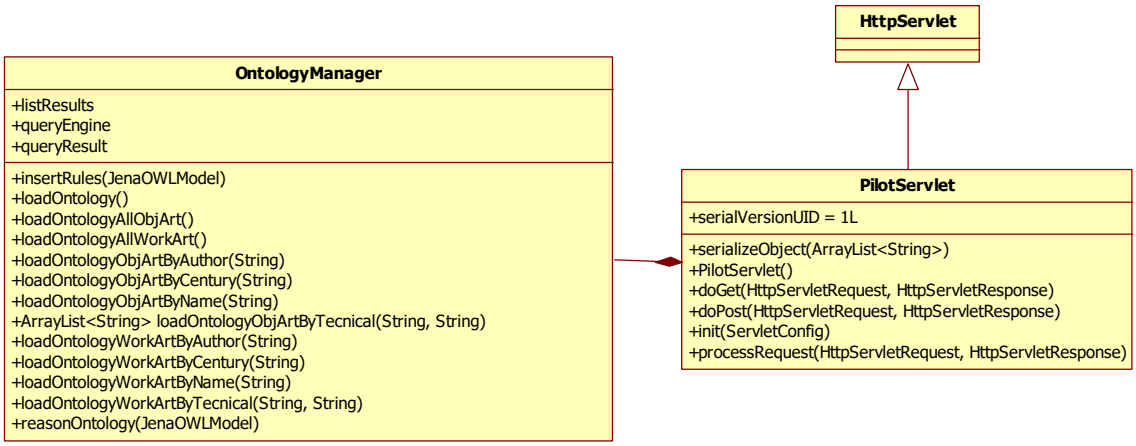


Figura 4. Diagrama de clases del Sistema

3. DIAGRAMAS DE SECUENCIA

3.1 Caso de uso: ConfigurarIP

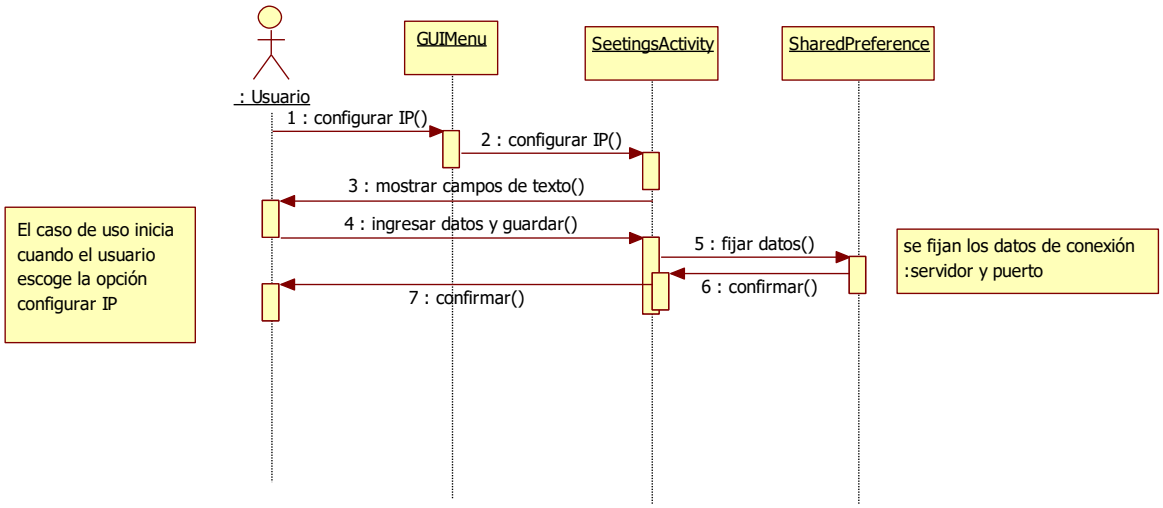


Figura 5. Diagrama de secuencia del caso de uso: Configurar IP.

3.2 Caso de uso: Realizar consulta por voz

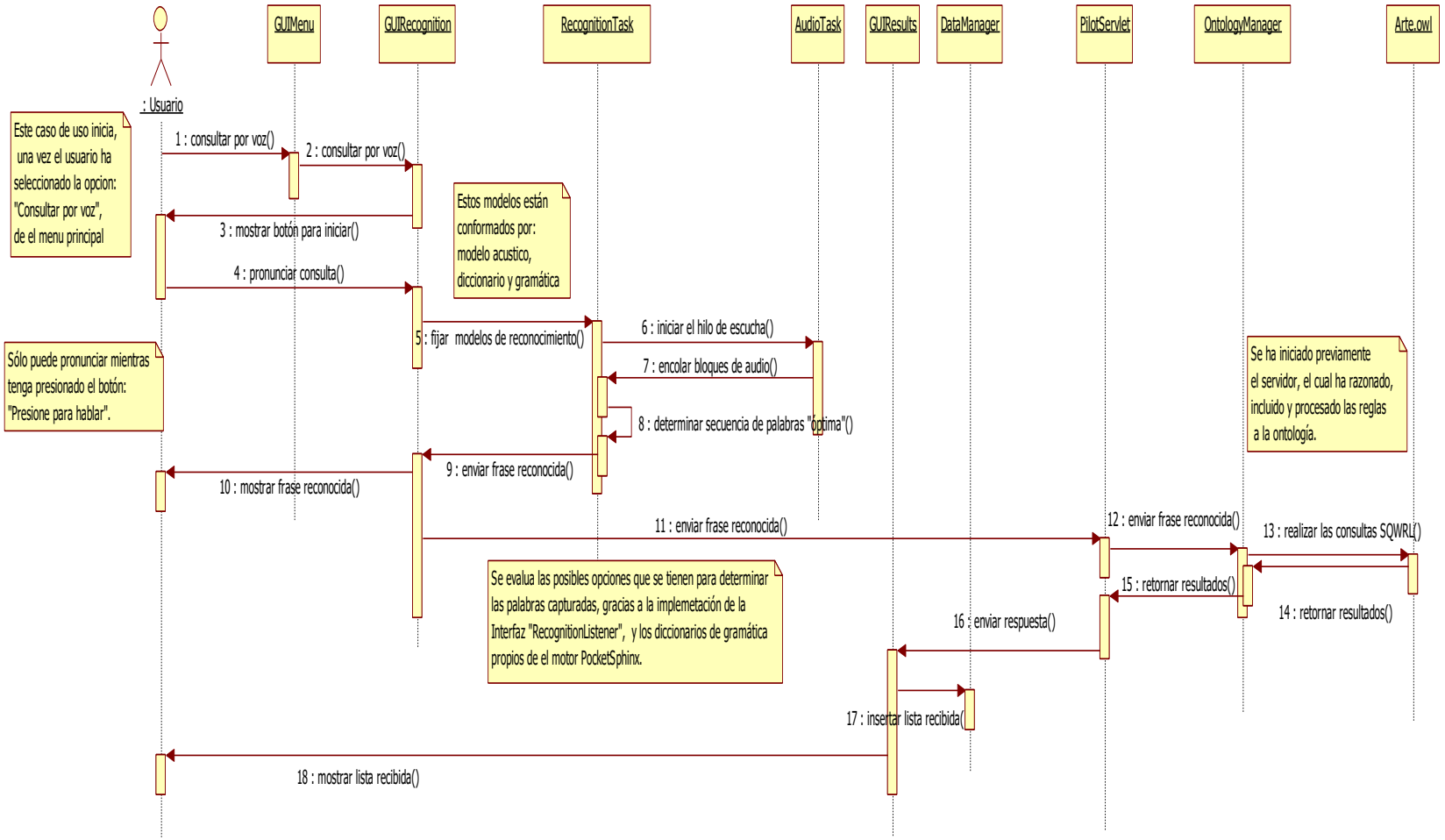


Figura 6. Diagrama de secuencia del caso de uso: Realizar consulta por voz.

3.3 **Caso de uso:** Obtener información detallada

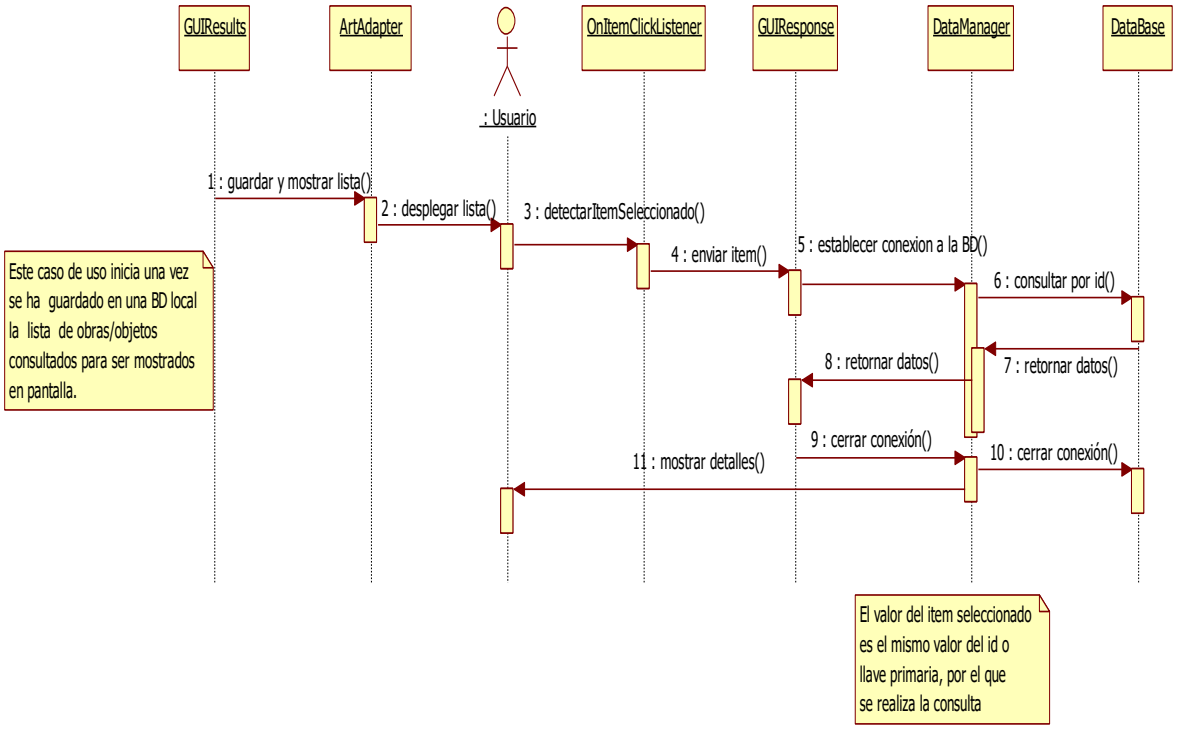


Figura 7. Diagrama de secuencia del caso de uso: Obtener información detallada.

4 DIAGRAMA DE COMPONENTES

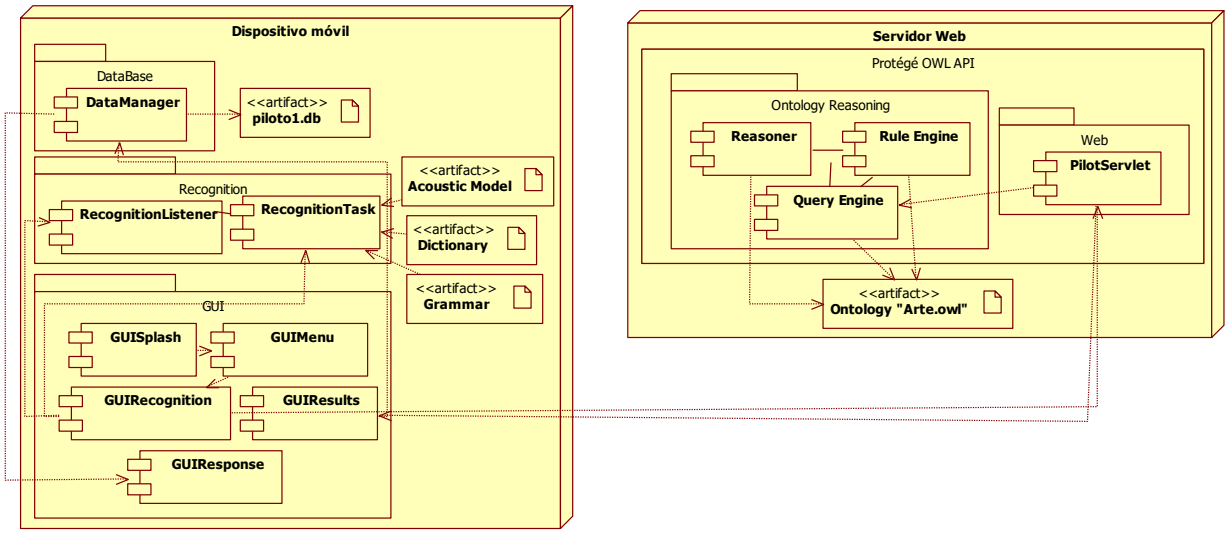


Figura 8. Diagrama de componentes del sistema.

ANEXO C

MANUALES DE INSTALACIÓN Y CONFIGURACIÓN DE LAS HERRAMIENTAS EMPLEADAS EN LA CONSTRUCCIÓN DEL MECANISMO

1. INSTALACION Y CONFIGURACION DE POCKETSPHINX EN ANDROID

a. Descargar y ejecutar PocketSphinx

Asegurarse de tener instalado Android NDK, en el caso contrario, descargarlo del siguiente enlace:

http://developer.android.com/sdk/ndk/index.html

Descargar la librería sphinxbase, que es común a todos los proyectos de CMUSphinx y necesaria para el funcionamiento de cualquiera de ellos, y asi como también descargar PocketSphinx, de la siguiente página:

http://cmusphinx.sourceforge.net/wiki/download/

Se recomienda descargar las últimas versiones disponibles para ambos software, en este caso:

- sphinxbase-0.7
- pocketsphinx-0.7

Descomprimir las carpetas en un mismo directorio.

Si el sistema operativo que se emplea es Windows, descargar Cygwin, que un software que proporciona un medio ambiente emulado Linux para Windows del siguiente enlace:

http://www.cygwin.com/

Después de extraer ambas carpetas, se deben instalar ambos software en el sistema operativo, para lo cual es necesario instalar “swig”:

apt-get install swig

Dirigirse a los directorios “sphinxbase-0.7” y “pocketsphinx-0.7” ejecutar los siguientes comandos para su instalación:

./configure

make
make install

b. Dirigirse a la carpeta 'jni' dentro del proyecto ClientePilotoTrabajoGradoV1.1 que se encuentra disponible dentro del soporte digital (CD).

   En el archivo 'Android.mk', ubicar la línea SPHINX_PATH, y especificar la ruta del directorio que contiene a 'pocketsphinx-0.7' y 'sphinxbase-0.7'. En seguida, ejecutar desde línea de comandos ndk-build para volver a generar el archivo libpocketsphinx_jni.so.

   the-path-to-your-ndk-folder/ndk-build –B

   Por supuesto sustituir la ruta real de la carpeta donde se encuentar instalado el ndk para el "the-path-to-your-ndk-folder".

c. Descargar el Modelo Acústico para la Lengua Española disponible en la página de CMU Sphinx:

   http://sourceforge.net/projects/cmusphinx/files/Acoustic%20and%20Language%20Models/Spanish%20Voxforge/voxforge-es-0.1.1.tar.gz/download

d. Descargar el diccionario de pronunciación del proyecto "The Automatic Speech Recognition for Spanish Language Project" del siguiente enlace:

   http://www.speech.cs.cmu.edu/sphinx/models/hub4spanish_itesm/proyectos/h4.dict

e. En el emulador/teléfono crear la carpeta 'edu.cmu.pocketsphinx' en la ruta '/sdcard/Android/data'. Dentro de la carpeta anteriormente creada, crear dos folders llamados 'hmm' y 'lm'. Ahora, dentro de la ruta '/sdcard/Android/data/edu.cmu.pocketsphinx/hmm' crear una carpeta llamada 'voxforge_es_sphinx', y copiar dentro de ella todos los archivos que se encuentran dentro de la ruta 'voxforge-es.0.1.1/model_parameters/voxforge_es_sphinx.cd_cont_1500'. Ahora dentro de la ruta '/sdcard/Android/data/edu.cmu.pocketsphinx/lm', copiar el diccionario 'h4.dict' descargado anteriormente, y crear un archivo de gramática con extesión ".jsgf", siguiendo la especificación JSGF del siguiente link:

   http://www.w3.org/TR/jsgf/

   Dicho archivo contendrá las frases que pueden ser pronunciadas por el usuario para un contexto específico. Para el caso del trabajo de grado se creó un archivo llamado "GrammarArt1.jsgf", que fue una instanciación de una ontología relacionada con el arte, disponible dentro del soporte digital (CD) entregado.

Así mismo, el diccionario descargado 'h4.dict' fue modificado ya que todas las palabras incluidas en la gramática no estaban en el diccionario, y fue necesario incluirlas junto con su respectiva pronunciación.

f. Ahora en Eclipse importar el proyecto ClientePilotoTrabajoGradoV1.1, y dentro de la clase 'RecognizerTask.java', observar que se estableció que se emplea un archivo de gramática (.jsgf) en vez de un archivo de lenguaje (.lm):

```
c.setString("-jsgf","/sdcard/Android/data/edu.cmu.pocketsphinx/lm/"+jsgf);
```

Dentro de la clase 'Cliente Piloto.java', observar que se fijan los nombres de los archivos a emplea para el reconocimiento: el modelo acústico, la gramática y el diccionario de pronunciación.

```
public void setConfiguration(int type){

String[] defaultConfig = new String[3];
switch(type){

        case 1:
        defaultConfig[0] = "voxforge_es_sphinx";
        defaultConfig[1] = "GrammarArt1.jsgf";
        defaultConfig[2] = "h4.dict";
        break;
```

g. Finalmente, corer la aplicación sobre el emulador/dispositivo, dando clic derecho sobre el proyecto Run As->Android Application.

h. En el dispositivo, comprobar que se tiene conexión de red y una dirección IP asignada, así mismo el Servidor debe encontrarse en el mismo dominio de red y comprobar conexión al mismo, colocando en un navegador web del dispositivo la siguiente dirección:

http://ipServidor:puertoServidor/AppServidorPiloto/PilotoServlet

En pantalla del dispositivo deberá aparecer 'Bienvenido al servidor Piloto', para comprobar la adecuada conexión al mismo.

i. Una vez iniciada la aplicación, seleccionar el botón "Configurar IP", para ingresar la dirección IP y el puerto sobre el que está corriendo el Servidor. En seguida iniciar el proceso de consultas seleccionando el botón "Consultar por voz".

2. INSTALACION Y CONFIGURACION DEL SERVIDOR WEB

Descargar e instalar el Editor Protégé 3.4.7 para la plataforma deseada, del siguiente enlace:

http://protege.cim3.net/download/old-releases/3.4.7/installanywhere/Web_Installers/

Para el caso del trabajo de grado, se descargó para la plataforma Windows. El editor Protégé 3.4.7 permite utilizar toda su interfaz gráfica para crear y gestionar ontologías.

a. Descargar y descomprimir la API Jena 2.6.4 del siguiente enlace:

   http://sourceforge.net/projects/jena/files/Jena/

b. Descargar y descomprimir el razonador Pellet 2.2.1 del siguiente link:

   http://clarkparsia.com/pellet/download

c. Descargar y descomprimir el motor de inferencia Jess de su página oficial:

   http://www.jessrules.com/jess/download.shtml

   En la anterior página dan 30 días de prueba para utilizar la librería e incluso se puede solicitar un año de prueba gratuita para uso académico (para el caso del trabajo de grado se solicitó la licencia académica), mientras que para uso comercial, requiere licencia.

d. Descargar y descomprimir el servidor Web Apache Tomcat v6.0:

   http://tomcat.apache.org/download-60.cgi

e. En primer lugar se debe tener bien instaladas y configuradas en el proyecto todas las librerías necesarias para trabajar con la API de Protege y Jess. Además, es necesaria la librería Jena, ya que Protege utiliza esta librería para realizar las operaciones de bajo nivel sobre la ontología.

   En concreto se necesitan:

   protege.jar y looks.jar *(Protege Core)*
   protege-owl.jar (*ProtegeOWL*)
   jess.jar (*Motor de inferencia*)
   swrl-jess-bridge.jar (*Puente de enlace entre el motor de inferencia y la ontología*)
   orphanNodesAlg.jar (*Requerida por ProtegeOWL en determinadas versiones*)
   Jena (*Todos los jars que componen la librería*)

Los dos primeros: ‘protege.jar’ y ‘looks.jar’ se encuentran en el directorio de instalación del editor Protégé 3.4.7 descargado anteriormente, ya que trae las versiones compiladas de las librerías de Protege-OWL.

La librería ‘jess.jar’ se encuentra dentro de la carpeta donde se descargó el motor Jess, anteriormente mencionado.
‘swrl-jess-bridge.jar’ es una librería necesaria para encapsular las funciones de bajo nivel de los motores de inferencia y se encuentra en el directorio: *<ruta-instalacion-protege 3.4.7>/plugins/edu.stanford.smi.protegex.owl/swlr-jess-bridge.jar.* En este mismo directorio se encuentra la librería **‘**orphanNodesAlg.jar’.
Y por último todos los jar que componen la librería Jena se encuentran dentro de la carpeta */lib* del directorio Jena2.6.4 descargado anteriormente.
Todas las anteriores librerías se deben incluir en la carpeta: *paht-in-your-computer\AppServidorPiloto\WebContent\WEB-INF\lib*, para que puedan ser correctamente cargadas en el momento de iniciar el Servidor.

f. Al abrir el Proyecto Web “AppServidorPiloto” disponible dentro del soporte digital entregado (CD), se debe fijar como servidor el Apache Tomcat descargado anteriormente.
Antes de correr el servidor, se debe iniciar el razonador Pellet a través de la interfaz DIG. En la línea de comandos se debe escribir lo siguiente:
>cd the-path-to-your-pellet-folder
>pellet dig

g. Para correr el servidor se debe dar clic derecho a la clase “PilotoServlet” y hacer clic en RunAS->Run on Server. En este momento el razonador se encarga de clasificar la ontología y la clase *SWRLRuleEngineBridge* proporcionada por Protégé, encargada de proporcionar un puente entre un modelo OWL con reglas SWRL y un motor de reglas, en éste caso el motor de inferencia Jess, importa las reglas incluidas en código y por tanto infiere axiomas a partir de ellas. El servidor queda a la espera de solicitudes HTTP provenientes de la aplicación cliente, para recibir la frase reconocida por la aplicación móvil y realizar la respectiva consulta SQWRL a la ontología por medio de la SQWRLQueryAPI para enviar la respuesta de nuevo l dispositivo móvil.

REFERENCIAS

[1] E. Ramos, *et al.*, "Esquema para evaluar ontologías únicas para un dominio de conocimiento," *Enlace*, vol. 6, 2009, pp. 57-71.

[2] L. Obrst, *et al.*, "The evaluation of ontologies," *Semantic Web*, pp. 139-158, 2007.

[3] D. L. McGuinness and F. Van Harmelen, "OWL web ontology language overview," *W3C recommendation*, vol. 10, 2004, pp. 2004-03.

[4] C. Masolo, *et al.*, "Wonderweb deliverable d17. the wonderweb library of foundational ontologies and the dolce ontology," research project in the IST Programme of the Commission of the European Communities as project number, 2002.

[5] A. Gomez-Perez, *et al.*, *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*: Springer Verlag, 2004.

[6] D. De la Lengua Española, "Real Academia Española," *Madrid*, 2001.

[7] M. Ehrig, *et al.*, "Measuring similarity between semantic business process models," in Proceedings of the fourth Asia-Pacific conference on Comceptual modelling 2007, pp. 71-80.