Academia y desarrollo

Paradigmas en la construcción de software

Sandro Javier Bolaños Castro

RESUMEN

Enfrentarnos con el modelamiento de la realidad ha sido una de las tareas que desde los principios de la humanidad hemos realizado. Las pinturas que sobre cavernas se han descubierto, no son más que la expresión de lo que el hombre sentía y veía en su entorno. Mucho después continuamos cuestionándonos sobre nuestra realidad y cómo representarla de manera que se pueda tener control sobre ella. Ésta es justo la propuesta que persigue un paradigma; estructurar un conjunto de mecanismos que permitan de manera formal organizar una realidad, sobre la cual se puedan tomar decisiones. En este artículo, se presentan los enfoques más importantes acerca de cómo puede ser vista y modelada la realidad vía la construcción de software.

Palabras claves: paradigma, mecanismos, modelamiento de la realidad, construcción de software

ABSTRACT: PARADIGMS TOWARD SOFTWARE CONSTRUCTION

The reality modeling has been one of the task that since the beginning of mankind has been made. The paintings discovered in caves are the expression of what man felt and saw around him. Sometime after that, we still continue asking ourselves about our reality and how to represent it in such away that we could master it. This is just the proposal that follows the paradigm, to structure a set of mechanims that allow, in a formal way, to organize the reality wich we could make decisions about. This article presents the most important points of view, about how reality, can be seen and modeled towards the software construction.

Key Words: paradigm, mechanims, reality modeling, software construction.

INTRODUCCIÓN

"Un cohete que impulsaba el Mariner, una sonda espacial con destino a Venus, tuvo que ser destruido 290 segundos después del lanzamiento el 22 de julio de 1962, la pérdida se estimó entre 18 y 20 millones de dólares.

El programa de computador en tierra debió contener el siguiente fragmento: ***Si no*** *hay contacto del radar con el cohete* ***entonces*** *no corregir su trayectoria*

El ***no*** *fue ignorado inadvertidamente, así que el computador en tierra continuó guiando a ciegas el cohete después de que se perdió el contacto con el radar. El cohete se extravió y fue destruido antes de que pudiera poner en peligro vidas humanas. El programa se había utilizado previamente, sin ninguna falla, en cuatro viajes a la luna"* [1].

Problemas como éste, en los que están involucradas grandes inversiones e incluso vidas humanas son frecuentes, la pregunta es quién es el responsable? Igual o quizá más compleja que la pregunta, es la respuesta, sin embargo, un buen punto de partida lo constituye la revisión del modelo de construcción de software que se emplee. Siempre que se trate de desarrollo de software se presenta un importante porcentaje de riesgos, el punto clave es administrarlos y/o en lo posible eliminarlos. No es tarea fácil saber todas las posibles variaciones que un problema pueda tener, y más aun cuando se trata de solucionar un sistema altamente probabilístico. Un ingrediente que se le suma al problema es el poder determinar con exactitud las variables implicadas y sobre las cuales lógicamente se tienen que determinar mecanismos de control, es decir, se debe tener pleno conocimiento del dominio del problema sobre el cual se quiere ofrecer una solución.

Todas las consideraciones que se tengan en cuenta para tratar de resolver el problema de desarrollar software confiable y de calidad, convergen sin lugar a dudas en el manejo de la complejidad que entorno al problema existe. Desafortunadamente no se tiene una fórmula mágica que garantice el ciento por ciento de efectividad, pero si propuestas que se acercan a una solución satisfactoria. Éstas propuestas, que las podemos reunir como modelos o paradigmas, precisamente buscan organizar el conocimiento y así hacerlo accesible y manejable. Un paradigma se puede definir como: "*un conjunto de teorías, estándares y métodos que representan en conjunción una forma de organizar el conocimiento*"[2] . y es justo estos elementos claves los que se proponen en modelos como el estructurado o el orientado a objetos.

Todas las consideraciones que se tengan en cuenta para tratar de resolver el problema de desarrollar software confiable y de calidad, convergen sin lugar a dudas en el manejo de la complejidad que entorno al problema existe.

I. EVOLUCIÓN DE LOS MODELOS

"*Nuestra vida se desarrolla, con base en paradigmas, en supuestos teóricos generales, en técnicas y procedimientos de aplicación que adaptamos por criterio de una comunidad*"[2].

La apreciación de Kuhn apunta a las propuestas que entorno al software se han desarrollado, precisamente ese conjunto de supuestos teóricos han sido producto de lo que históricamente ha pasado en las ciencias computacionales. En el sentido estricto de la palabra, desde la propuesta de Von Neumann acerca de la primera máquina, no se ha avanzado mucho, excepto por las mejoras tecnológicas que se han hecho, pero fundamentalmente manteniéndose el mismo modelo. Éste es el primer parámetro de referen-

Cada enfoque propuesto por un paradigma enfatiza en una filosofía que pretende describir las formas de ver el mundo y cómo lógicamente expresarlo en una máquina.

cia sobre el cual crecerían las teorías que buscan dar una solución para el mundo real, apoyándose en una máquina sobre la cual se esquematiza una interpretación lógica que permite organizar el conocimiento.

El planteamiento teórico realizado en un modelo, debe mezclar la interpretación del mundo real con la interpretación que, al nivel de la máquina, se debe hacer del problema. Ambos enfoques deben converger a una solución viable sustentada sobre un sistema computacional. Dentro de los paradigmas de desarrollo de software más importantes podemos destacar: imperativo, funcional , de modelamiento de datos, declarativo, orientado objetos, incluso, el paradigma de modelamiento.

La perspectiva que se tenía del mundo y que enmarcó el modelo imperativo, fue influenciada fundamentalmente por la mismas ideas que concebían la organización como unidades funcionales y por tanto, las abstracciones significativas, igualmente debían ser procesos en los que entraban insumos para ser transformados en productos. El método utilizado por el paradigma imperativo fue el método estructurado en el que se reflejaba la organización, cuyo problema podía ser resuelto en una función, "proceso", a la cual se le entregaban argumentos, "insumos", y de la cual se obtenía un resultado, "productos". El modelo imperativo asume una realidad de procesos por tanto el principal énfasis es la acción, "orden", sobre el conjunto de variables que configuran el problema. Otra propuesta interesante es la hecha por el modelo funcional, ésta tiene una gran influencia del modelo imperativo, pues al igual que en éste, el centro de abstracción lo constituye la función, la cual es tratada como valor de primera clase, y de ahí que herede su nombre. La diferencia radical está en el almacenamiento; mientras en el modelo imperativo es fundamental, en el modelo funcional el almacenamiento es implícito, eliminando los graves problemas de transparencia referencial. Aunque el modelo propone una buena alternativa a los problemas colaterales ocasionados al tratar con funciones; es poco práctico en su semántica. Otro enfoque es el planteado por el paradigma de modelamiento de datos, en éste; se ofrece una solución basada fundamentalmente en la necesidad de contar con un esquema ágil para el manejo de información; el centro de abstracción se dirige hacia los datos que reunidos constituyen entidades y que relacionados constituyen un sistema de información. El paradigma de modelamiento de datos o modelo relacional se aparta un poco del esquema propuesto por el modelo imperativo centrando su interés en las variables del problema, dejando las operaciones en segundo plano e incluso agrupadas dentro de un conjunto de interés compuesto entre otras por: consultas, búsquedas, y almacenamientos. Otro de los enfoques interesante lo constituye el modelo declarativo, en éste; se organiza el conocimiento basándose en reglas con las que se garantizan inferencias conducentes a soluciones. El modelo declarativo busca mostrar la realidad como una consecuencia lógica de relacionar coherentemente las variables del problema. Finalmente se tiene como modelo relevante, el orientado a objetos, cuya abstracción de la realidad converge al objeto como entidad dotada de operaciones, información, semántica y relaciones significativas dentro del dominio del problema. El paradigma orientado a objetos emplea como método la orientación a objetos en la que se busca modelar una perspectiva del entorno, basada en atributos encapsulados junto con sus comportamientos.

Cada enfoque propuesto por un paradigma enfatiza en una filosofía que pretende describir las formas de ver el mundo y cómo lógicamente expresarlo en una máquina. Sin embargo, algunos ofrecen mejores mecanismos para organizar el conocimiento y hacerlo fácilmente manejable. Se debe tener presente que un modelo no es la expresión pura de abstracción de la realidad, más bien es una conceptualización combinada de modelos, por tanto podría ser un referente importante tomar el modelo que ofrezca los mecanismo que, de manera comprobada, permitan controlar mejor la complejidad.

II. ¿POR QUÉ ESCOGER ORIENTACIÓN A OBJETOS?

La orientación a objetos no es un modelo tan novedoso como se pudiera pensar, las primeras ideas sobre el modelo ya venían desde Small Talk, sin embargo, el predominio del modelo imperativo no lo dejó emerger con fuerza hasta los ochenta. Desde una perspectiva histórica, la orientación a objetos surge como resultado de un proceso de evolución en los mecanismos de abstracción, los cuales pueden ser vistos como la progresión lógica que empezó en la función , luego pasó por los módulos y TADs para llegar finalmente a los objetos [3]. La función que se constituye como el principal mecanismo de abstracción en los modelos imperativo y funcional; es la primera forma importante de representar la realidad, la consigna fundamental es "dividir y vencer", como manera de romper un procedimiento en varios subprocedimientos y obtener una solución atomizando el problema; es la edad de oro del método estructurado y del mundo de procesos. Las exigencias tecnológicas hacen que la función no sea lo suficientemente robusta como para soportar ciertos requerimientos de modelaje comenzando a acusar problemas como la falta de seguridad, identidad, reusabilidad a mayor escala, y consistencia entre otros. Aparece entonces el módulo en el cual se agrupan datos y operaciones, con lo cual se eliminan problemas, como la falta encapsulación, y se adicionan principios, como el de seguridad, permitiendo además establecer un espacio de nombres con visibilidad privada y pública, que constituyen implementación e interfaz respectivamente, pero aún faltaría algo importante, establecer mecanismos de instanciación.

Llegan entonces los TADs en los que se incorporan todas las ideas y mecanismos de los módulos, introduciendo la identidad producto de la instanciación y que permite sobre ella montar conceptos como reutilización, semántica y formalización. Finalmente, surgen los objetos como medio de hacer concreto el planteamiento formal expresado en los TADs y sobre los cuales se incorporarán una serie de principios que hacen del modelo orientado a objetos, buena elección en el momento de pensar en la construcción de software. Lee y Tepfenhart [3], proponen que en la orientacion a objetos los principios fundamentales son: encapsulación, ocultación de información, paso de mensajes, atado posterior, delegación, clase - objetos, herencia - polimorfismo y relaciones. Ver fig 1.

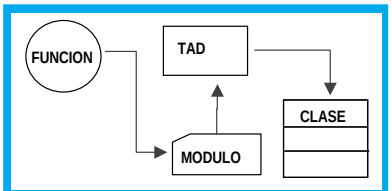

Figura No. 1

La encapsulación es uno de los mecanismos obtenidos por evolución más interesantes, éste permite reunir estados y comportamientos en un módulo que divide el espacio de nombres, en los contextos público y privado, introduciéndose así, el concepto de privacidad. Es importante tener presente que la encapsulación no es propia de la orientación a objetos, el mismo modelo imperativo, incluso el funcional, tienen encapsulación a nivel de sus procedimientos, la diferencia radical está en que el objeto tiene un mayor nivel de encapsulamiento, ya que en él se reúnen tanto operaciones, como atributos. La encapsulación por sí misma debe ser complementada con la ocultación de información, la cual define cómo los servicios de un objeto ponen a disposición sus métodos para que otros objetos los utilicen, permitiendo que se hagan las restricciones sobre lo que el objeto utiliza para sí y lo que el objeto ofrece; además buscando al máximo que sus características estructurales no sean accedidas desde fuera a no ser que se empleen sus interfaces controladas. Dado que la percepción de la realidad se dirige hacia la abstracción de objetos es necesario establecer un mecanismo que les permita comunicarse, éste es precisamente el envío de mensajes, con el cual se garantiza que un objeto agente utilice los servicios ofrecidos y que necesita de un objeto cliente, previo cumplimiento del protocolo exigido. Como en el paso de mensajes se exige un origen, un destino y una información, a nivel de un objeto en particular se asegura que sea exactamente ese el servicio requerido y no otro, situación que otros modelos no se garantiza. Cuando se solicita un servicio a un objeto se puede optar por definirlo con antelación o definirlo justo en el momento del requerimiento, ésta posibilidad está estrechamente ligada con el comportamiento polimórfico que puede tener un objeto y que garantiza una abstracción versátil. El modelo también permite utilizar delegación, basada en el principio de burocracia perfecta en la cual se cumple con un trabajo buscando ayuda en las fuentes que tienen la información y en las que tienen los servicios que puedan procesarla, es decir desde la perspectiva del cliente el agente es quien tiene el servicio que el cliente necesita, por tanto él delega la tarea sin que ello implique delegar la responsabilidad, pues ella sigue estando en el cliente quien es propietario de la información que será manipulada. Éste principio es parecido a la división de tareas manejado en otros modelos pero, dista en filosofía. Una de las consecuencias más importantes que trajeron los TADs, fue precisamente el poder hacer concreta una abstracción, es decir poder inferir de una clase una instancia que constituirá el objeto, centro del modelo. El hablar de objeto es hablar de identidad, es decir, la propiedad de ser un única entidad, incluso en un mismo contexto en donde existan objetos de la misma especie, propiedad que lógicamente modelos como el imperativo o funcional no poseen. Los objetos también pueden ser organizados como estructuras jerárquicas de herencia, copiando uno de los principios básicos de la naturaleza y planteamiento fundamental de la teoría evolucionista de Darwin. La taxonomía que permite organizar las especies también nos permitirá organizar nuestras abstracciones de la realidad, en la que los objetos pueden reutilizar características de sus padres, tanto estructurales como de comportamiento, incluso modificarlas y ampliarlas. Finalmente la orientación a objetos ofrece el principio de las relaciones, en las que se establecen las asociaciones, como relación estructural que conecta los objetos y que permite la construcción de modelos bien formados.

Cada uno de los principios de la orientación a objetos trae consigo mecanismos importantes para lograr que la abstracción de la realidad sea, entre otras características, coherente, flexible, escalable y segura. Otros modelos hablan de principios similares en otras escalas, como es el caso del modelo relacional en el cual se tienen tablas, campos y registros que corresponderían a clases, atributos y objetos respectivamente. Incluso se plantean otros esquemas en los que no son tan identificables los principios orientados a objetos, como es el caso del modelo declarativo. Pero sea cual sea el modelo se puede identificar una estrecha relación con la orientación a objetos.

Los objetos también pueden ser organizados como estructuras jerárquicas de herencia, copiando uno de los principios básicos de la naturaleza y planteamiento fundamental de la teoría evolucionista de Darwin.

III. EL PAPEL DE LA PROGRAMACIÓN EN LA ORIENTACIÓN A OBJETOS

Aunque un modelo busca expresar las diferentes formas de ver la realidad, no es ajeno al esquema computacional sobre el cual se monta y de hecho no es ajeno a los mismos lenguajes de programación sobre los cuales en últimas hacemos la construcción. Ello ha hecho que dentro del mismo núcleo del paradigma existan enfoques de lenguajes de programación que tratan de ajustarse a la propuesta orientada a objetos, estas propuestas se pueden agrupar en forma progresiva acorde al enfoque de programa-

Los modelos utilizados para la construcción de software son influenciados por las tendencias que en el medio imperen, principalmente económicas y tecnológicas.

ción, según Lee y Tepfenhart [3], como: programación basada en objetos, programación basada en clases, programación orientada a objetos, programación orientada a objetos avanzados y programación orientado a objetos en el límite. Ver fig 2.

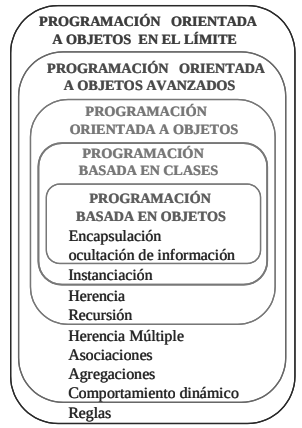

Figura No. 2

La programación basada en objetos, es una programación fundamentada en los principios de encapsulación y ocultación de información, carece de los restantes principios de la orientación a objetos; el Ada es lenguaje representativo para este enfoque. La programación basada en clases absorbe las características del anterior esquema de programación, añadiendo mecanismos propios de las clases, como es la instanciación; el CLU es el lenguaje representativo para este enfoque. La programación orientada a objetos absorbe las características del anterior esquema de programación, añadiendo mecanismos propios de la herencia y recursión; el Small Talk es el lenguaje representativo para este esquema. La programación orientada a objetos avanzados absorbe las características del anterior esquema de programación, añadiendo características como la herencia múltiple, asociaciones, agregaciones y comportamiento dinámico; el C++ es el lenguaje representativo para este esquema. La programación orientada a objetos en el límite observe las características del anterior esquema de programación, añadiendo el mecanismo propio del modelo declarativo, las reglas; el R++ es el lenguaje representativo para este esquema. En este último se hace la propuesta interesante de unir a través de un lenguaje el modelo orientado a objetos y el modelo declarativo.

El poder contar con una clasificación, incluso dentro de la misma propuesta orientada a objetos es fundamental, como base de elección tecnológica en el momento de construir los modelos que se plantean sobre un problema. Se pierden muchos esfuerzos y dinero si no se escoge adecuadamente una plataforma y un buen modelo. Por tanto el objetivo primordial es iniciar determinando los elementos conceptuales y tecnológicos que se requieren para sacar adelante un proyecto de software.

CONCLUSIONES

Los modelos utilizados para la construcción de software son influenciados por las tendencias que en el medio imperen, principalmente económicas y tecnológicas. Se aprecia además una progresión en la que se corrigen errores de modelos anteriores, señalando un camino de evolución de esquemas que inicialmente fueron menos complejos y que con el tiempo requieren absorber más mecanismos que permitan hacer mejores representaciones de la realidad.

Los paradigmas tienen un ciclo de vida en el que se desarrollan y son útiles hasta cuando se quedan cortos para ofrecer una solución adecuada al problema de organizar la complejidad. Justo en el momento en el que un modelo no es lo suficientemente robusto como para soportar el modelamiento de un problema surge una crisis que exige de nuevas propuestas, éstas generalmente son referidas a la crisis del software.

No se puede afirmar categóricamente que un modelo sea mejor que otro, lo que si su puede sostener es que un modelo sea más completo que otro, visto desde la perspectiva de los mecanismos que un modelo ofrece para permitir hacer una mejor abstracción de la realidad. Lógicamente entre más formas de ver la realidad compile un modelo, resultará más complejo utilizarlo.

La orientación a objetos es el modelo más ampliamente utilizado, por la variedad de características y riqueza de sus mecanismos de abstracción. Sin embargo no es el santo grial y de hecho que se ha tenido que expandir a propuestas más avanzadas como la propuesta de componentes, patrones y arquitecturas que siguen teniendo su base en el modelo, pero con conceptos más adecuados a las exigencias de las nuevas tecnologías entre las cuales se debe destacar las dirigidas a la web.

REFERENCIAS

[1] Ravi Sethi, Programming Languages, Concepts and Constructs. Wilmington Delaware USA, Ed. Addison Wesley 1992.

[2] Thomas Kuhn, Estructura de las Revoluciones Científicas, Mexico, Ed. Fondo de Cultura 1975.

[3] Richar C. Lee, William M. Tepfenhart, UML and C++: A Practical Guide to Object Oriented Development, New Jersey, Ed. Prentice Hall 1997, 446 p.

BIBLIOGRAFÍA

[1] Bertran Meyer, Construcción de software Orientado a objetos, Segunda Edición, Madrid Ed. Prentice Hall 1999, 1248 p.

[2] Peter Coad y Edward Nash Yourdon, Object Oriented Analysis, Englewood Cliffs N.J, Ed. Prentice Hall 1990.

[3] Richar C. Lee, William M. Tepfenhart, UML and C++ : A Practical Guide to Object Oriented Development, New Jersey, Ed. Prentice Hall 1997, 446 p.

[4] James Runbaught, Ivar Jacobson, and Grady Booch, The Unified Modeling Language Reference Manual, USA, Ed. Addison-Wesley Object Tecnology Series 1999, 480 p.

[5] Walker Royce, Software Project Management, USA, Ed. Addison-Wesley Object Tecnology Series 1998, 448 p.

Sandro Javier Bolaños Castro
Ingeniero de Sistemas, Universidad Distrital. Candidato a Maestría en Teleinformática, Universidad Distrital. Profesor Facultad de Ingeniería, Universidad Distrital.