Abstracción y Conceptualización: Un Enfoque Cognitivo

Manuel Imaz, PhD

BlendMind
Madrid. España
imaz@mac.com

Resumen En este artículo mostramos que, al igual que el concepto de abstracción en su doble aspecto de abstracción como simplificación y abstracción como categorización, existe otro concepto –el de conceptualización– que tambien desempeña un rol importante en ingeniería de software. Existe una bibliografía extensa sobre el concepto de abstracción que implica un mecanismo de abajo hacia arriba (bottom-up) mediante el cual vamos simplificando los fenómenos o creando nuevas categorías con propiedades comunes de todas las instancias que se toman en cuenta. La conceptualización, por su lado, apunta a un mecanismo de arriba hacia abajo (top-down) que presenta la concretización de elementos que permitan identificar algo que todavía no existe y que pretendemos enriquecer a partir de un trabajo de análisis que es posible realizar sobre la nueva idea generada.

Palabras clave: Abstracción. Categorización. Conceptualización.

Abstract. In this paper we will show, as it happens with the concept of abstraction meaning both abstraction as simplification and abstraction as categorization, that there exists another concept, conceptualization, that also plays an important role in software engineering. There is a large bibliography about the concept of abstraction which implies a bottom-up mechanism whereby we simplify phenomena or create new categories with common properties of instances we take into account. Conceptualization, on the other hand, shows a top-down mechanism which presents concretization of elements that allow to identify something that does not exist yet and we intend to enrich by means of an analysis work that it is possible to apply to the new generated idea.

Key words: Abstraction, Categorization, Conceptualization.

1. Introducción

En un artículo anterior [6], planteábamos la cuestión de hasta qué punto podíamos aplicar la denominación de ingeniería a la disciplina conocida como Ingeniería de Software. Alli hemos expuesto varios argumentos que cuestionan la validez del enfoque ingenieril de la ingeniería de software y propugnábamos –como una forma de cubrir las carencias de dicho enfoque ingenieril– una mayor inclusión de ciencias humanas dentro de esta disciplina. Es por eso que aquí presentaremos un análisis de tipo cognitivo para mostrar que ciertos conceptos esenciales de la informática, como es el de abstracción, al igual que otro que proponemos denominar como conceptualización, pueden tener una referencia en las ciencias cognitivas en las cuales poseen, además, un alcance más amplio que en la ingeniería de software y que por esa razón merecen un análisis detallado que puede inspirar aplicaciones a esta última. El concepto de abstracción ha sido presentado como un concepto básico de la Ciencia de la Computación al mismo tiempo que últimamente se han diferenciado dos aspectos especialmente pertinentes del mismo [9]: por un lado, el de eliminar detalles a efectos de simplificar y centrar la atención en lo importante y, por el otro, el proceso de generalización para identificar el núcleo común o esencia de un conjunto de entidades. El concepto de conceptualización no ha sido planteado en informática –al menos hasta donde llegan nuestras investigaciones– de la forma que vamos a presentar en este artículo, aunque hay alguna propuesta llamada abstracciones concretas [3] que es un intento en esa dirección pero diferente a la nuestra.

Queremos aclarar, de entrada, que el concepto de *conceptualización* que presentamos en este artículo difiere del que habitualmente se utiliza en relación con las ontologías, tal como podemos observar en el título de un libro reciente en el cual se articulan la Ontología, la Conceptualización y la Epistemología para los sistemas de información [18].

Es evidente que la palabra "conceptualización" conlleva sus dificultades, porque está muy arraigada y además está asociada a la forma en que conocemos el mundo. Tom Gruber señala en su clásico artículo ¿Qué es una Ontología? [1] que la palabra "ontología" es controvertida porque, mientras que en filosofía se la utilizó para hacer referencia a lo *existente*, en el contexto de las ciencias de la computación y de la información una ontología es "una especificación de una conceptualización". Vemos entonces que hay un primer desplazamiento desde la ontología tal como la entiende la filosofía –centrada en lo existente– hacia la epistemología –lo que conocemos– al plantear la cuestión de una 'conceptualización', aunque la cuestión de cómo efectuamos esa conceptualización no es algo que concierne a esta concepción informática de la ontología. Al final, entonces, esa conceptualización debe especificarse –formalizarse– a efectos de posibilitar que el conocimiento sea compartido y reutilizado y esa especificación sí es el cometido de la ontología tal como la usamos en informática. En un artículo más reciente, Gruber nos plantea que "una ontología define un conjunto de primitivas representacionales con las cuales podemos modelar un dominio de conocimiento o discurso" [2].

Nuestra intención en este trabajo se aparta también de una epistemología (al menos en su sentido tradicional, porque una epistemología debería abarcar el conocimiento de lo existente y de lo que pensamos crear como nuevo), ya que no nos planteamos la forma en que conceptualizamos lo existente, es decir los diferentes dominios de la realidad. Lo que nos ocupa es cómo podemos conceptualizar lo que todavía no existe, lo nuevo, que es el trabajo habitual del diseño.

2. Semántica Cognitiva

El enfoque tradicional objetivista implica una concepción del conocimiento por el cual éste consiste en un conjunto de símbolos que se aplican a un determinado dominio del mundo, las reglas mediante las cuales se manipulan y el conjunto de correspondencias que otorgan un significado a dichos símbolos. Tal como señala Lakoff [10], de acuerdo a los objetivistas, el conocimiento consiste en conceptualizar correctamente y categorizar las cosas del mundo y en captar las conexiones objetivas entre esas cosas y las categorías. Cuando consideramos un signo (una palabra u oración) como compuesto por un significante, un significado y un referente, existen algunos ejemplos que son pertinentes (tal como la famosa distinción entre la estrella matinal y la estrella vespertina planteada por Frege como dos expresiones distintas que tienen el mismo referente, el mismo planeta Venus) pero se excluyen otros, como cuando intentamos encontrar una condición de verdad necesaria compartida entre el 'ver' que encontramos en 'veo el perro en la silla' y el que tenemos en 'veo lo que quiere decir'. El ejemplo previo muestra que incluso algunos conceptos de la vida cotidiana tal como el de ver –que refleja estructuras cognitivas comunes que dependen de nuestras interacciones sensoriales con la realidad– pueden resultar muy complejos y mostrar diferentes significados. Nuestra experiencia como ingenieros de software puede mostrarnos que, cuando diseñamos diagramas de entidad o de clase, el problema de definir conceptos no resulta tan sencillo como podríamos suponer. Una de las dificultades de la teoría clásica de conceptos es lo que Margolis y Laurence denominan el problema de Platón: el problema más básico que se ha planteado contra la Teoría Clásica es que, para la mayoría de los conceptos, simplemente no hay ninguna definición [14]. A esta altura alguien podría argumentar que ésta es justamente la finalidad de las ontologías, tal como se entienden en ingeniería de software: intentar dar definiciones más o menos precisas a las palabras que no se han definido previamente. La mayoría de los actuales trabajos sobre esta disciplina está basada en la teoría clásica de categorías, que durante siglos ni siquiera ha sido concebida como una teoría. No nos sorprende, entonces, que como una posición filosófica asumida, otro filósofo, Wittgenstein, haya podido cuestionar la teoría clásica de categorías. Él explica su posición mostrando el ejemplo de la categoría de 'juego': "Consideremos, por ejemplo, los procesos que denominamos 'juegos'. Me refiero a los juegos de tablero, de cartas, de pelota, olímpicos, etc. ¿Qué hay de común en todos ellos? No me digan: Debe de haber algo común, o no se llamarían juegos, sino que busquen y vean si hay algo en

común a todos ellos. Porque si los observan, no encontrarán algo que sea común a todos, sino similitudes, relaciones y un conjunto de éstas en los mismos" [21]. Esto es un obstáculo importante de las ontologías clásicas, acostumbradas a utilizar la teoría clásica de categorías, con colecciones de instancias que comparten un conjunto de atributos.

3. Esquemas de Imagen y Modelos Cognitivos

La semántica cognitiva incluye además otros conceptos para elaborar su teoría. Uno de ellos es la noción de *esquemas de imagen*, considerados por Johnson [8] como patrones incorporados (en inglés, embodied) de una experiencia organizada significativamente, tal como las estructuras de movimientos corporales y de interacciones perceptuales. Algo que debemos tener en cuenta es que los esquemas de imagen no son imágenes concretas, ricas o una visión mental, es decir lo que normalmente consideramos imágenes, sino estructuras que organizan nuestras representaciones mentales. Algunos ejemplos de esquemas de imagen son los de *origen-camino-destino*, *enlace* o *contenedor*. La semántica cognitiva considera que las estructuras conceptuales significativas derivan de la naturaleza estructurada de la experiencia corporal y social y de nuestra capacidad innata para proyectar imaginativamente desde ciertos aspectos bien estructurados de la experiencia corporal e interactiva a estructuras conceptuales abstractas. Una diferencia importante entre el enfoque objetivista y la semántica cognitiva es que, mientras que el primero considera el significado desde el punto de vista de la teoría de correspondencias (la asociación de símbolos con objetos externos), la segunda ve el significado como abarcando una proyección imaginativa usando mecanismos de metáfora, metonimia, categorización, esquematización, etc. Estos mecanismos se usan para trasladarnos desde lo que experimentamos de una forma estructurada como patrones recurrentes con nuestros cuerpos hacia estructuras más abstractas (modelos cognitivos).

4. La Abstracción como Simplificación

Decíamos en la introducción que hay dos aspectos de la abstracción, uno de los cuales era el de eliminar detalles a efectos de simplificar y centrar la atención en lo importante. Esto, de acuerdo a Kramer [9], se logra mediante dos procesos:

- El hecho de extraer o sacar algo.
- El acto o proceso de dejar fuera de consideración una o más propiedades de un objeto complejo para prestar atención a otros.

Kramer señala que la abstracción como simplificación es ampliamente utilizada en otras disciplinas, tales como el arte y la música y ofrece diversos ejemplos de esas disciplinas. Tal como ocurrió con el concepto de conocimiento, que fue considerado durante largo tiempo como un fenómeno inherente exclusivamente a las disciplinas científicas hasta que los trabajos de Piaget [17] mostraron que, en realidad, es algo que se construye de forma continua a través de los diversos estadios por los que pasa el ser humano (y de ahí la denominación de Epistemología

Genética), también es posible mostrar una situación similar con la abstracción, en el sentido de que la utilizamos constantemente en nuestra vida cotidiana y no exclusivamente en la informática, la matemática, el arte o la música. Tomemos por ejemplo el caso de la palabra 'ventana' y los distintos sentidos que puede adquirir en función de la oración en la que la empleamos. Tal como indica Taylor [20], podemos comparar 'romper la ventana' y 'pintar la ventana' en sus interpretaciones normales. Al analizar ambas oraciones podemos reconocer aquí diferentes entidades. En el primer caso, lo que se rompe, probablemente, es un panel de vidrio (pero no el marco que lo rodea); en el segundo caso lo que se pinta es el marco que rodea los paneles pero no los vidrios. Por supuesto que existen otras lecturas: romper la ventana podría significar la ruptura del marco con el vidrio incluido y pintar la ventana podría indicar que lo que se pinta es el vidrio pero no el marco. Pero no significa esto que exista una ambigüedad en dichas oraciones debida a estas posibles lecturas. Más bien

> *las lecturas preferidas parecen emerger del conocimiento general de lo que son las ventanas y de lo que está involucrado convencionalmente en 'romper' como opuesto a 'pintar una ventana'.* [20] p. 265

El lingüista que ha aclarado este tipo de fenómeno ha sido Langacker [12] al plantear que cuando una entidad e está involucrada en alguna predicación, normalmente sólo algunos aspectos de e –la 'zona activa'– participan en la relación. También señala el autor que el fenómeno de la zona activa es omnipresente y que solamente en casos muy raros todas las partes de una entidad participan igualmente en la predicación (un ejemplo sería el de *el coche chocó con el camión*, oración en la que están involucradas ambas entidades totales : el coche y el camión). Al seleccionar sólo una parte activa de la totalidad como cuando elegimos sólo el panel de vidrio o el marco en el caso de la ventana estamos efectuando una extracción, una selección que corresponde al concepto de abstracción. La observación de Langacker en el sentido de que se trata de un fenómeno omnipresente, nos permite limitarnos en la cantidad de ejemplos aunque podrían incluirse innumerables. En otro ejemplo, el de el perro mordió al gato también estamos seleccionando una zona activa, en este caso el hocico del perro, sus dientes, la mandíbula y los músculos asociados. Tal como indica Langacker:

> *La zona activa no debe concebirse como una región discreta o claramente delimitada dentro de la entidad total, es más exacto pensarlo como el área focal de la interacción relacional, la participación de una región se vuelve más tenue a medida que se aleja de su foco.* [12] p. 190

Vemos entonces que la abstracción por simplificación o por adecuación es un fenómeno que aparece constantemente en la vida cotidiana mediante el uso del lenguaje y que trasladamos a otras disciplinas a efectos de poder manejarnos con procesos de una cierta complejidad. A diferencia del uso que hacemos en informática, en la vida cotidiana no explicitamos –ni representamos– las cualidades específicas que hemos seleccionado mediante la zona activa.

5. La abstracción como generalización

El segundo aspecto de la abstracción, el de la generalización, es el mecanismo que sirve para identificar el núcleo común basado en las siguientes definiciones:

- El proceso de formular los conceptos generales extrayendo las propiedades comunes de las instancias y

- Un concepto general formado mediante la extracción de características comunes de ejemplos específicos

En particular, este mecanismo puede verse claramente en la programación con el uso de abstracciones de datos y clases en la programación orientada a objetos. Kramer [9] comenta que la interpretación abstracta para el análisis de programas es otro ejemplo de generalización, en la que el dominio del programa concreto se hace corresponder con un dominio abstracto para capturar la semántica de la computación mediante el análisis de programas.

> *En realidad, la abstracción es fundamental para las matemáticas y la ingeniería en general, jugando un rol crítico en la producción de modelos para el análisis y en la producción de sólidas soluciones de ingeniería.* [9], p. 40

Vamos a mostrar que la abstracción como generalización corresponde al concepto de la semántica cognitiva de categorización. La idea de que las categorías se definen en base a propiedades comunes a un conjunto de entidades es algo inherente a la teoría tradicional de lo que es una categoría y la hemos heredado a través de más de dos mil años de historia. Esta visión clásica de la categorización que ha sido puesta en tela de juicio por la semántica cognitiva no debe considerarse como totalmente errónea sino solamente como limitada. La prueba de su validez es que muy frecuentemente categorizamos a las cosas sobre esta base y que, evidentemente, es esta manera de categorizar la que podemos asociar directamente con la abstracción como generalización, tal como la plantea Kramer. En las definiciones de arriba vemos claramente que se habla de formular o formar conceptos en base a la extracción de características comunes y eso corresponde a la visión clásica de la categorización. Pero lo que sostiene Lakoff es que la categorización es mucho más compleja que lo que sostiene la teoría clásica:

> *Ha surgido una nueva teoría de la categorización, denominada teoría de prototipos. Muestra que la categorización humana está basada en principios que se extienden mucho más allá de los previstos en la teoría clásica. Uno de nuestros objetivos es estudiar las complejidades de la forma en que la gente realmente categoriza. Por ejemplo, el título de este libro [Mujeres, Fuego y Cosas Peligrosas] ha sido inspirado por el idioma aborigen de Australia, el Dyrbal, que tiene la categoría balan, que en realidad incluye a las mujeres, al fuego y las cosas peligrosas. También incluye pájaros que no son peligrosos además de animales excepcionales...* [10], p. 5.

Indica, entonces, que la teoría de prototipos abarca otros fenómenos lingüísticos que generan categorías con elementos tan heterogéneos –en nuestra concepción occidental– como las que menciona Lakoff y que no tienen cabida dentro de la teoría clásica que sólo incluye los casos de categorías basadas en la teoría conjuntística clásica. Lo importante a esta altura es mostrar que existen otros mecanismos de categorización, no basados exclusivamente en la recogida de atributos comunes sino en otros procesos cognitivos como pueden ser la metáfora, la metonimia, etc. Estas formas alternativas de categorizar nos permitirán esbozar otros mecanismos utilizados también en informática como veremos luego con el mecanismo de concretización.

6. Conceptualizar lo nuevo

Aparte del mecanismo de la abstracción, otro al que debemos apelar muy frecuentemente es el de conceptualizar un nuevo sistema (sistema operativo, aplicativo, middleware, etc.). A lo largo de la historia de la informática hemos usado una forma no tradicional de crear los nuevos conceptos que se necesitaban. Hemos mostrado en el libro Designing with Blends [5] que la mayor parte de los conceptos –si no todos– utilizados en informática son figurativos y tomados de diferentes dominios: del dominio jurídico (sentencia, proceso, ejecución, etc.), de la construcción (arquitectura, base, marco, etc.), de la manufactura (tubería, herramienta, caja de herramientas, artefacto, etc.), de la organización y los negocios (cliente, agente, broker, servicio, etc.), de la oficina (carpeta, fichero, papelera, etc.), de la actividad social (cola, sesión, control, etc.), de la biología (virus, taxonomía, gusano, etc.) y de otros dominios igualmente.

Es decir, hay un esfuerzo constante por presentar de una forma concreta cosas que serían difíciles de 'visualizar' por sus características abstractas, pero eso no nos hace olvidar sus orígenes metafóricos. Por eso mismo, no debemos olvidar tampoco las características metafóricas de la propia ingeniería de software. Aquí podemos indicar también que, al igual que se establece una diferencia entre la ingeniería tradicional y la ciencia, tal como lo hace Petrosky [16] al señalar que:

- La ciencia busca entender lo que existe, mientras que
- La ingeniería busca crear lo que nunca existió

También podríamos diferenciar, de la misma manera, la ingeniería de software respecto de la ciencia de la computación sobre la que esta especial ingeniería se basa. No se trata de describir la ingeniería –nos dice Petrosky– simplemente como ciencia aplicada puesto que hay en la primera un elemento añadido, un componente que podría considerarse como creativo o artístico. El ingeniero que diseña un puente no deduce su forma a partir de leyes científicas o de ecuaciones matemáticas sino que lo visualiza previamente en su mente, de la misma forma que se concibe un poema o un cuadro. Y finalmente señala que, en ingeniería, el análisis sigue a la síntesis y no al revés.

Este es el aspecto que queremos recalcar, que plantea Petrosky en términos de síntesis que antecede al análisis, y que contradice lo que históricamente se

ha mantenido en ingeniería de software, una secuencia que parte de la captura de requisitos, continúa con el análisis y sigue con el diseño. En esta secuencia, el análisis precede al diseño, que sería el equivalente de la síntesis. Este aspecto requeriría un análisis más detallado de la forma en que se entrelazan el análisis y el diseño.

Esto nos indica que hay un trabajo de concretar, de dar forma al sistema que vamos a construir antes de comenzar la siguiente etapa de análisis. Esta forma de conceptualizar lo nuevo se aparta de la categorización (abstracción como generalización) puesto que no disponemos de instancias concretas a partir de las cuales se puedan extraer atributos comunes. Necesitamos un concepto que nos provea una imagen global de la que no disponemos previamente, razón por la cual vamos a generarla a partir de ciertos procesos cognitivos como puede ser la metáfora y las mezclas.

7. Metáforas y Mezclas

La metáfora establece una correspondencia entre dos dominios diferentes y es uno de los mecanismos centrales en nuestros procesos de pensamiento. Se la utiliza muy frecuentemente en el lenguaje de la vida cotidiana, por ejemplo cuando usamos expresiones tales como 'hay que agarrar el toro por los cuernos', 'el tiempo es oro'. o 'sacó a sus hijos adelante'. Producimos una metáfora cuando conceptualizamos un dominio o esfera de actividad en términos de entidades y relaciones de otro. Lo importante de la metáfora es que no se trata de una mera figura del lenguaje sino que es un proceso cognitivo cuya manifestación son las diversas expresiones lingüísticas que se derivan de la misma metáfora conceptual, es decir, el proceso cognitivo que subyace a todas ellas.

Por ejemplo, diversas expresiones que se utilizan, tales como 'no entrar al trapo'. (para indicar que deben evitarse las provocaciones), 'cambiar de tercio' (para indicar que se cambia de tema) o la misma 'agarrar el toro por los cuernos' derivan de una misma metáfora conceptual y que es la de 'la vida es una corrida de toros'.

Podemos referirnos a la metáfora en términos de los nombres de los dominios que nos da la siguiente estructura: El dominio meta *es* el dominio fuente. Es importante recalcar que la metáfora es una verdadera relación de equivalencia. A diferencia de la analogía, que establece que una cosa es similar a otra, la metáfora dice que *A es B*, como cuando en lugar de decir que vamos a 'cambiar de tema' lo expresamos como 'cambiar de tercio' y no añadimos que cambiar de tercio es análogo a decir que cambiamos de tema, sino que son equivalentes.

Esta característica de la metáfora es la base para otro proceso cognitivo en el que mezclamos dominios diferentes (a semejanza de los compuestos químicos) para producir nuevos conceptos que llamamos mezclas (en inglés, blends). Es el caso histórico de la ingeniería de software, para lo cual es preciso recordar que "el término ingeniera de software fue elegido de forma deliberada por ser provocativo" [15] por los fundadores de la disciplina en una legendaria conferencia de la NATO de 1968.

En este caso, se produce en primer término una metáfora (el desarrollo de software es una ingeniería) y a partir de dicha metáfora producimos a continuación la mezcla que conocemos en la actualidad como ingeniería de software, es decir el emergente que consiste en un nuevo tipo de ingeniería. En esa mezcla, se han proyectado desde el dominio original de las ingenierías una serie de disciplinas que se han incorporado en el SWEBOK [19] como 10 áreas de conocimiento y entre las cuales podemos mencionar los requisitos, el diseño, la construcción, las pruebas o la calidad del software.

En toda mezcla aparecen características emergentes y en el caso de la ingeniería de software podemos mencionar el caso de la construcción como área independiente, a imagen y semejanza de las ingenierías tradicionales. Pero a esta etapa se la ha independizado cuando de lo que se trata es de diseño, que comienza con los requisitos y luego se van traduciendo a diversos lenguajes hasta llegar finalmente al lenguaje binario comprensible para la computadora. En ese sentido

> *Elaboramos una serie de modelos cuya traducción está en un nivel inferior, pero ninguno de esos modelos tiene un status privilegiado como para llamarlo construcción. Se trata de una forma metafórica de hablar, derivada de la metáfora original de la ingeniería de software* [6]

8. Conceptualización o concretización

Decíamos arriba que, para conceptualizar lo nuevo, necesitamos un concepto que nos provea una imagen global de la que no disponemos previamente. Este procedimiento es crucial para concebir lo que vamos a diseñar aunque podríamos decir que es el primer paso fundamental del diseño y es preferible llamar conceptualización a lo que previamente hubiéramos podido llamar categorización. Hemos visto que, en realidad, la categorización es una forma de abstracción que selecciona aspectos comunes de las entidades para determinar un cierto tipo de similitud entre ellas. Podemos decir –usando la jerga de la informática– que se trata de un enfoque de abajo hacia arriba (bottom-up) o que parte de lo concreto y se dirige hacia lo abstracto.

Pero lo que necesitamos en el momento de concebir un sistema nuevo es un concepto que todavía no existe –o puede no existir–, que es necesario crear de forma adecuada para una instancia que habrá que generar como resultado de ese diseño, que visualizamos en la mente de la misma manera que lo hacemos con un poema o un cuadro, tal como plantea Petrosky. Es evidente que con frecuencia disponemos de otras instancias en la realidad que nos permitirían proyectar varios (o todos) los atributos a efectos de diseñar la nuestra. Pero es frecuente querer dar el toque de autor mediante la inclusión de atributos nuevos que lo diferencien de las otras instancias.

Es en este momento en el que disponemos de los procesos cognitivos de la metáfora y la mezcla. Consideremos el ejemplo paradigmático –por la emergencia de lo nuevo– de la creación de la interfaz gráfica para el sistema operativo del Macintosh en el año 1984. Lo importante es analizar lo que ocurrió, abstrayendo

lo esencial independientemente de dónde se produjo. El primer paso básico fue la decisión de concebir la interfaz gráfica como una metáfora del escritorio. En ese sentido, la primera idea es la de conceptualizar, es decir plantear la metáfora original, la de "el sistema operativo es un escritorio" que nos permitirá, a partir de entonces, analizar los elementos de esa totalidad. Una vez concebida la idea, hemos obtenido una síntesis a partir de la cual se nos permitirá realizar un análisis, tal como indica Petrosky cuando habla de la síntesis que precede al análisis o en los términos habituales de arriba hacia abajo (top down).

Para proceder al análisis, disponemos de una imagen concreta, en la medida que podemos hablar de concreto al referirnos a un escritorio con todos sus elementos. Es decir, podemos entrar a analizar los diversos elementos de la interfaz gráfica –el dominio destino– buceando en los elementos de un escritorio real –el del dominio fuente– y, de esa forma, ir reconociendo ciertas entidades tales como los ficheros, las carpetas, la papelera, etc.

La conceptualización mediante el uso de una metáfora no es la única forma de hacerlo. Es común también utilizar más de una metáfora cuando las características del producto así lo requieren. Por ejemplo, Gerald Johnson [7] se declara sorprendido de que un repositorio pueda ser al mismo tiempo un diccionario e incluso una base.

> *IBM ha planificado anunciar su largamente esperado repositorio, o diccionario central de datos, para la Arquitectura de Sistemas de Aplicación (SAA) para comienzos de 1989 como muy tarde, de acuerdo a consultores del sector. IBM espera que el repositorio aparezca como una facilidad de almacenamiento para información vital a través del sistema operativo. Pero en su primera implementación, el diccionario actuará ampliamente como una base subyacente para un sistema de ingeniería de software asistido por computadora (CASE)* [7] p. 100

En realidad no existe ninguna dificultad en la descripción multimetafórica puesto que el párrafo elabora una integración conceptual o mezcla en base a una serie de metáforas (repositorio, diccionario, almacenamiento y base) que aportan algunas características al nuevo producto mediante el mecanismo de proyección selectiva [5].

La conceptualización también puede corresponder a la visión de una escena o sistema, y en ese caso la perspectiva desde donde vemos dicha escena es un arreglo visual, cuyo aspecto más obvio es el punto de vista (en inglés, vantage point) asumido. La misma situación objetiva puede observarse y describirse desde muchos puntos de vista diferentes que derivan en diferentes conceptualizaciones que pueden tener diversas consecuencias [13].

Un ejemplo reciente de cómo resultan distintas conceptualizaciones en función del punto de vista es la dificultad de obtener consenso en la definición de SOA (Service Oriented Architecture), tal como muestran Holley y Arsanjani [11] con las diferentes definiciones ofrecidas por el OASIS Group y el Open Group.

Así, el arquitecto de tecnología de la información (IT) conceptualiza SOA como una solución arquitectónica para integrar diversos sistemas suministrando

un estilo arquitectónico que promueve el acople débil y la reutilización. Para el desarrollador, SOA es un modelo de programación o paradigma en el que los servicios web y los contratos aparecen como un diseño dominante para lograr la interoperabilidad; para el director de arquitectura o arquitecto de empresa, SOA es un medio de crear aplicaciones dinámicas, altamente configurables y colaborativas para lograr el cambio y reducir la complejidad y la rigidez de IT y –finalmente, aunque incluyen otras conceptualizaciones– para el analista de negocio SOA es una manera de desbloquear el valor, porque los procesos de negocio dejan de estar bloqueados en silos de aplicaciones.

9. Conclusiones

Hemos mostrado que, al igual que la abstracción, hay otro proceso cognitivo que tiene un rol importante en informática y es el de conceptualización. Además de lo que pregona la teoría clásica de la categorización en el sentido de buscar los atributos comunes a una serie de instancias para constituir una determinada categoría que las abarque, existen formas alternativas y usuales mediante las cuales utilizamos la metáfora, la metonimia y otros tropos para crear nuevos conceptos. Ha sido a través del uso de metáforas (arquitectura, bus, broker, virus, escritorio) que vamos generando nuevos conceptos informáticos y, por ejemplo, fue a través de una famosa –la del escritorio– que hemos podido concebir o conceptualizar una nueva interfaz para los sistemas operativos. Esta metáfora original permite luego una elaboración más detallada a través de otros procesos cognitivos que son con frecuencia los de *mezclas* para completar la elaboración de nuevos sistemas, aplicaciones u objetos.

Otros ejemplos recientes nos ilustran sobre la utilización de múltiples definiciones simultáneas (o uso de múltiples metáforas) para un mismo fenómeno en función de los puntos de vista que adoptamos y que se refleja en informática con las diversas visiones de los involucrados (stakeholders) cuando creamos un nuevo sistema o aplicación.

Esto es una demostración de que la informática –y la ingeniería de software– pueden verse beneficiadas en gran medida mediante una mayor inclusión de ciencias humanas entre las disciplinas que las constituyen.

Agradecimientos. El autor agradece a Mauricio Milchberg los valiosos comentarios sobre una versión previa del artículo.

Referencias

1. Gruber, T.: What is an Ontology? http://www-ksl.stanford.edu/kst/what-is-an-ontology.html
2. Gruber, T.: Ontology. In Encyclopedia of Database Systems. Eds. Ling Liu, M. Tamer Özsu. Springer Science+Business Media, LLC (2009)

3. Hailperin, M., Kaiser, B and Knight, K.: Concrete Abstractions: An Introduction to Computer Science Using Scheme. Brooks/Cole Publishing Company, a division of International Thomson Publishing Inc. (1999). Como está agotado, puede obtenerse en: http://www.gustavus.edu/+max/concrete-abstractions.html
4. Hollan, J., et al.: Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research. ACM Transactions on Computer-Human Interaction 7, no. 2: pp. 174-196 (2000)
5. Imaz, M., Benyon, D.: Designing with Blends. The MIT Press. pp. 63-66 (2007)
6. Imaz, M.: Ingeniería de Software: ¿Roca sólida o evanescencia?. CLEI 2010, XXXVI Conferencia Latinoamericana de Informática, 18 al 22 de octubre San Lorenzo, Paraguay. 2.1.1_41CLEIImaz_Paper. (2010)
7. Johnson, G.: Of Metaphor and the Difficulty of Computer Discourse. CACM 37, no. 12: pp. 97-102 (1994)
8. Johnson, M.: The Body in the Mind: The Bodily Basis of Reason and Imagination. University of Chicago Press. (1987)
9. Kramer, J.: Is Abstraction the Key to Computing?. Communications of the ACM. Vol. 50, No. 4, pp. 37-42 (2007)
10. Lakoff, G.: Women, Fire, and Dangerous Things: What Categories Reveal about the Mind. Chicago: University of Chicago Press (1987)
11. Kerry, H and Arsanjani, I.: 100 SOA Questions – Asked and Answered. Pearson Education, Inc. (2011)
12. Langacker, R.: Active zones. In Concept, Image, and Symbol: The Cognitive Basis of Grammar. Cognitive Linguistics Research. 1. Berlin: Mouton dc Gruyter, 189-201. (1990)
13. Langacker, R.: Cognitive Grammar: A Basic Introduction. Oxford University Press, Inc., New York. (2008)
14. Margolis E. and Laurence, S. (Eds.): Concepts: Core readings. A Bradford Book. The MIT Press. (1999)
15. Naur, P., Randell, B. (eds.): Software Engineering: Report on a Conference Sponsored by the NATO SCIENCE COMMITTEE Garmisch, Germany, 7th to 11th October (1968)
16. Petroski, H.: The Essential Engineer: Why Science Alone Will Not Solve Our Global Problems. Knopf. (2011)
17. Piaget, J.: L'epistémologie génétique. Collection "Que sais-je?" Quadrige, PUF. (2011)
18. Sicilia, M.A., Kop, C. and Sartori, F., (Eds.): Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Science. Springer-Verlag, Berlin, Heidelberg. (2010)
19. SWEBOK. Se podía descargar libremente, ahora puede adquirirse on-line en el sitio oficial de IEEE Computer Society Store.
20. Taylor, J: Linguistic Categorization: Prototypes in Linguistic Theory. Oxford University Press, USA, 2nd. edition. (1995)
21. Wittgenstein, L.: Philosophical Investigations (1953). In Margolis and Laurence (Eds.). A Bradford Book. The MIT Press.