Revista Difusiones, ISSN 2314-1662, Num. 21, 2(2) julio-diciembre 2021, pp.92-110
Fecha de recepción: 29-10-2021. Fecha de aceptación: 09-11-2021

Ingeniería de requerimientos usando LEL en proyectos de cátedras de Administración de Proyectos y Sistemas de Información II de la Carrera Ingeniería en Informática del Departamento Académico San Salvador de la Universidad Católica de Santiago del Estero

Engineering of requirements using LEL in projects of the chair Information Systems II and Project Administration of the Bachelor's Degree in Computer Engineering of the San Salvador Academic Department of the Catholic University of Santiago del Estero

Eduardo Antonio Graneros[1]
egraneros@yahoo.com.ar
Universidad Católica de Santiago del Estero. Departamento Académico San Salvador,
San Salvador de Jujuy, Jujuy, Argentina

Fabiana Judith Chiquello[2]
faby_chiquello@yahoo.com.ar
Universidad Católica de Santiago del Estero. Departamento Académico San Salvador,
San Salvador de Jujuy, Jujuy, Argentina

Mirta Eulalia Blas[3]
me_blas@yahoo.com.ar
Universidad Católica de Santiago del Estero. Departamento Académico San Salvador,
San Salvador de Jujuy, Jujuy, Argentina

1 Ingeniero en Computación, Especialista en Enseñanza de la Educación Superior, Profesor en Computación, Docente de la Universidad Católica de Santiago del Estero Departamento Académico San Salvador.
2 Ingeniera en Computación, Especialista en Enseñanza de la Educación Superior, Profesora en Computación, Docente de la Universidad Católica de Santiago del Estero Departamento Académico San Salvador.
3 Ingeniera en Informática, Experta en Educación Superior, Docente de la Universidad Católica de Santiago del Estero Departamento Académico San Salvador.

Resumen

En la actualidad, el software ha ganado un amplio terreno a lo largo del mundo y se encuentra presente en diversos aspectos de la vida. Miles de sistemas informáticos se implementan a diario para resolver diversos tipos de problemas, y generan así, un amplio conjunto de infraestructuras que dependen de dichos sistemas. En este contexto, la ingeniería de software, se ha dedicado a definir un conjunto de procesos, métodos y herramientas que permitan a los desarrolladores generar software de calidad. Estos procesos representan un enfoque adaptable que permite a los desarrolladores especificar un conjunto apropiado de acciones y tareas para desarrollar una aplicación. Sumado a lo anterior, la aparición de UML como un estándar para la descripción de sistemas orientados a objetos y el desarrollo de metodologías ágiles han aportado un nuevo enfoque al desarrollo de software, estableciendo métodos efectivos de especificación, diseño e implementación. Por otra parte, existen herramientas para la automatización de diversas tareas involucradas a lo largo de proceso de desarrollo de software. Sin embargo, muy pocas dan soporte en los primeros procesos de la ingeniería en Requerimientos como en el proceso de elicitación de requerimientos. El Léxico Extendido del Lenguaje (LEL) permite modelar el vocabulario del sistema y, si bien existen trabajos de investigación realizados sobre la aplicación de LEL, muy pocos automatizan dicha práctica para poder obtener un diseño a partir de esta herramienta. En esta investigación se utilizó la herramienta para elicitación de requerimientos LEL junto con un prototipo de compilador/traductor gUML, en los trabajos de cátedras de Sistemas de Información II y Administración de Proyectos de la carrera de Ingeniería en Informática del Departamento Académico San Salvador de la Universidad Católica de Santiago del Estero (DASS/UCSE).

Palabras clave

Ingeniería de Requerimientos, LEL, Compiladores

Abstract

Nowadays, software has gained ground throughout the world and is present in various aspects of life. Thousands of computer systems are deployed daily to solve several problems, thus generating a wide array of infrastructures that depend on such systems. In this context, software engineering has focused on defining a set of processes, methods, and tools that allow developers to generate quality software. These processes represent an adaptive approach that allows developers to specify an appropriate set of actions and tasks to develop an application. Besides, the emergence of UML as a standard for the description of object-oriented systems and the development of agile methodologies have brought a new

approach to software development, establishing effective methods of specification, design, and implementation. On the other hand, there are tools for automating various tasks involved throughout the software development process. However, very few support the first processes of engineering in Requirements as in the elicitation of requirements. The Extended Lexicon of the Language (Léxico Extendido del Lenguaje, LEL) allows modeling the system vocabulary, and although there are research works carried out on the application of LEL, very few automate this practice to be able to obtain a design from this tool. In this research, the LEL requirement elicitation tool was used together with a prototype gUML compiler/translator, in the work of chairs of Information Systems II and Project Administration of the Bachelor's Degree in Computer Engineering of the San Salvador Academic Department of the Catholic University of Santiago del Estero (DASS/UCSE).

Key Words

inverter, photovoltaic plant, jujuy, puna

Introducción

Lograr calidad en un software no es sencillo, depende del dominio de aplicación, la complejidad y tipo del sistema, de los costos generados para el análisis y diseño del sistema y de la forma de gestionar los riesgos que pueden surgir a lo largo del proceso, entre otros factores. Una de las principales desventajas del uso de metodologías tradicionales o no ágiles para el desarrollo de software, recaen en que uno de los errores generados en las etapas más tempranas del desarrollo afectan gravemente a los procesos finales del ciclo de vida del software.

La etapa de relevamiento de requerimientos es un área clave para el éxito de un proyecto. En particular, el construir una especificación de requerimientos (SRS, por sus siglas en inglés) adecuada es esencial, ya que constituyen un medio de comunicación entre stakeholders y miembros del equipo de desarrollo, y es necesario que el equipo de desarrollo posea los requerimientos correctos para construir el producto. (Antonelli, Rossi, Leite, y Oliveros, 2012).

Además de los problemas comunes que debe afrontar un equipo de desarrollo al momento de recopilar los requerimientos que debe cumplir un determinado software, estos deben estar formalizarlos en diagramas o modelos que permitan representar las funciones que el software va a desarrollar. A su vez, tanto los diagramas como lo requerimientos, deben guardar consistencia entre sí, es decir, que los esquemas realizados deben garantizar que están representando a los requerimientos especificados y estos a su vez, deben ser fieles a las necesidades del cliente (Pons y Pérez, 2008).

Muchas veces esta consistencia es difícil de lograr debido a la ambigüedad propia del lenguaje natural, lo cual incurre en gastos de tiempo y recursos por parte de los desarrolladores para poder verificar dicha propiedad.

El Léxico Extendido del Lenguaje (LEL) es un glosario que permite describir el lenguaje del dominio utilizando el lenguaje natural. Entre las características más importantes de esta herramienta se pueden mencionar la facilidad para aprenderla y usarla como así también se ha demostrado que el LEL es útil en dominios complejos, en donde se notó como los involucrados lograron un conocimiento del dominio. (Antonelli, Rossi, Leite, y Oliveros, 2013).

El Lenguaje Unificado de Modelado (UML) es un lenguaje gráfico para visualizar, construir, documentar y proporcionar una forma estándar de representar los componentes de un sistema; comprende tanto elementos conceptuales, como los procesos de negocio y las funciones del sistema y elementos concretos, como las clases escritas en un lenguaje de programación específico y componentes software reutilizables. (Booch, Rumbaugh y Jacobson, 2006).

Los diagramas de casos de uso pertenecen a UML y tienen como propósito especificar el comportamiento de una parte del sistema, sin embargo, no describe la estructura interna del sistema o sus detalles de implementación. Un caso de uso se dibuja como una elipse con

un nombre dentro que identifica al caso de uso, especificando de manera clara la funcionalidad que representa (por lo general usando verbos en infinitivo). Estos, se conectan mediante líneas de trazo continuo con los actores que se comunican con él.

Un diagrama de clase es un tipo de diagrama de estructura estática de UML que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

Una mayor abstracción y automatización son claves para dominar la complejidad inherente al proceso de construcción de software, y con los modelos se pretende obtener una reducción del salto semántico entre la forma en la que los desarrolladores piensan en las soluciones y la forma en la que deben expresarlas, la cual redunda en un mejor esfuerzo en la tarea de programar y por tanto en una mayor productividad, en programas más comprensibles y en un mantenimiento menos costoso. (Molina, García Rubio, Pelechano, Vallecillo, Vara y Vicente-Chicote,2008)

Un compilador es un programa que puede leer un programa en un lenguaje (el lenguaje fuente) y traducirlo en un programa equivalente en otro lenguaje (el lenguaje destino) (Aho, Ullman, y Sethi, 2017). De esta forma, un lenguaje objeto puede representar un lenguaje abstracto específico del área informática (por ejemplo: UML). Por otra parte, un lenguaje fuente, puede representar por ejemplo a un subconjunto del lenguaje natural, con un grupo de palabras reservadas y bajo ciertas restricciones en cuanto a su estructura para la redacción de oraciones que permitan describir un requerimiento funcional dentro del proceso de Análisis y Desarrollo de Software.

Dentro del proceso de compilación, existen dos subprocesos, el proceso de análisis y el de síntesis. Durante el proceso de análisis se divide el lenguaje fuente en componentes imponiendo una estructura gramatical sobre ellos y mediante los cuales se crea una representación intermedia del programa fuente, a su vez, se recolecta información sobre lenguaje fuente y se almacena en una estructura de datos denominada tabla de símbolos. Luego, el proceso de síntesis, utilizando la representación intermedia y la información de la tabla de símbolos, construye el programa deseado, expresado en lenguaje objeto.

Este trabajo consistió en estudiar e implementar técnicas y herramientas que permitan mejorar la comunicación entre los stakeholdes y los miembros del equipo de desarrollo, concretamente se trabajó sobre los proyectos de cátedra de las materias Sistemas de Información II y Administración de Proyectos de la carrera de Ingeniería en Informática del DASS/UCSE. Para los diferentes proyectos de desarrollo propuestos por los alumnos se les solicitó que escriban el LEL y luego utilicen el compilador traductor gUML, con las nuevas funcionalidades desarrolladas, para que facilite el pasaje de vocabularios del sistema modelado en LEL, hacia un diseño Orientado a Objetos.

Objetivos

Los objetivos generales de esta investigación fueron incorporar prácticas de elicitación de requerimientos con LEL utilizando el prototipo de compilador / traductor gUML en los proyectos de cátedras de Sistemas de Información II y Administtración de Proyectos de la carrera de Ingeniería en Informática del DASS/UCSE; refinar el compilador / traductor gUML mejorando su diseño e incorporando nuevas funcionalidades.

Como objetivos específicos implementar y documentar la aplicación de LEL junto con el compilador traductor gUML en los proyectos de cátedras, favorecer los procesos de enseñanza y de aprendizaje mediante la implementación del compilador / traductor como apoyo a la especificación de requerimientos en dichos proyectos de cátedra. Promover el intercambio y capacitación entre docentes interesados que puedan requerir de esta herramienta propuesta en la investigación.

LEL (Léxico Extendido del Lenguaje)

El LEL es un modelo que permite representar y documentar, con tecnología hipertextual un conjunto de símbolos que representan el lenguaje de la aplicación, sin necesidad de entender el problema. Consiste en una descripción de los términos significativos del macrosistema, acotando el lenguaje externo con el uso de símbolos definidos en el mismo LEL y a su vez, minimiza el uso de símbolos externos al lenguaje de la aplicación. El lenguaje de la aplicación es el lenguaje usado por los actores en todas las etapas del desarrollo y es una extensión del lenguaje natural usado normalmente. Una forma de obtener el lenguaje de aplicación es capturar los términos propios del mismo.

Este lenguaje puede contener palabras y construcciones específicas del ambiente de la aplicación, o bien, expresiones del lenguaje natural con un significado distinto del corriente. Lo que se busca con el LEL es identificar estos símbolos, para su posterior definición con ayuda de los usuarios.

El principal objetivo del LEL es conocer el vocabulario del usuario. Que el usuario y el desarrollador compartan el mismo lenguaje asegura la comunicación entre ambos en particular.

En el proceso de Ingeniería de Requerimientos, la validación de los diferentes productos requiere una fuerte interacción con el usuario la que se ve facilitada por el vocabulario común usuario-desarrollador, (Hadad, G., Kaplan G., Oliveros, A. y Leite J., 1996). Otro objetivo de la utilización del LEL, es la posibilidad de utilizarlo como herramienta para la capacitación de los miembros del equipo de desarrollo que se incorporan en etapas avanzadas del desarrollo, basado esto en una documentación consistente de los símbolos que representan el lenguaje de la aplicación.

Características del LEL

El LEL está compuesto por un conjunto de símbolos que identifican el lenguaje de la aplicación. Los símbolos son, en general, las palabras o frases utilizadas por el usuario y que repite con más frecuencia. También se incluyen aquellas palabras o frases que son relevantes para el dominio del problema, más allá de su frecuencia de repetición. Los símbolos se adquieren, por ejemplo, de entrevistas, observaciones o lectura de documentos. Con esta información se crea una lista con todos los símbolos reconocidos.

Durante el proceso de recolección, el ingeniero de software procura entender el significado de cada símbolo. La semántica de cada símbolo se representa con una o más nociones y uno o más impactos, éste último puede no existir. La noción indica qué es el símbolo y el impacto, cómo repercute en el sistema. Por lo tanto, cada símbolo tiene un "nombre" que lo identifica, una "noción" y un "impacto" que lo describen.

Análisis y resultados del trabajo realizado

El primer objetivo general junto con los objetivos específicos fueron alcanzados satisfactoriamente, se pudo realizar una mejora en el prototipo del compilador traductor gUML con el cual se obtuvieron resultados muy favorables con la traducción del LEL a un diseño Orientado a Objeto más específicamente a Diagramas de Clases, cumpliendo con las funcionalidades esperadas, si bien el compilador no se encuentra en su versión definitiva en donde se pretende integrar más diagramas de UML, fue posible que los alumnos interactúen con esta herramienta, cumpliendo con los objetivos.

Con respecto a los objetivos específicos se obtuvieron resultados positivos en cuanto a la enseñanza de LEL en las materias Sistemas de Información II y Administración de Proyectos, se realizó una cartilla con los temas introductorios a la elicitación de requerimientos utilizando LEL también se realizaron pruebas del prototipo del compilador traductor gUML esto se pudo observar en los trabajos realizados por los alumnos de dichas materias. A continuación, un ejemplo de una especificación en LEL de un proyecto de cátedra de Administración de Proyectos para la solicitud y asignación de certificados de residencia de la comisaría sexta de la policía de la provincia de Jujuy.

Sujetos

SUJETO: <SOLICITANTE>

Noción: <usuario cuya solicitud fue aceptada>

Tiene un nombre

Tiene un DNI

Puede tener una fecha de solicitud

Solicitante que accede a visita de oficial visitante

Impactos:
Incluye solicitar visita
Incluye cancelar visita
Puede incluir consulta de horario de visita
Solicitante hace consulta de oficial visitante
<Tener conocimientos que quienes son usuarios y quienes son los solicitantes>

SUJETO: <SISTEMA>
Noción: <Sistema es un software de la comisaria>
Sistema sistematiza procesos
Impactos:
<Sistema que permite la sistematización de procesos >
Incluye calcular posibles rutas
Incluye asignación de horarios

SUJETO: <OFICIAL VISITANTE>
Noción: <personal de la comisaria encargado de visitar solicitante>
Oficial visitante que presta servicio al solicitante
Tiene un código de identificación
Impactos:
Incluye una visita al solicitante
<Contiene una verificación del solicitante>

SUJETO: <OFICIAL OPERADOR>
Noción: <personal de la comisaria selecciona ruta adecuada>
Tiene un código de operador
Impactos:
<Permite realizar la verificación de solicitudes y ruta>
Incluye elegir una ruta
Incluye aprobar una solicitud
Puede contener el registro de fecha de visita

Objetos
OBJETO: <CERTIFICADO>
Noción: <Es documento de carácter legal>
Certificado es un documento que certifica el buen estado de un solicitante por parte de oficial de visitante
Tiene un numero

Tiene un nombre de solicitante
Tiene una fecha de emisión
Puede tener un periodo de validez
Impactos:
Incluye la recepción del certificado por parte del solicitante
< Permite a los solicitantes acceder a diversos servicios y derechos>

OBJETO: <SOLICITUD>
Noción: <Son los datos necesarios para registrar solicitante >
Solicitud tiene los datos que serán verificados para registrar visita
Tiene un numero
Tiene un nombre del solicitante
Tiene una fecha de emisión
Tiene un estado de solicitud
Tiene una posición geográfica para saber dónde realizar la visita
Tiene un motivo por el cual se realiza la solicitud
Impactos:
Información que necesita el sistema para autorizar visita>
Incluye el envio de solicitud de los usuarios
Incluye la aceptación de solicitud por el Oficial operador
Incluye el rechazo de solicitud por el Oficial operador

OBJETO: <NOTIFICACIÓN>
Noción: <Es un aviso de próxima visita domicilio>
La notificación contiene información importante para el solicitante
Tiene un numero de solicitud
Tiene un mensaje
Tiene un estado para saber si el usuario recibió la notificación.
Impactos:
<Informa al solicitante de la futura visita>
Incluye el envió de la notificación por parte del Sistema

OBJETO: <RUTA>
Noción: <Es una secuencia de direcciones>
Tiene direcciones
Puede tener un tiempo estimado
Impactos:
Incluye la generación de rutas por parte del sistema

Incluye la elección de una ruta por parte del operador
<Información que usa el oficial visitante para realizar visita>

OBJETO: <FIRMA>
Noción: <Tiene una marca >
Firma da fe que el oficial visitante verifico estado de solicitante
Impactos:
Incluye el ingreso de la firma por parte del Oficial visitante.
< garantiza que el certificado es válido >

OBJETO: <DOMICILIO>
Noción: <Lugar en el que el SOLICITANTE espera recibir al OFICIAL VISITANTE para recibir CERTIFICADO >
Tiene una dirección
Tiene un numero de solicitante al que pertenece esa dirección.
Impactos:
< Lugar donde se realiza el ingreso de datos >
Incluye la carga de certificado por el oficial visitante

OBJETO: <COMISARIA>
Noción: < Edificio donde la institución policial brinda diversos servicios a la comunidad.>
Impactos:
Tiene un numero
Tiene una dirección
Tiene un responsable
< Lugar desde donde se realiza el TRAZADO>

Verbos
VERBO: <VISITAR SOLICITANTE >
Noción: <OFICIAL VISITANTE se presentar en DOMICILIO>
Impactos:
Oficial operador despacha oficial visitante
Oficial visitante se presenta en domicilio
Solicitante verifica identidad de oficial visitante
< certificar buen estado y firmar certificado >

VERBO: < COMPROBAR DATOS>
Noción: < comprobación de datos de solicitud>

Oficial operador revisa los datos ingresados en la SOLICITUD
Impactos:
Oficial operador verifica si datos están correctos
Oficial operador verifica si el ASUNTO es meritorio de una visita
<autoriza visita y agrega dirección a destino a visitar>
VERBO: < TRAZAR RUTA>
Noción: < Sistema evalúa direcciones registradas y genera rutas>
Impactos:
Oficial operador se identifica en sistema
Oficial operador solicita trazar ruta
Sistema genera rutas disponibles
Oficial operador selecciona ruta
< genera rutas para recorrido >

VERBO: <FIRMAR CERTIFICADO >
Noción: < se firma el certificado >
oficial visitante firma certificado
Impactos:
Oficial visitante verifica identidad
Oficial visitante certifica buen estado
Oficial visitante firma certificado
<valida el certificado>

VERBO: < SOLICITAR VISITA>
Noción: <usuario ingresa y envía datos de solicitud >
Impactos:
Usuario ingresa datos de solicitud
Usuario envía datos de solicitud
<registrar visita>

VERBO: < CERTIFICAR BUEN ESTADO>
Noción: < oficial visitante verifica que el solicitante se encuentra en buen estado >
Impactos:
Oficial visitante comprueba visualmente estado de solicitante
Oficial visitante registra buen estado
<validar condición de solicitante>

VERBO: < REGISTRAR SOLICITUD>

Noción: < oficial operador registra solicitud>
Impactos:
Oficial operador registra nuevo destino
Oficial operador registra nuevo solicitante
<registrar nuevo destino y solicitante >

VERBO: < VERIFICAR DATOS>
Noción: < oficial operador controla solicitud>
Oficial operador controla que el formato de los datos de la solicitud es correcto
Impactos:
Oficial operador verifica solicitud
< evitar inconsistencias en los datos >

VERBO: < VERIFICAR IDENTIDAD>
Noción: < corrobora que los datos de la solicitud>
oficial operador corroborar que los datos de la solicitud correspondan al solicitante entrevistado
Impactos:
Oficial operador busca datos de solicitante
Oficial operador confirma identidad de solicitante
< confirmar identidad de solicitante >

Compilador traductor gUML

gUML lee un conjunto de requerimientos especificados en un lenguaje gramatical acotado de la lengua española y descritos en un archivo .txt, el cual es examinado para determinar que ha sido redactado correctamente siguiendo las restricciones y demás particularidades que presenta el compilador traductor gUML. Después de la fase de análisis (léxico, sintáctico y semántico), si el programa fuente es correcto, se genera el código intermedio en un archivo de salida de extensión .txt, el cual sirve como entrada a una aplicación encargada de traducir el código y generar el gráfico correspondiente.

Estructura de gUML

gUML es un traductor de dos pasadas, en la primera pasada se realiza el proceso de análisis, y en la segunda se encarga de traducir el código intermedio a partir del cual generar el diagrama correspondiente. De esta forma se distinguen 6 etapas bien diferenciadas asignadas a las aplicaciones que conforman gUML de la siguiente manera:
Compilador (Fase de Análisis):

- Análisis Léxico
- Análisis Sintáctico
- Análisis Semántico
- Generación de Código Intermedio

Traductor a DOT (Fase de Síntesis):

- Análisis del código intermedio
- Generar diagrama.

Mediante la implementación de código intermedio, es posible reemplazar la fase de síntesis por otros traductores. Para el presente proyecto, se desarrolla un traductor adicional, para la herramienta web.

Pruebas del compilador traductor gUML en proyectos de cátedra

A continuación, un ejemplo de la utilización del compilador traductor gUML por parte de los alumnos de Administración de Proyectos para la solicitud y asignación de certificados de residencia de la policía de Jujuy. El prototipo del compilador traductor gUML desarrollado durante el proyecto de investigación no se puede considerar un producto final; si bien es funcional tiene tareas pendientes tales como:

- Desarrollo de interfaces más amigables para el usuario.
- Integración del proceso de análisis y síntesis en un solo producto.
- Reconocimiento de caracteres especiales (el carácter

Para la siguiente prueba se toma como datos de entrada la especificación definida en LEL para la solicitud y asignación de certificados de residencia.

La misma se realiza a partir de los casos de usos obtenidos del LEL, teniendo en cuenta, los sujetos, objetos y verbos definidos.

- El "Sistema" debe permitir al "Oficial Visitante" "Visitar al Solicitante"
- El "Sistema" debe permitir al "Oficial Operador" "Comprobar los Datos" de una solicitud
- El "Sistema" debe permitir al "Oficial Operador" "Trazar Ruta" de recorrido
- El "Sistema" debe permitir al "Oficial Visitante" "Firmar un Certificado"
- Para firmar un certificado el "Sistema" debe permitir al "Oficial Visitante" "Certificar el Buen Estado"
- El "Sistema" debe permitir al "Usuario" "Solicitar Visita"
- El "Sistema" debe permitir al "Oficial Operador" "Registrar una Solicitud"
- Para dicha registración el "Sistema" debe permitir al "Oficial Operador" "Verificar los Datos"
- Ademas el "Sistema" debe permitir al "Oficial Operador" "Verificar la Identidad"

Ejecución del Compilador

En la siguiente figura se muestra la ejecución del Compilador en una ventana de comando.

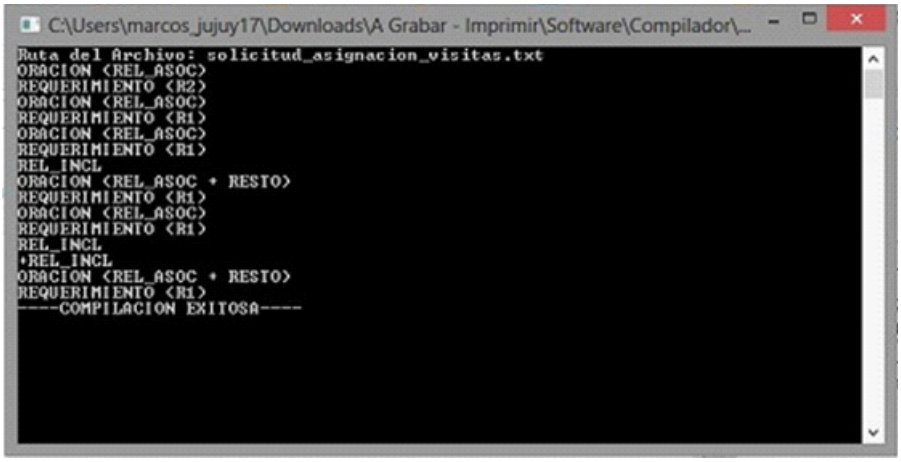

Figura 1: Ejecución del compilador traductor gUml.

Código Intermedio

A continuación, el código intermedio generado por la ejecución del compilador traductor gUML:

```
SOLICITANTE,0,1;
NOMBRE,1,3;
DNI,2,3;
FECHA DE SOLICITUD,3,3;
ESTADO DE SOLICITUD,4,3;
SOLICITAR VISITA,5,4;
CANCELAR VISITA,6,4;
CONSULTA DE HORARIO DE VISITA,7,4;
SISTEMA,8,1;
CALCULAR POSIBLES RUTAS,9,4;
ASIGNACION DE HORARIOS,10,4;
OFICIAL VISITANTE,11,1;
CODIGO DE IDENTIFICACION,12,3;
VISITA AL SOLICITANTE,13,4;
VERIFICACION DEL SOLICITANTE,14,4;
OFICIAL OPERADOR,15,1;
CODIGO DE OPERADOR,16,3;
ELEGIR UNA RUTA,17,4;
APROBAR UNA SOLICITUD,18,4;
REGISTRO DE FECHA DE VISITA,19,4;
CERTIFICADO,20,2;
NUMERO,21,3;
NOMBRE DE SOLICITANTE,22,3;
FECHA DE EMISION,23,3;
PERIODO DE VALIDEZ,24,4;
RECEPCION DEL CERTIFICADO,25,4;
SOLICITUD,26,2;
NUMERO,27,3;
NOMBRE DEL SOLICITANTE,28,3;
FECHA DE EMISION,29,3;
```

```
ESTADO DE SOLICITUD,30,3;
POSICION GEOGRAFICA,31,3;
MOTIVO,32,3;
ENVIO DE SOLICITUD,33,4;
ACEPTACION DE SOLICITUD,34,4;
RECHAZO DE SOLICITUD,35,4;
NOTIFICACION,36,2;
NUMERO DE SOLICITUD,37,3;
MENSAJE,38,3;
ESTADO,39,3;
ENVIO DE LA NOTIFICACION,40,4;
RUTA,41,2;
DIRECCIONES,42,3;
TIEMPO ESTIMADO,43,3;
GENERACION DE RUTAS,44,4;
ELECCION DE UNA RUTA,45,4;
FIRMA,46,2;
MARCA,47,3;
INGRESO DE LA FIRMA,48,4;
DOMICILIO,49,2;
DIRECCION,50,3;
NUMERO DE SOLICITANTE,51,3;
```

Figura 2: Código Intermedio generado por gUML.

Traducción del Código Intermedio, traducción a yUML

yUML es una herramienta online que permite crear diagramas UML a partir de texto plano. Los diferentes tipos de diagramas que se pueden realizar son diagramas de casos de uso, diagramas de clases y diagramas de actividad (Harris,2018).

Para el caso de solicitud y asignación de certificados de residencia, el compilador traductor gUML genera el siguiente código interpretable por la herramienta yUML:

```
[SOLICITANTE|nombre;dni;fecha_de_solicitud;estado_de_solicitud|solicitar_visita();cancelar_visita()]
[SISTEMA|calcular_posibles_rutas();asignación_de_horarios()]
[OFICIAL_VISITANTE|código_de_identificacion|visita_al_solicitante();verificación_del_solicitante()]
[OFICIAL_OPERADOR|código_de_operador|elegir_una_ruta();aprobar_una_solicitud();registro_de_fecha_de_visita()]
[CERTIFICADO|numero;nombre_de_solicitante;fecha_de_emision;periodo_de_validez|recepcion_del_certificado]
[SOLICITUD|numero;nombre_del_solicitante;fecha_de_emision;estado_de_solicitud;posición_geografica;motivo|envio_de_solicitud;aceptación_de_solicitud|rechazo_de_solicitud]
[NOTIFICACION|numero_de_solicitud;mensaje;estado|envio_de_la_notificacion]
[RUTA|direcciones;tiempo_estimado|generacion_de_rutas();eleccion_de_una_ruta()]
[FIRMA|marca|ingreso_de_la_firma()]
[DOMICILIO|direccion;numero_de_solicitante|carga_de_certificado()]
[COMISARIA|numero;direccion;responsable]
```

Figura 3. Código interpretable para yUML.

Como resultado de la utilización de la herramienta yUML, utilizando como entrada, el código expuesto anteriormente, se obtiene el siguiente gráfico:

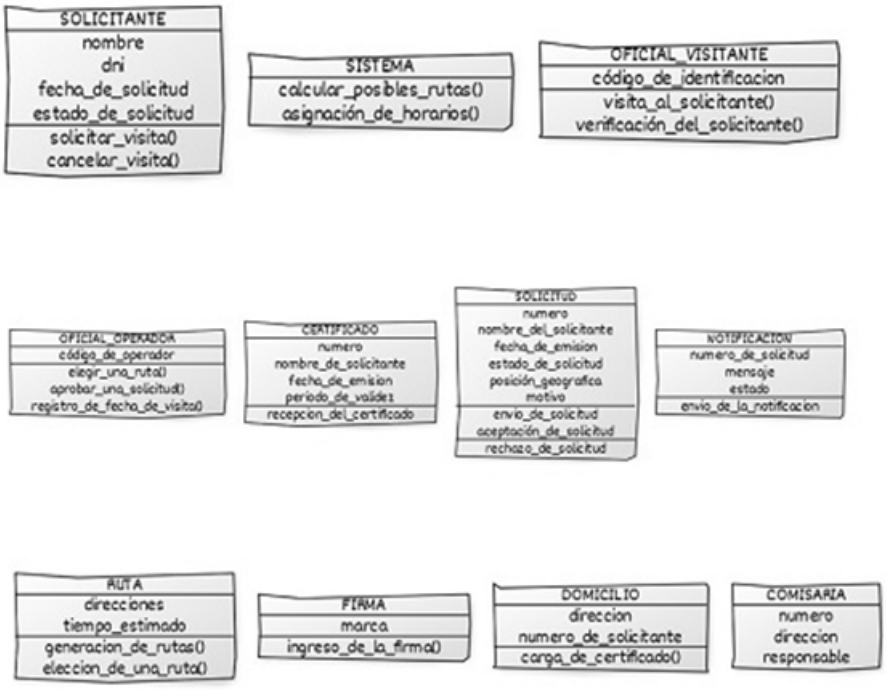

Figura 4. Diagrama de Clase generado por la herramienta yUML.

Otra funcionalidad del compilador traductor gUML es que desde una especificación en LEL se puede generar diagramas de casos de uso, para esto se deberá realizar los mismos pasos anteriores utilizando el compilador traductor gUML para casos de uso, este genera un código intermedio que es interpretado por la herramienta yUML y se obtiene el siguiente diagrama de caso de uso para el ejemplo de solicitud y asignación de certificados de residencia de la policía de la provincia de Jujuy.

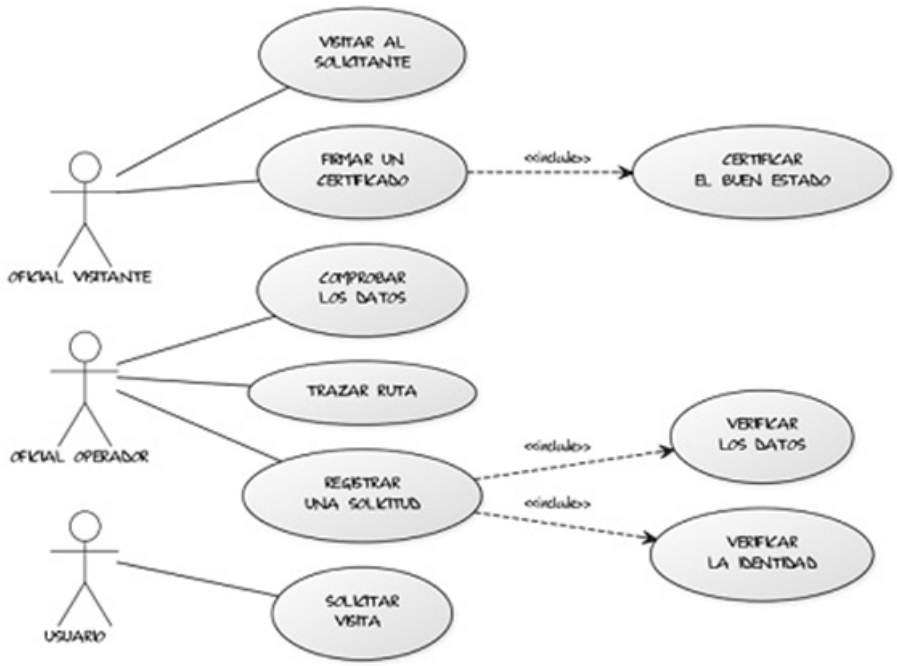

Figura 5. Diagrama de Casos de Uso generado por la herramienta yUML.

Conclusión

De acuerdo con el trabajo realizado, tanto en la parte investigativa como en la parte técnica involucrada en la implementación de la propuesta, fue posible percibir la importancia que tiene en el proceso de desarrollo de software la Ingeniería de Requerimiento y sobretodo la especificación de requerimientos.

Las herramientas como LEL junto con los de diagramas de Casos de Uso y los diagramas de clases son entendidos en general por clientes, usuarios y demás involucrados en el proceso de desarrollo de software.

En particular, LEL, utiliza símbolos que son, en general, las palabras o frases utilizadas por el usuario los cuales identifican el lenguaje de la aplicación por lo que resulta ser un medio de comunicación eficiente al utilizar palabras que le son familiares en su entorno de trabajo; gUML traduce estos símbolos utilizados en LEL y genera como salida un diseño orientado a objetos.

En las cátedras de Administración de Proyectos y Sistemas de Información II se realizaron prácticas de elicitación de requerimientos utilizando LEL en los proyectos de cátedras propuestos por los alumnos. En dichas prácticas se pudo utilizar la nueva versión del compilador traductor gUML, al cual se le incorporó nuevas funcionalidades, que

permitieron obtener un diseño orientado a objetos.

Se concluye que LEL junto con el compilador traductor gUML son de gran valor para los procesos de la Ingeniería en Requerimientos, estas herramientas permiten un claro entendimiento como así también una referencia para modelar requerimientos.

Finalmente, como trabajo futuro se podría considerar su integración con otras herramientas CASE para otorgar una funcionalidad más completa en la gestión de requerimientos y su asociación a los diagramas UML generados a partir de estos; también aplicar estos conceptos a otras herramientas de modelado de requerimientos a partir de los cuales generen otros tipos de diagramas.

Referencias

Aho, A. V., Ullman, J., Lam, M. S.-L., & Sethi, R. (2017). Compiladores: Principios, Técnicas y Herramientas. Segunda Edición. Pearson Educación.

Antonelli, L, Rossi G., Leite, J.C. y Oliveros, O. (2012). Deriving requirements specifications from the application domain language captured by Language Extended Lexicon, Workshop in Requirements Engineering (WER), Buenos Aires.

Antonelli, L., Rossi G., Leite J.C.S y Oliveros A (2013). Buenas prácticas en la especificación del dominio de una aplicación. Montevideo, Uruguay.

Booch, J, Rumbaugh J., Jacobson I (2006). El Lenguaje Unificado de Modelado UML 2.0 2da. Edición. Madrid: Addison Wesley.

Hadad, G., Kaplan G., Oliveros, A. y Leite J., (1996). Integración de Escenarios con el Léxico Extendido del Lenguaje en la elicitación de requerimientos: Aplicación a un caso real, Revista de Informática Teórica e Aplicada.

Harris, T (2018). yUML a pocketworks application: http://yuml.me/diagram/scruffy/usecase/draw,.

Molina, J., García Rubio F., Pelechano V., Vallecilo A., Vara J. y Vicente-Chicote C (2008) Desarrollo de Software Dirigido por Modelos: Conceptos, Métodos y Herramientas, México, Alfaomega.

Pons, C., Giandini, R. y Pérez G. (2008). Desarrollo de Software Dirigido por Modelos Conceptos, UNLP, La Plata.