Analogías Graficas como Método de Aprendizaje en un Curso de Programación Orientada a Objetos

Leoncio Jiménez[1] **Pascale Zaraté**[2] **Elizabeth Vidal**[3]

[1]Departamento de Computación e Informática, Universidad Católica del Maule, Talca – Chile
ljimenez@spock.ucm.cl

[2] IRIT, UMR 5505 CNRS – INPT – ENSIACET – GI, Toulouse – France
Pascale.Zarate@irit.fr

[3] Escuela Profesional de Ingeniería de Sistemas, Universidad Nacional de San Agustín, Arequipa – Perú
e.vidal@usp.edu.pe

RESUMEN

Es complejo entender la fenomenología de un lenguaje de programación orientado a objetos, dado que la descripción de los fenómenos, tales como: objetos, eventos, mensajes, etc., y las relaciones entre ellos, son organizadas y estructuradas en un lenguaje artificial, que en definitiva permitirá construir un *programa* (sistema) que será entendido (compilado) por la máquina. Este lenguaje artificial lo llamamos *lenguaje técnico* (entendible por la máquina), para diferenciarlo del *lenguaje social* (entendible por el hombre). Esos dos lenguajes existen simultáneamente y necesariamente en el espacio dual *hombre/máquina*. Por ello, identificar, distinguir e explicar los conceptos de un lenguaje de programación orientado a objetos es una tarea compleja. Ahora bien, para comprender y explicar un *lenguaje de programación* o *lenguaje técnico* por medio de un *lenguaje social*, en [14] propusimos el uso del concepto de *analogía*. En este artículo proponemos una generalización de ese concepto, en base a los conceptos de *analogías superficiales* y *profundas*.

Palabras Clave: lenguaje de programación orientado a objetos, pensamiento o razonamiento analógico, analogías superficiales, analogías profundas, modelo conceptual, paradigma hombre/máquina.

1. Introducción

Según el *enfoque autopoiético de Valparaíso*[1] [17,18,19], las *empresas* existen simultáneamente y necesariamente en dos dominios[2]. El primero es el **dominio social**, que está constituido de las **relaciones de negocio**, es decir las relaciones simbólicas o culturales en torno a los procesos de negocio, mientras que el segundo es el **dominio físico** formado, por una parte,

[1] Este enfoque postula que la empresa es vista como un sistema cerrado en el seno de un sistema abierto, que auto produce con su operar sus propias relaciones entres sus componentes [17,18,19]. Además, este enfoque complementa (1) el enfoque sistémico, en este caso la empresa es vista como una red de transformaciones de entradas en salidas [4]; y el enfoque cibernético, en este caso la empresa es vista como un conjunto de variables a controlar y regular [20]. En [27] se detallan los postulados del enfoque *autopoiético* aplicado al dominio biológico, y que ha sido desarrollado por Humberto Maturana y Francisco Varela.

[2] Limone y Bastias, dicen: "para nosotros, la empresa, como sistema social existe en dos dominios simultáneamente: el de las personas y objetos (materiales, y energéticos) y el social (lo simbólico)" [17]. En este artículo nosotros hemos adaptado dicha argumentación para identificar los objetos y procesos de negocio.

por las **personas** (objetos de negocio humanos descritos a través de su capital intelectual y emocional) y, por otra parte, la **tecnología** (objetos de negocio materiales y energéticos descritos a través de atributos o propiedades de los mismos). Lo anterior no hay que confundirlo con el *enfoque socio técnico*, dado que el *enfoque autopoiético de Valparaíso* rescata de la componente *social* del *enfoque socio técnico* a las *personas*, para posicionarlas en el dominio físico, dado que en el *enfoque autopoiético de Valparaíso* la organización existe en un espacio dual formado por el dominio social (simbólico o cultural) y el dominio físico (material). Lo que quiere decir que las relaciones simbólicas o culturales de la organización son materializadas en la estructura por las personas y la tecnología. Sin embargo, la dualidad organización/estructura exige la existencia necesaria y simultánea de estos dos dominios: **social** (**procesos de negocio**) y **físico** (**objetos de negocio**) que forman una red de transformaciones, en una palabra la *empresa*.

Ahora bien, si nos centramos en el *dominio físico* y específicamente en la *tecnología*, podemos decir que la *tecnología orientada a objetos*, por la dualidad organización/estructura puede ser utilizada para describir los procesos de negocio, un trabajo en esa dirección puede ser consultado en [26]. Este cambio, no sólo exige la adopción de dicha tecnología en el parque informático de la empresa, sino que constituye una verdadera revolución en el plano simbólico o cultural de la empresa. Por ejemplo, si pensamos incorporar un sistema de información de tipo e-procurement [2] para gestionar la cadena de abastecimientos de una empresa, es decir la red de transformaciones, que va desde el aprovisionamiento hasta la venta, pasando por la contabilidad y el control de inventarios, esto requiere la redefinición de los procesos y objetos de negocio de la realidad en cuestión. Mas grave aún si queremos pasar de una empresa a una e-empresa [28], el cambio va mucho más allá y requiere la transformación total, por una parte, de las relaciones (procesos de negocio) en el dominio social (simbólico o cultural), y por otra parte, de las componentes (objetos de negocio) de esas relaciones en el dominio físico.

Este nuevo paradigma implica, por una parte, la incorporación de *modelos conceptuales* para entender, explicar y comunicar la fenomenología en sus dos dominios: *social* y *físico*, y por otra parte, la posibilidad de contar con informáticos mejores preparados, no sólo para entender los conceptos de un lenguaje de programación orientado a objetos, sino que también para entender la realidad del negocio en términos de objetos [26], lo que puede hacer mucho más interesante la perspectiva de un proceso de software.

La incorporación de tecnología exige la emergencia de herramientas conceptuales capaces de explicar la realidad, en base a modelos. Este nuevo paradigma descriptivo se ha hecho sentir con el tiempo, de donde nace el *pensamiento* o *razonamiento analógico* para explicarlos. En [15] presentamos un *razonamiento analógico* que basa sus raíces en el Diagrama de Flujo de Datos (DFD), que con la experiencia pedagógica de su uso se convirtió en un modelo de aprendizaje, llamado: DFD*. Dicha herramienta se ha mostrado especialmente adecuada y práctica para complementar el *método de casos*, que groso modo intenta reproducir una realidad de negocio, dada a conocer en forma de texto, para que el alumno pueda desarrollar su capacidad de diagnosticar, evaluar y recomendar acciones. En tanto, en [14] presentamos nuestra primera experiencia pedagógica frente al paradigma de la programación orientada a objetos basada en el uso de *analogías*, que ahora hemos generalizado, en *analogías superficiales* y *profundas*.

El resto del artículo está organizado de la siguiente forma: en la Sección 2 se describen los conceptos de analogías superficiales y profundas. La Sección 3 presenta la construcción de una analogía superficial y profunda para explicar el concepto de *objeto*. Mientras, que en la Sección 4 explicamos concepto de evento/mensaje a través de otras analogías superficiales y profundas. En este caso el lenguaje utilizado es SQLWindows [9]. Finalmente, en la Sección 5, se describen las conclusiones y los trabajos futuros.

2. Analogías Superficiales y Profundas

En [14] presentamos la utilidad del concepto de *analogías* para explicar los conceptos de *objeto* y *evento/mensaje* de un lenguaje de programación orientado a objetos[3]. Dicho artículo surgió de la experiencia docente, del primer autor, en el curso de Taller de Software del Departamento de Computación e Informática de la Universidad Católica del Maule. En particular, de un informe elaborado por un alumno del curso.

Una *analogía* es un "puente", entre la actividad humana (lenguaje social entendible por el hombre), y lo artificial (lenguaje técnico entendible por la máquina). En decir una *analogía* es el puente que debemos cruzar para poder comprender los conceptos de un lenguaje de programación. La *analogía* tiene su utilidad, dado que en general los fenómenos que se desean describir de la naturaleza, tienen entre ellos, en ciertos casos, un parecido evidente en sus manifestaciones, sus comportamientos y/o sus funcionamientos. Esto quiere decir, que es común, que la observación de un fenómeno evoque en el espíritu del observador, otros fenómenos que normalmente se encuentran algo alejados del dominio donde el fenómeno estudiado está definido o existe, por lo tanto, existe una cierta imagen de algo, un cierto conocimiento en otro dominio que puede ayudar en la explicación del fenómeno estudiado. Ahora, bien el *pensamiento* o *razonamiento analógico* es justamente eso, es decir el hecho de pasar (o cruzar un puente) de un fenómeno conocido a un fenómeno que se quiere conocer, por medio de algo, ese algo lo llamamos: *analogía*.

Sin embargo, debemos dejar en claro que según la *teoría de las bases biológicas del conocimiento* [27], el cerebro humano (lenguaje social) no funciona en base a una compilación (lenguaje técnico) de eventos y mensajes que gatillan acciones sobre los objetos, pero si nos apoyamos en la *teoría cartesiana*, es posible imaginar que si lo hace (cerebro = computador) [24]. Esto último valida o fortalece, entonces, nuestro "puente".

Ahora bien, según el pensamiento analógico, el pensar o razonar por analogías, quiere decir que el razonamiento o pensamiento se funda sobre los parecidos o las relaciones de un fenómeno con otro [19]. Un ejemplo de ello es el caso de la "manzana de Newton". En efecto, para el físico inglés Isaac Newton (1642-1727) la existencia y la explicación de una fuerza de gravedad pudo ser posible a través del hecho de ver caer una manzana de un árbol. Está *analogía* entre la caída de una manzana y una fuerza que la provoca, gatilló en Newton la aparición del *concepto de gravedad* en el dominio de la física tradicional.

[3] El lenguaje utilizado fue SQLWindows [9], pero la elección de dicho lenguaje no le resta generalidad a nuestra propuesta.

Luego, las *analogías* pueden ser útiles en la construcción de *modelos conceptuales* que permitan describir una realidad objetiva, la fenomenología, tanto desde el punto de vista *sistémico* [4,16] (entrada/transformación/salida), como del punto de vista *analítico* [4,16].

Aquiles Limone en [19] presenta dos tipos de analogías: las *analogías superficiales* y las *analogías profundas*. Esta última será explicada en términos del concepto de *triangulo sistémico* de Jean-Louis Lemoigne[4].

La analogía superficial pretende establecer un simple parecido, intuitivo o espontáneo entre el fenómeno que se describe y el fenómeno que se conoce, mientras que la analogía profunda establece una relación entre la *estructura* (atributos o propiedades de las componentes y sus relaciones en la estructura), la *función* (organización de la estructura) y la *evolución* (comportamiento de la organización) de los fenómenos vistos como sistemas. Esto permite establecer relaciones causa/efecto de los fenómenos estudiados, y la formulación de un modelo conceptual, que para Limone no es otra cosa que una analogía profunda [19]. Bien que Limone dice, citando a Bertalanffy: <<*las analogías superficiales no tienen ningún valor científico*>> [19], es decir ellas no son una contribución en la formulación de modelos conceptuales, nosotros pensamos, y coincidimos con lo dicho por Limone [19], que el argumento de Bertalanffy es un poco fuerte. Prueba de ello, es el caso de la "manzana de Newton". En efecto, la caída de una manzana de un árbol, fue lo que gatilló en Newton el concepto de gravedad, que más tarde se convertiría en una bella ecuación, es decir en un *modelo conceptual*.

Esto quiere decir, que un *modelo conceptual* se construye progresivamente a través el paso de las analogías superficiales en analogías profundas. Lo que implica que un modelo conceptual constituye la formalización de una analogía profunda. Al respecto, Limone dice: <<*un modelo es la formalización explicita de una analogía profunda, formalización que puede ser matemática o no*>> [19].

3. Analogías superficiales y profundas para explicar el concepto objeto

La programación orientada a objetos constituye un avance significativo para el desarrollo moderno de la ingeniería de software. Ella encuentra sus orígenes en la inteligencia artificial, mientras que sus raíces se encuentran definidas por James Rumbaugh, Graddy Booch y Ivan Jacobson, en los modelos OMT-2 [23], OOD [5] y OOSE [11], respectivamente. Sin dejar de mencionar el hecho que esos tres autores y otros, en el año 96, propusieron el modelo conceptual UML (Unified Modeling Language) [7,8,10,11], que hoy se perfila como una verdadera *norma* para la construcción de sistemas de información [21]. Lo que significa que la enseñanza de la programación orienta a objetos debe estar presente en toda carrera del campo de la informática y de la computación.

El concepto central en la programación orientada a objetos es el concepto de **objeto** [3,9], que a lo largo del uso de los diferentes modelos, antes citados y otros, ha recibido variados significados, para nuestros propósitos un *objeto* es un elemento identificable de la

[4] Aquí encontramos cómodo usar el concepto de *triangulo sistémico* de Jean-Louis Lemoigne [16] para definir una analogía profunda. En este caso lo que se busca en tener dos fenómenos isomorfos que sean descritos por el mismo triangulo sistémico. Es bueno decir también que la descripción usada por Limone se basa en el estudio de la fenomenología de Bertalanffy, que en groso modo coincide con el triangulo sistémico de Jean-Louis Lemoigne [16], que explicamos más abajo.

realidad objetiva, este objeto puede ser concreto (se puede distinguir de otros objetos) o puede ser abstracto (su creación o disparición es independiente del observador). Un *objeto* es caracterizado por una **estructura** (sus datos) y unos **servicios** (sus operaciones o métodos) [3,9]. La estructura del objeto corresponde a los **atributos** y **valores** de los datos del objeto, mientras que los servicios corresponden a las **funciones** o formas de actuar y responder del objeto según un cierto **comportamiento** del mismo [3,9]. Los objetos existen en cuanto a *instancias* (miembros) de una *clase* definidos sobre un *dominio*. Los objetos colaboran por intercambio de *mensajes*, cada mensaje corresponde a una operación del objeto destinatario [3,9]. En definitiva, un *objeto* es algo que puede ser identificado en un cierto dominio, según tres puntos de vista: (1) lo que el objeto sabe hacer (**estructura**); (2) lo que el objeto puede hacer (**servicio**); y (3) cuando el objeto debe hacer lo que sabe hacer (**comportamiento**). Ahora, para que un objeto haga algo se le envía un *mensaje*. Para nosotros, estos tres puntos de vista están relacionados con los conceptos de *triangulo sistémico*[5] de Jean-Louis Lemoigne [16], y *operación de distinción*[6] de Aquiles Limone [19], como lo veremos en la construcción de *analogías superficiales* y *profundas.*

En la construcción de las *analogías superficiales* hemos tomado prestado las figuras 1, 2 y 3 del libro de Fannader et Leroux, llamado: *UML Principios de Modelización* [8]. Estas tres figuras nos permiten construir nuestras *analogías superficiales*, que serán la base de una *analogía profunda* para explicar el concepto de *objeto*. La descripción entonces de un *objeto*, según el *triangulo sistémico* (modificado por nosotros), es en base a tres puntos de vista. El primer punto de vista (**la estructura**) describe **lo que el objeto sabe hacer**, en términos de componentes y atributos. La operación de distinción es este caso es la **definición de la estructura del objeto**. El segundo punto de vista (**la organización**) describe **lo que el objeto puede hacer**, en términos de funciones o servicios. La operación de distinción es este caso es **la relación de la organización de la estructura de los diferentes servicios de los objetos**. El tercer punto de vista (**el entorno**) describe **cuando el objeto (estructura) debe hacer lo que sabe hacer (servicio)**. La operación de distinción en este caso es el **dominio de existencia del entorno** que hace posible la existencia de las relaciones entre los objetos y sus atributos/funciones, es decir el comportamiento del objeto es descrito en términos de su entorno, y no en términos de atributos/funciones.

Un *objeto* queda definido entonces como una triada dada por: (1) la **definición** del objeto; (2) la **relación** del objeto con otros objetos; y (3) el **dominio** de existencia de los objetos y sus relaciones (ver figuras 1, 2 y 3). Esta triada es la estructura de una *analogía profunda*, que se compone de las siguientes *analogías superficiales*:

La analogía superficial de la figura 1 describe el *objeto* a través de su **definición**. En efecto, una "bicicleta", una "máquina distribuidora de bebidas", una "automóvil", un "cliente", una "máquina distribuidora de bencina" tienen sus propios atributos que los distinguen.

[5] Para efectos de nuestros propósitos, nosotros hemos llamado a esos tres puntos de vista "estructura", "organización" y "entorno", para hacer aparecer la dualidad organización/estructura. Los puntos de vistas originales son: "estructura", "función", "evolución". El primer punto de vista describe las componentes del sistema, dado sus atributos, el segundo punto de vista describe las relaciones de las componente para ofrecer una función o servicio, el tercer punto de vista describe la evolución de las relaciones en entre componentes para ofrecer los servicios, es decir la evolución o continuidad del sistema. Un estudio detallado de del triangulo sistémico de Lemoigne puede ser consultado en [12].

[6] La operación de distinción es otra herramienta sistémica para distinguir la dualidad organización/estructura de un sistema organizado [19].

La analogía superficial de la figura 2 describe el *objeto* a través de su **relación** con los otros objetos. Una primera relación podría estar dada por un servicio de *transportar*, en este caso, los objetos "bicicleta" y "automóvil" se encuentran relacionados, mientras que una segunda relación, que podría estar dada por el servicio de *distribuir*, los objetos relacionados son "máquina distribuidora de bebidas" y "máquina distribuidora de bencina", mientras que el objeto "cliente" no pertenece a ninguna de estas dos relaciones distinguidas (transportar o distribuir).

La analogía superficial de la figura 3 describe el *objeto* a través del **dominio**. Por ejemplo, el dominio "bomba de bencina" valida la existencia de transacciones validas entre atributos/funciones de los objetos. En este caso el servicio *distribuir* puede interpretarse como la distribución de energía. Es evidente que la energía que necesita un automóvil y una persona para realizar sus funciones, son distintitas. Por lo tanto, decir que el automóvil requiere de la energía que proviene de una distribuidora de bencina, y no de la energía que proviene de una máquina distribuidora de bebida, tiene sentido, y con ello el dominio existencial de los objetos que participan a las relaciones. Estas tres *analogías superficiales* las hemos llamado como su operación de distinción: *definición*, *relación* y *dominio*, y la unión de ellas forma una analogía profunda.

Figura 1: Definición

Figura 2: Relación

Figura 3: Dominio

Pero también en [8] encontramos otras *analogías superficiales* que son utilizadas para ilustrar otros conceptos de la programación orienta a objetos, tales como: *clase*, *herencia*, *encapsulación*, *polimorfismo*, etc. En particular, en [8] se define el polimorfismo como el hecho que un mismo mensaje puede tener tantas interpretaciones como tenga de objetos destinatarios. La figura 4 muestra la *analogía superficial* utilizada para ilustrar ese concepto.

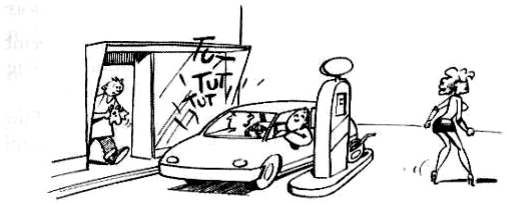

Figura 4: Polimorfismo

Es claro que todas estas *analogías superficiales* pueden tener un gran valor pedagógico para relacionar los conceptos de la programación orientada a objetos, con fenómenos de la vida

real, ya que esto permite estimular la activad creativa, y eso facilita la comprensión y la manipulación de todos esos conceptos. Sin embargo, en la programación orientada a objetos, no solamente están los *objetos* sino que también está la *programación*, es decir el concepto *evento/mensaje*, que discutiremos a continuación.

4. Analogías superficiales y profundas para explicar el concepto evento/mensaje

En esta sección describimos dos analogías superficiales para explicar el concepto *evento/mensaje*. La primera tiene relación con la significación de los objetos a través de analogías (de definición, de relación y de dominio), mientras que la segunda, tiene relación con la significación de la forma en que colaboran esos objetos. Estas analogías nos permitirán construir una *analogía profunda* dada por un *modelo conceptual*. Para explicamos el concepto *evento/mesaje*.

Un **evento** corresponde a la acción determinada por el usuario para realizar una determinada operación [9], por ejemplo, en el *formulario* (interfaz Paciente) de la figura 6 el usuario puede mover el *ratón* para pinchar el *botón* (Ingresar), y como su nombre lo dice, ingresar los datos del paciente en una tabla de la base de datos que lleva ese mismo nombre (ver figura 6). Lo importante es hacer notar, que es el usuario quien determina los eventos, pero también puede ocurrir que sea el sistema operativo que los gatille. El **mensaje** se refiere a la forma de comunicar que tienen los objetos [9], por ejemplo en el *formulario* de la misma figura, puede existir perfectamente una comunicación entre los dos *botones*, o entre un *botón* y un *campo de datos* (Rut Paciente), o entre el *formulario* y uno de los *radio botones* (Femenino, Maculino), etc., para ello es necesario especificar en el código de la aplicación: primero quién envía los mensajes; segundo dónde son enviados los mensajes; y tercero cómo son capturados los mensajes.

En la figura 5, se muestra el código SQLWindows que corresponde al objeto pushbutton (Ingresar). La sección *Message actions* de la misma figura, cumple dos objetivos: (1) permite atrapar y enviar mensajes a otros objetos, y (2) contiene la lógica del botón, es decir sus distintas operaciones o métodos. En efecto, en un entorno de programación SQLWindows, el evento pinchar el *botón* (Ingresar) tiene asociado el mensaje *Sam_Click*, que es atrapado en la *Message actions*[7] del objeto *Pushbutton* (Ingresar) con la sentencia *On*. La acción a realizar es el poblado de la tabla (Pacientes, ver figura 6), por medio del llamado a la función *SalTblPopulate* con la sentencia *Call*.

- *Pushbutton: Ingresar*
- *Message actions*
  - *On Sam_Click*
    - *Call SalTblPopulate(hWndForm,hSql, :colrut, :colsexo,TBL_FillNormal)*[8]

Figura 5: Concepto *evento/mensaje* del objeto *pushbutton* (o botón)

[7] La *message action* corresponde al "método" del objeto, es decir es la sección del objeto donde se describen las operaciones (servicios) que puede hacer el objeto. En programación orienta a objetos, cada objeto está compuesto, por una parte, por sus *datos*, y por otra parte, las *operaciones* o *métodos* que pueden manipular esos datos [9].

[8] Los parámetros de la función, corresponden al nombre del formulario, al nombre del query Insert que poblara la tabla, con los campos de datos especificados.

Mientras que la figura 6 muestra el esquema Cliente-Servidor que corresponde al ejemplo de la figura 5, y que nosotros implementamos en SQLWindows. Los datos del paciente (Rut y Sexo) son ingresados, por el usuario, en el formulario Paciente y luego ingresados a la tabla del mismo nombre.

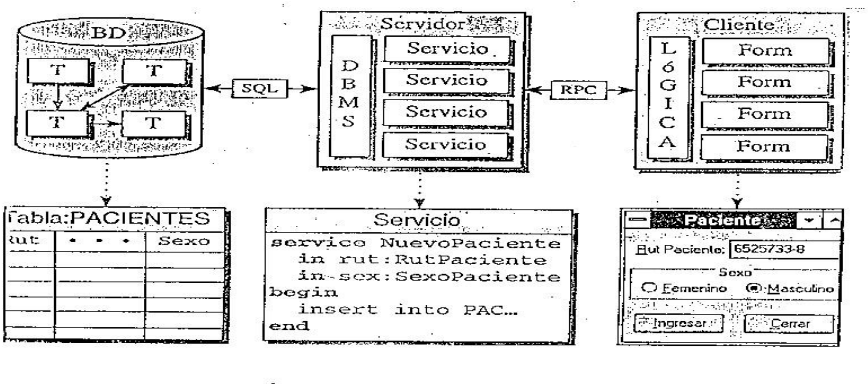

Figura 6: Esquema cliente/servidor. Fuente: Empresa Orden

Para describir el concepto *evento/mensaje* de la figura 5, y sin restarle generalidad a otro lenguaje orientado a objetos que se utilice, que en este caso ha sido SQLWindows, usamos una *analogía* que fue presentada en [14], y que fue elaborado por coautor de ese artículo, y que nosotros la hemos adoptado como una *analogía superficial*. Para ello imaginémonos un escenario (dominio) formado por un sistema organizacional, en que se realicen diversas actividades humanas, dicho sistema puede ser organizado en base a competencias o conocimientos que requieran las diferentes tareas realizadas por el sistema. Para Uribe: <<*estas tareas son realizadas por humanos, tales como: médicos, carpinteros, músicos, profesores, electricistas, abogados, agricultores, etc.*>> [14].

Luego, Uribe dice: <<*no es difícil darse cuenta que en dicha organización un médico no podrá reparar una falla eléctrica, como de igual forma, un abogado no sabrá en que fecha se plantan los tomates, ni tan poco de su cuidado, menos del abono que hay que ocupar para que los tomates crezcan sanos y bonitos*>> [14]. Lo anterior nos indica que cada *tarea* tiene asociado un *actor* que posee un cierto *conocimiento* para ejecutar la tarea según un *objetivo*. De igual forma, en dicha *analogía superficial*, existe la idea de *especialización* y *generalización* de la tarea.

La primera *analogía superficia*l que nos interesa construir nos permitirá ilustrar el concepto de *objeto* (dominio técnico) en términos de la *definición*, *relación* y *dominio* (ver figuras 1,2 y 3, más arriba). Supongamos entonces, que todo *actor* (dominio social) de nuestro sistema puede ser conceptualizado como un *objeto*. Eso significa que los *objetos* (dominio técnico) deben heredar atributos de los actores (dominio social). Luego, según Uribe: <<*podemos decir que los objetos están vivos, ya que son seres humanos*>> [14], lo que significa que poseen características propias (analogía de definición), por ejemplo, Uribe dice: <<*una profesión que les permita hacer algo*>> [14], por lo tanto, estos objetos tienen el poder o la facultad de hacer trabajos o realizar funciones diferentes (analogía de relación). En términos

genéricos cada objeto tiene definido en su estructura un conjunto de operaciones que puede realizar, según un cierto escenario (analogía de dominio), y esas acciones son las funciones propias del objeto. Esto significa, que la estructura del objeto puede ser definida por *especialización* o *generalización* de otra estructura. Lo que implica también, que los objetos deben tener la capacidad de responder de cierta forma y no de otra, ante un estimulo determinado por el entorno o dominio para realizar una tarea.

Por ejemplo, para Uribe: <<*en un escenario hospitalario, el objeto carpintero no puede realizar la función de cortar una pierna a un paciente, sin embargo, en un escenario de un taller de carpintería, lo que ese objeto puede hacer es cortarle una pata a una silla*>> [14]. Por lo tanto, la *estructura* y la *organización* del objeto están definidos por "el mismo", es decir por su propio conocimiento, y no por su *entorno*. Esto último, justifica la modificación que nosotros hemos hecho del concepto de *triangulo sistémico* de Jean-Louis Lemoigne [16]. En efecto, para caracterizar una *analogía profunda* nosotros lo hacemos según tres puntos de vista: "**estructura** (definición)", "**organización** (relación)" y "**entorno** (dominio)", que a su vez justifica la hipótesis que sostienen el carácter autopoiético de los sistemas de información [1], es decir, los sistemas de información son vistos como un sistema cerrado en el seno de un sistema abierto, que autoprocude con su operar sus propias relaciones entre sus componentes [1], y que es equivalente a lo dicho en nuestra introducción. La figura 7 muestra algunos ejemplos de la dualidad estructura/organización del sistema organizacional que Uribe identificó en [14]. Es bueno notar, que está dualidad se encuentra embebida en el código SQLWindows de un objeto. En efecto, la sección *Message actions* corresponde a la **organización** (**lo que el objeto puede hacer**) queda definida en el código del objeto, mientras que la **estructura** (**lo que el objeto sabe hacer**) queda definida por el nombre del objeto. En efecto, si vemos la figura 5, la estructura del objeto queda definida por su nombre (*Pushbutton*: Ingresar), mientras que su organización queda definida por el código SQLWindows de la *Message actions*. En otras palabras, el objeto "botón" tiene una estructura que lo diferencia de otros objetos, y una organización que define lo que el ese "botón" en particular puede hacer, lo que significa que el "botón" pertenece a una sola *clase*, pudiendo ser diferenciados varios botones por lo que hacen.

Estructura	Organización
Medico	Operar, dar primeros auxilios, mandar a hacer exámenes, diagnosticar, recetar medicamentos, etc.
Carpintero	Reparar casas, construir casas, reparar puertas o ventanas, techar, arreglar muebles, etc.
Electricista	Arreglar un cortocircuito, instalar luminaria, reparar motores eléctricos, etc.
Agricultor	Arar la tierra, regar, cosechar, preparar la tierra, etc.

Figura 7: Ejemplos de la dualidad organización/estructura de un objeto dado un sistema

La segunda analogía superficial, que nos interesa construir nos permitirá ilustrar el concepto de *mensaje* (dominio técnico), es decir tiene que ver con la forma de comunicación de los objetos. Ahora bien, si queremos que cada objeto realice una función de acuerdo a lo que el objeto sabe hacer (**estructura**), y como ya sabemos que cada objeto tiene funciones propias de acuerdo a lo que el objeto puede hacer (**organización**), es decir, funciones diferentes para cada uno de ellos, debemos crear necesariamente un medio (**entorno**) de comunicación por donde son enviados los mensajes cuando se produzca un evento. Al respecto, Uribe dice: <<*imaginemos entonces que el medio, es decir, nuestro entorno, sea un supermercado, donde se anuncia por el altoparlante que una persona necesita primeros auxilios*>> [14], entonces según la figura 7, hay un solo objeto que responderá a ese llamado, dado que en su **estructura**

tiene definidos los atributos que lo definen como un "medico", y en su **organización** están definidas las funciones que puede hacer, una de ellas es justamente la de dar primeros auxilios (ver figura 7). Por lo tanto, solo será ese objeto y no otro que capture ese mensaje y realice la acción (en este caso, dar los primeros auxilios). Ahora bien, Uribe agrega: <<*que pasaría si en el supermercado hubiera una enfermera, entonces los dos acudirían al llamado para dar los primeros auxilios*>> [14]. Esto explica que en la dualidad organización/estructura los objetos se comunican entre si a través de mensajes para realizar una acción adecuada, que es justamente la forma en que los objetos en SQLWindows se comunican, para ello se utiliza la sentencia *On* que permite atrapar los mensajes y la sentencia *Call* para realizar una acción, en este caso llamar a una función, como se puede ver en la *Message actions* del objeto de la figura 5.

De las dos analogías superficiales anteriores, podemos concluir que es el *evento* (**entorno**) quien gatilla los mensajes de respuesta de acción de los *objetos*, y no la dualidad organización/estructura.

Finalmente, nos resta construir la *analogía profunda* en términos de un *modelo conceptual*, dado que anteriormente dijimos que el *modelo conceptual* es la formalización explicita de una *analogía profunda*. En nuestro caso de estudio, esto quiere decir, que es la *analogía profunda* quien genera el significado del *modelo conceptual* que permita a su vez interpretar un lenguaje en particular. La figura 8 muestra: (1) las *analogías profundas* distinguidas de las dos *analogías superficiales* discutidas anteriormente; (2) su representación en un modelo conceptual; y (3) su interpretación en SQLWindows.

Analogía Profunda	Modelo Conceptual	SQLWindows
<humano–conocimiento–tarea>	*<objeto–función–tarea>*	*<Pushbutton–Messageactions–SalTblPopulate>*
<evento–humano–conocimiento–tarea>	*<mensaje–objeto–función–tarea>*	*<Sam_Click–Pushbutton–Messageactions–SalTblPopulate>*

Figura 8: Analogía profunda/Modelo conceptual

En la figura 8, la sentencia Sam_Click tiene una significación de evento (analogía profunda) y de mensaje (modelo conceptual), lo que corresponde bien a su significación dada por el concepto *evento/mensaje* en SQLWindows (ver figura 5).

5. Conclusiones y trabajo futuro

En este artículo se ha presentado la construcción y usos de *analogías superficiales* y *profundas*, para ilustrar ciertos paradigmas de la programación orientada a objetos, en particular los conceptos de *objeto* y *evento/mensaje*. Esto puede ser de gran utilidad como un modelo de aprendizaje que facilite el entendimiento de dicha disciplina. No solamente con software propietario, que es el caso de SQLWindows, pero también con software libre, por ejemplo Python.

La programación orientada a objetos ha sido ampliamente enseñada en el campo de la ingeniería de software, pero también en otras áreas como lo son las redes de petri a objetos [6],

y el modelado de objetos y procesos de negocio [26]. En todos los casos su base conceptual se apoya en el hecho que el mundo es percibido en términos de relaciones entre objetos. Lo que requiere el uso de *analogías superficiales* y *profundas* para comprender sus diferentes conceptos asociados. Este artículo ha contribuido en dicha labor, de igual forma se ha ampliado nuestro conocimiento adquirido en [14,15].

Como trabajo futuro, se contempla aplicar el uso *analogías superficiales* y *profundas* en dos campos principalmente:

El primero tiene relación con el razonamiento por contradicciones, específicamente con el método TRIZ [12,13] aplicado a los procesos de software, vistos como un conjunto de relaciones contradictorias de especificación de requerimientos. En [25] es discutido el uso de TRIZ en la ingeniería de software.

El segundo tiene relación con el leguaje UML (Unified Modeling Language) [7,8,10,11]. En efecto, la motivación de este lenguaje es proveer una semántica formal de manera natural e intuitiva para la construcción de modelos (diagramas en términos UML). Estos modelos son construidos sistémicamente, lo que permite entender y explicar la realidad que se intenta modelar en base a tres puntos de vistas según el concepto de *triangulo sistémico* de Jean-Louis Lemoigne [16]. El primer punto de vista, intenta distinguir los *objetos* de la realidad. El segundo punto de vista, intenta distinguir las *funciones* de la realidad, es decir lo que es posible hacer con esos objetos. El tercer punto de vista, intenta distinguir el *comportamiento* de esos objetos y funciones, es decir cuando es posible hacer las diferentes operaciones con los objetos, o cuando los objetos prestan sus servicios. Ahora bien, para integrar, sistémicamente esos tres puntos de vista (objetos, funciones y comportamiento), existen nueve tipos de diagramas en UML para modelar los aspectos *funcionales* (según esos tres puntos de vista: "objetos", "funciones" "comportamiento"), y *físicos* del sistema. Así, los requerimientos funcionales del sistema son capturados por los diagramas de *casos de uso* Para la realización de estos casos de uso, debemos proveer, por una parte, estructuras estáticas del modelo que están dadas por los diagramas de *clases* y los diagramas de *objetos* (que son, bien entendido, las instancias del diagrama de clases), y por otra parte, estructuras dinámicas del modelo que están dadas por los diagramas de *secuencia*, de *colaboración*, de *estado*, y de *actividad*. En cambio los aspectos físicos (programación, plataforma) del sistema son modelados a través de los diagramas de *componentes* y de *despliegue*. Lo interesante de UML es que todas sus herramientas son graficas, luego es un campo fértil para la construcción de *analogías profundas* y *superficiales*.

Referencias

[1] Abou-Zeid E., *Towards an Autopoietic Approach for Information Systems Development*. Information Modeling in the New Millennium. Idea Group 2001, pp. 34-52, ISBN 1-878289-77-2

[2] Avendaño L., *Un FrameWork para la Planificación de una Cadena de E-procurement en Empresa*, Memoria
Ingeniero de Ejecución en Computación e Informática, Universidad Católica del Maule, Talca Chile, 2004.

[3] Ayache M., Flory A., *Approche Orientée Objet*, Económica, 1996.
[4] Bertanlanffy L. V., *Teoría General de Sistemas*. Fondo de cultura económica, 1976. ISBN 9-68-160627-2.
[5] Booch G., *Object-Oriented Design with Applications*, Redwood city (CA), Benjamin Cummings, 1991.
[6] David R., Alla H., *Du Grafcet aux réseaux de Petri*, Hermes, Paris, 1997.
[7] Fannader R., Leroux H., *UML Principes de mise en oeuvre*, Dunod, Paris, 2000.
[8] Fannader R., Leroux H., *UML Principes de modélisation*, Dunod, Paris, 1999.
[9] Gietz W., *SQLWindows Programming*, Gupta Corporation, 1993. Versión en francés: *Gupta SQLWindows Techniques de programmation client-serveur*, Armand Colin, Paris, 1993.
[10] Jacobson I., Booch G., Rumbaugh J., *The Unified Software Development Process*, Addison-Wesley, 1999.
[11] Jacobson I., Christerson M., Jonsson P., Overgaard G., *Object-Oriented Software Engineering: a use-case driven approach*, Addison-Wesley, 1992.
[12] Jiménez L., *Contribution à la mise en application de la méthode TRIZ, pour l'aide au développement des décisions liées aux processus innovants*, Institut National Polytechnique de Toulouse, Mémoire DEA, Toulouse, France, 2000.
[13] Jiménez L., *The Altshuller's Contradiction Matrix in the Knowledge Creation of Innovation. Case Study: The Honda City*, TRIZ Journal, November 2000. http://www.triz-journal.com/archives/2000/11/b/index.htm
[14] Jiménez L., Uribe E., Experiencias metodológicas para enfrentar el curso de Taller de Software. Un caso de Estudio, en: *XVII Congreso de Educación en Ingeniería*, 8 al 10 de Octubre de 2003, Universidad Católica del Norte, Antofagasta, Chile.
[15] Jiménez L., Urrutia A., Kendall P., Ibáñez J., Propuesta Metodológica para Analizar Casos como Modelo de Aprendizaje, en: *IV Simposio Nacional de Análisis Organizacional, II del Cono Sur*, Buenos Aires, Argentina, 11 al 12 de Agosto de 1999. Publicado en *La Revista Electrónica del DIIC*, edición 4, 1999. ISSN 0717-4195. http://www.inf.udec.cl/revista/
[16] Lemoigne J.-L., *La modélisation des systèmes complexes*, Dunod, 1990. ISBN 2-04-019704-4.
[17] Limone A., Bastias L., La Empresa Como Sistema Autopoiético, en: *II Conferencia Iberoamericana en Sistemas, Cibernética e Informática (CISCI 2002)*, julio 2002, Orlando, EE.UU.
[18] Limone A., *La Empresa: Una Red de Transformaciones*, Editorial Jurídica, Valparaíso, 1999.
[19] Limone A., *L'autopoïèse dans les organisations*. Thèse, Université Paris IX Dauphine, Paris, France, 1977.
[20] Mélèse J. *L'analyse modulaire des systèmes (AMS)*, Les éditions d'organisation, 1991. ISBN 2-7081-1347-X.
[21] Morley C., Hugues B., Leblanc B., *UML pour l'analyse d'un système d'information*, Dunod, Paris, 2000.
[22] Roques P., Vallée F., *UML en action*, Eyrolles, 2000.
[23] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W., *Object-Oriented Modeling and Design*, Englewood Cliffs (NJ), Prentice-Hall, 1991.
[24] Simon A.-A., *Sciences des systèmes Sciences de l'artificiel*, Dunod, 1991, ISBN 2-04-019815-6
[25] Stanbrook T., TRIZ for Software Improvement, in: *26th International Computer Software and Applications Conference (COMPSAC 2002)*, 26-29 August 2002, Oxford,

England, Proceedings. IEEE Computer Society 2002, pp. 466-468, ISBN 0-7695-1727-7
[26] Taylor D., *Business Engineering with Object Technology*, John Wiley & Sons, Inc., 1995.
[27] Varela F., *Autonomie et connaissance*, Seuil, 1989. ISBN 2-02-010030-4
[28] Vidal E., Jiménez L., Metodología para cambiar de Empresa a e-Empresa, en: *II Simposio Internacional de Sistemas de Información e Ingeniería de Software en la Sociedad del Conocimiento, SISOFT2003*, 20 al 23 de Agosto de 2003, Pontificia Universidad Católica de Perú, Lima, Perú.