Aplicando la Filosofía de las Ciencias de la Complejidad a la Ingeniería del Software

David Benavides, Antonio Ruiz-Cortés y Miguel Toro

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
Avda. Reina Mercedes S/N, 41012 Sevilla,
{benavides, aruiz, mtoro}@lsi.us.es

Resumen

El método de Bacon, la filosofía cartesiana y la física de Newton han sido los pilares en los que se ha apoyado hasta la actualidad la investigación en ciencia y tecnología. No obstante, el avance de la filosofía y las ciencias de la complejidad han hecho aparecer nuevos métodos, teorías y paradigmas que mucha veces hacen inválidas las postulaciones de la ciencia clásica. En mayor parte, la ingeniería del software ha estado utilizando métodos de esta ciencia clásica ignorando aportaciones de la ciencia contemporánea como la cibernética, la visión holística o la teoría del caos. En este artículo, proponemos el uso de los métodos de las ciencias de la complejidad en ingeniería del software, poniendo ejemplos de cómo pueden ser usados algunos de estos nuevos paradigmas, ya que, en nuestra opinión pueden aportar mucho, al igual que ya lo están haciendo tanto en otras ingenierías como en ciencias fundamentales.

1. Introducción

Poco antes de su muerte Francis Bacon con su obra *Novum Organum* [2] planteó la necesidad de un método que fuese válido en todas las ramas del conocimiento y con el que fuese posible abordar cualquier inquietud del hombre con garantías de éxito. Bacon hizo una dura crítica a la filosofía griega a la que tildó de superflua por no resolver los problemas del mundo "real". Según Bacon, había que plantearse la filosofía como un instrumento para resolver problemas de la sociedad y para ello se hacía imperiosa la definición de un método universal. Gracias a las aportaciones de Bacon el pensamiento al que se puede llamar racionalista empezó a asentarse. A partir de entonces se tiene fe ciega en la razón humana como única herramienta de conocimiento verdadero. Más tarde, el matemático y filósofo francés René Descartes introdujo su filosofía cartesiana que ante todo era racionalista. Suyas son las palabras "Estoy convencido de que las matemáticas son un instrumento de conocimiento más poderosos que ningún otro que nos haya llegado por agencia humana, pues es la fuente de todas las cosas". Casi al mismo tiempo, Newton

definió lo que hoy llamamos física clásica. Con sus tres leyes revolucionó la física y aseguró que sus ecuaciones podían predecir el comportamiento del mundo, ya que lo único que hacía falta es tener un conocimiento cierto de las condiciones iniciales de un sistema dado que a partir de ellas era posible predecir el comportamiento futuro de los elementos del mismo. No hacía falta conocer el todo como tal, sino descomponerlo en partes (visión reduccionista) y unirlas más tarde para obtener verdadero conocimiento del todo. Bacon, Descartes y Newton han sido llamados los tres arquitectos de la visión mecánica del mundo [13] siendo ellos quienes han construido las tres columnas en donde la ciencia, la investigación y la técnica, en particular y la sociedad, el pensamiento y la religión occidental en general, se han apoyado hasta el momento. A principios del siglo XX empiezan a hacerse evidentes algunos aspectos que contradecían los paradigmas de la ciencia clásica. Entre ellos cabe destacar la teoría de la relatividad de Einstein [8] y el principio de incertidumbre de Heinsenberg [9] con los postulados de Popper [12] que vislumbraban el fin del determinismo. A partir de ahí, han sido muchas las aportaciones a la filosofía y a la ciencia: cibernética [16], teoría de sistemas [3], teoría del caos [6], etcétera.

La ingeniería del software es uno de los campos que ha hecho uso de los métodos de la ciencia clásica. En nuestra opinión, son todavía pocas las aportaciones que se han hecho haciendo uso de los paradigmas de la ciencia contemporánea. Sin embargo, en otras áreas de la ciencia ya han existido avances importantes hacia el uso de estos paradigmas contemporáneos. Así por ejemplo, toda la física que se viene desarrollando prácticamente prescinde de los paradigmas clásicos, aristotélicos y newtonianos sustentando sus bases en los paradigmas de las ciencia moderna [7].

Debido a la complejidad de los sistemas de información y por lo tanto de la ingeniería del software, creemos que es necesario reconducir las tendencias actuales que en mayor parte hacen uso de la filosofía de la ciencia clásica hacía el uso de la filosofía de las ciencias de la complejidad. Creemos que la ya famosa crisis del software [11] pudo deberse al reiterado uso de métodos de la filosofía de la ciencia clásica y que una posible manera de enfrentar esa crisis, la cual todavía renace cada día, sería integrando a los avances que ya se han hecho técnicas que hagan uso de los paradigmas contemporáneos. En este artículo haremos un breve repaso a algunas de las aportaciones de la ciencia contemporánea poniendo ejemplos de su posible aplicación en la construcción de sistemas de información y en ingeniería del software.

El resto del artículo está organizado de la siguiente manera. En la sección 2., presentamos lo que es la cibernética y cómo se pueden aprovechar sus aportaciones en ingeniería del software. Seguidamente, en la sección 3., repasamos el concepto de caos y su posible provecho en la construcción de sistemas de información. En las secciones 4., y 5., presentamos los conceptos de análisis holístico y atractores y apuntamos asímismo las posibles aplicaciones en sistemas de información e

ingeniería del software. Por último, en la sección 6. damos algunas conclusiones y trabajos futuros.

2. Cibernética en Ingeniería del Software

El matemático y filósofo alemán Norbert Wiener puede ser considerado como el padre de la cibernética. Él dio luz a este término que tiene como origen la palabra griega *kybernetike* (timonel). Wiener la definió como la ciencia del control y la comunicación tanto en animales como en máquinas [16]. Se puede decir que la cibernética no se preocupa de preguntar qué son los objetos, es decir, de definirlos, de observarlos como componentes separados. Por el contrario, se pregunta qué es lo que los objetos hacen y cómo lo pueden hacer, es decir, se preocupa de su comportamiento y de su relación con el medio. Una definición más cercana a lo que son los sistemas de información y la ingeniería del software es la que dio Kolmogorov [15], él definió la cibernética como la ciencia que estudia los sistemas de cualquier naturaleza capaces de recibir, almacenar y procesar información así como usarla para su control.

Para poder hacer converger uno de los aportes que la cibernética ha dado a la ciencia y al conocimiento contemporáneo con los sistemas de información y la ingeniería del software, vamos a centrarnos en las relaciones causa efecto que la cibernética redefine. En la ciencia clásica los efectos en las relaciones causa-efecto venían definidas sólo por el pasado, es decir, las relaciones causa-efecto eran lineales. Gracias a las aportaciones de la cibernética, las relaciones causa-efecto ya no sólo dependen de las causas del pasado sino también de las causas del futuro, convirtiéndose en relaciones causales que ya no son lineales sino que se basan en la retroalimentación. En el momento en el que una causa A influye sobre B, B sobre C y C nuevamente sobre A, puede decirse que se establece una relación causal no lineal, es decir, retroalimentada (figura 1)

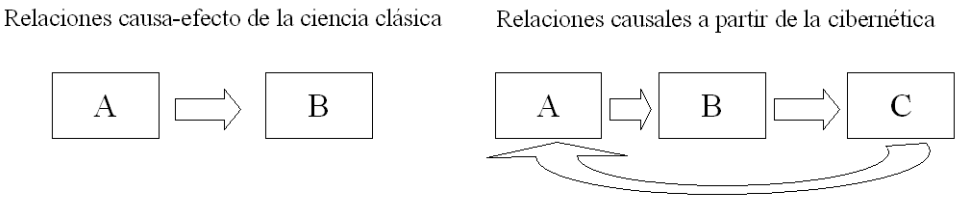

Figure 1: Diferencias en las relaciones causa-efecto en la ciencia clásica y moderna.

En ingeniería del software todavía se está haciendo uso de las relaciones causa-efecto lineales en mucho ámbitos. Un ejemplo de ello es la planificación de proyectos. Hasta el momento uno de los modelos mas extendido para la gestión de

proyectos software en cuanto a coste y tiempo ha sido el modelo COCOMO [4]. Las primeras versiones de este modelo estaban basadas en ecuaciones lineales que dadas unas entradas y aplicando una fórmula ofrecen una predicción de las salidas, en el caso de estimación de proyectos, de coste y tiempo. Ésta es una manera lineal de planificar proyectos software que tal vez pueda resultar válida para otro tipo de proyectos en los que los factores que influyen dentro del proceso pueden estar dentro de un rango de control. Sin embargo, los proyectos de desarrollo de software son procesos complejos en los que continuamente se están cambiando los requisitos y las condiciones del sistema, por lo que estos métodos lineales de planificación de proyectos pueden resultar en ocasiones ineficaces. Es habitual ver al director de un proyecto, al principio del mismo, sacar su hoja de cálculo (totalmente lineal y determinista) y empezar a hacer previsiones del mismo y decir: "tantos técnicos, mas tanto el técnico-día...: ¡Ya tengo el precio que podemos poner en la oferta!". Habría que plantearse: ¿Son válidas este tipo de técnicas lineales en la construcción de sistemas de información?. Es sensato pensar que a lo largo del ciclo de vida de un proyecto software sea necesario plantear líneas de retroalimentación que hagan posible corregir el rumbo del proyecto con el objeto de converger hacia las previsiones iniciales de coste y tiempo. En 1991 Abdel-Hamid [1] planteó un nuevo modelo para planificar proyectos de desarrollo de software que se basaba en ecuaciones diferenciales. Las relaciones existentes entre las variables que influyen en el proyecto se definieron en términos de relaciones causales con ciclos de retroalimentación. Así, dentro del desarrollo del proyecto podía haber información que modifique las previsiones iniciales de tiempo y coste. El modelado del sistema con este tipo de ecuaciones de retroalimentación hace posible que las variables se reajusten con objeto de alcanzar las previsiones iniciales [18]. Aunque teóricamente es un modelo válido son muy pocas las empresas que lo usan debido a la escasez de herramientas CASE que implementen este tipo de métodos. Por lo tanto nos planteamos: ¿Está la ingeniería del software práctica aún lejos de seguir los principios de la cibernética? ¿Y la investigación?.

3. IS, Caos y Auto-Organización

Caos: "Confusión, desorden". Ésta es una definición de caos que aparece en el diccionario de la Real Academia Española (el término caos viene del griego *Kaos,* abertura). Parece que por las costumbres y métodos utilizados tanto en ciencia, filosofía y otras facetas de la vida occidental, el caos es algo contra lo que hay que ir, es decir, siempre se ha buscado que sea el orden el que impere. Los postulados de Newton en física apostaban por un universo determinista el cual estaba ordenado por las leyes de la física y las matemáticas. El modo de entender este mundo determinista era observarlo y medirlo y sólo la escasez de acierto en estas medidas ponían límite al total conocimiento del mismo. Los descubrimientos de la física atómica demostraron que esto no era tan cierto como parecía y que a la hora de determinar por ejemplo la trayectoria de una partícula elemental con estas leyes

el objetivo se veía frustrado. Se empezó entonces a platear la existencia de un grado de desorden en la materia del Universo y dentro de los sistemas en general. Aunque parezca paradójico se definió un parámetro para medir este desorden: la entropía, a mayor entropía mayor grado de desorden. La idea del caos está muy cerca de la idea de entropía. Se llega a hablar de sistemas caóticos como aquellos en los que no existe orden o en los que la entropía es muy elevada. No obstante, es también cierto que los sistemas caóticos tienden a una cierta auto-organización que los hace funcionar correctamente. Un ejemplo de ello es el del latido del corazón. Está demostrado que el ritmo cardíaco sigue pautas caóticas. De este modo, si el ritmo del corazón se vuelve periódico y ordenado la vida del corazón está en peligro [6].

¿Podemos pensar en sistemas caóticos dentro de la ingeniería del software?. Internet es un claro ejemplo de un sistema caótico y auto-organizado a la vez. Millones de ínter nautas están conectados al mismo tiempo a través de la red con infinidad de objetivos e intenciones. Cada uno se puede considerar que actúa de un modo caótico pero sin embargo es evidente que existe una cierta organización dentro de este caos. Uno de los ejemplos más claros es el de las comunidades de código abierto. En ellas participan personas de todo el mundo desarrollando a la vez y después de un tiempo se ve como de esta auto-organización surgen productos tangibles. No obstante, se intenta evitar el caos y se lucha contra él. Todo se intenta guardar bajo control incluso en los detalles más bajos de implementación. Este es el caso de las especificaciones formales, por ejemplo, en donde se intenta que línea por línea de código este bajo control. Esto dentro de una organización compleja como puede ser una de desarrollo de software, dentro de proyectos que hagan uso de tecnologías siempre cambiantes, con personal técnico que cambia también con mucha facilidad, es muy difícil por no decir imposible de llevar a cabo. Por eso creemos que en ingeniería del software se deberían plantear otros métodos de trabajo que contemplaran la auto-organización como resultado del caos, entendiendo este como la cooperación en lugar de la competencia.

4. Análisis Holístico en Sistemas de Información

El holismo es una de las doctrinas de la filosofía contemporanea. Se puede resumir que el holismo defiende que una realidad no es igual a la suma de sus partes. Esto choca frente al análisis reduccionista de la filosofía clásica en donde se defiende que cada parte esta compuesta por partes más pequeñas y que la suma de estas forma el todo. En esta visión del mundo el análisis y estudio de las partes se hace de un modo aislado sin tener en cuenta las interrelaciones que entre los diferentes componentes del sistema existen. La visión holística por el contrario ve el todo desde el punto de vista de las relaciones entre las partes y entre el todo y las partes. Asimismo considera al todo a su vez como una parte y a las partes como un todo (nos recuerda esto a la geometría fractal [10] y a su vez a la retroalimentación).

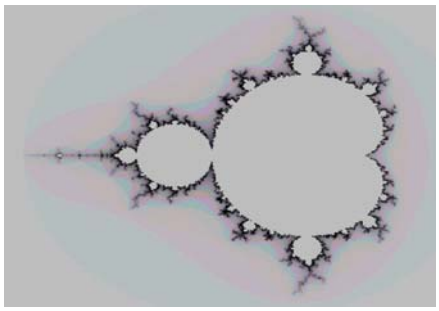

Figure 2: Conjunto de Mandlebrot

En la figura 2 se ve representado el primer fractal descubierto por Mandlebrot en el que se observa como cada parte del todo es a su vez un todo y está compuesto de partes. Otro ejemplo de fractal que se da en la naturaleza es el de una coliflor. Si cogemos una rama de la coliflor vemos como es a su vez una coliflor que esta formada por aun más pequeñas coliflores. La geometría fractal ha inspirado incluso a arquitectos urbanistas que plantean la ciudad como un conjunto fractal en donde cada parte de la misma es a su vez un todo [17].

En nuestra opinión las tendencias más modernas a la hora de construir sistemas de información se acercan más a la visión mecanicista que a la holística. Así, la investigación en el mundo de los componentes es una de las que más frutos ha dado en los últimos tiempos. Los componentes se definen como pedazos de software que desempeñan tareas concretas. Uniendo componentes se obtiene un todo que es el sistema de información. El problema surge cuando se obvian las interrelaciones que existen entre estos componentes y los resultados son muchas veces inesperados. El estudio de los modelos de componentes en sistemas de información incluyendo en ellos los modelos de clases y las librerías, no se preocupan en gran medida de las relaciones que existen entre los diferentes componentes del sistema de modo que sea posible tener una visión global del mismo viendo cada parte a su vez como un todo. No hay una manera de contemplar múltiples interrelaciones al mismo tiempo o de modelar el conjunto entero de interrelaciones del sistema de manera que podamos tener una visión global del producto que pueda ser analizado como un todo y a su vez como una parte. En este artículo no pretendemos dar un nuevo modelo para construir sistemas software, pero sí hacernos la pregunta: ¿Al construir sistemas de información, se está más cerca de la visión reduccionista que la visión hólistica?. En nuestra opinión las técnicas y la filosofía que se sigue en ingeniería del software tanto en investigación como en desarrollo son más cercanas a la filosofía de la ciencia clásica que a la filosofía de las ciencias de la complejidad.

5. Atractores y Sistemas de Información

Un atractor está definido por una zona del espacio (ya sea matemático, físico o incluso filosófico o psicológico), llamada cuenca del atractor, y por otra zona donde convergen todos los elementos que en la primera se encuentran. En un tipo de atractores, una vez que un elemento cae en la cuenca del atractor se dice que entra en un ciclo límite ,es decir, que por mucho que se haga el elemento acabará metido en el atractor. Existen infinidad de ejemplos de atractores en el Universo, uno de los más citados es el del "depredador-presa". A menor número de depredadores mayor número de presas. Con el paso del tiempo, los depredadores se empiezan a reproducir rápidamente debido a la facilidad para encontrar alimento. Una vez que se llega a un número de depredadores, son ahora las presas las que empiezan a tener dificultades para subsistir, pero a su vez esto hace que el alimento para los depredadores se haga difícil y se vuelve a generar el ciclo. En este caso se observan dos ciclos: uno en el que las presas abundan y por lo tanto los depredadores caen en el atractor de reproducirse y otro en el que lo que abundan son los depredadores y por tanto las presas tienden a desaparecer. Este tipo de atractores son los que se llaman "simples", es decir, más o menos predecibles. Existen sin embargo, los atractores llamados "extraños" que se diferencian de los anteriores por el hecho de ser imposible su predicción. Los atractores "extraños" fueron definidos por David Ruelle y Floris Takens [14] en 1989. Ruelle y Takens demostraron que por ínfimas que fuesen las variaciones en las condiciones iniciales en un sistema con atractores extraños, el resultado podía variar enormemente de una muestra a otra. Supongamos que existe un sistema que toma dos valores X e Y. Uno de ellos en una de las pruebas vale 5,001 y el otro 6,002. Éstas pueden ser las entradas de un sistema en el que el nivel de precisión que se ha tomado es de dos decimales y en el que se ha querido representar un valor de X igual a 5 y de Y igual a 6 aunque vemos que no son exactamente esos valores. En el caso de sistemas con atractores extraños la salida del es distinta, y muchas veces nada parecida, si introducimos como valor para la X 5,001 o 5,002 que para el caso de un nivel de precisión de dos decimales ambos valores se considerarían iguales . Pero eso no es todo, este tipo de sistemas con atractores extraños son sensibles a todos los cambios que se introduzcan en las entradas con lo que por mucho que se ajuste la precisión de las condiciones iniciales el valor de las salidas será impredecible. Para que la salida fuese predecible sería necesario tener una precisión en los decimales infinita lo cual es imposible y más si se trata de cálculos hechos sobre máquinas en las que como sabemos el número de decimales de precisión es siempre limitado. A partir de este descubrimiento se puede inferir que en el mundo hay partes que no son deterministas y que existe cierto nivel de caos (no necesariamente de desorganización) en el universo.

Un posible campo dentro de la ingeniería del software donde los atractores podrían ser estudiados, es de nuevo, la gestión y planificación de proyectos de desarrollo de software. Muchas veces, dentro de un proyecto software, debido a su complejidad,

nos vemos acorralados por ciclos límite provocados por atractores que incluso pueden llegar a ser atractores extraños. Así pues, nos encontramos con el caso en el que las previsiones de tiempo y coste, llegado un determinado momento en el desarrollo del proyecto (cuando se entra dentro de la cuenca del atractor), no se pueden satisfacer de ninguna de las maneras. Por muchos esfuerzos materiales o humanos que se hagan, una vez atrapados por el atractor es seguro, por ejemplo, que llegaremos tarde al final del proyecto y que el coste además sobrepasará las previsiones iniciales. Un posible trabajo de investigación que nos hemos planteado es el poder identificar estos atractores dentro de un proyecto de desarrollo de software para poder intentar bien evitarlos o bien actuar sabiendo que se está dentro de uno de ellos. En planificación de proyectos lo que se hace es planificar y hacer unas previsiones que dependan de la naturaleza del proyecto, de la madurez de la organización, etcétera, pero en ningún caso se hace un estudio de los posibles ciclos límite que a lo largo del desarrollo puedan encontrarse. Bien es cierto que los sistemas con atractores suelen aplicarse a sistemas en donde el tiempo tiende a infinito, lo cual puede ser inviable en el campo de la gestión de proyectos.

6. Conclusiones y Trabajos Futuros

En este artículo hemos presentado algunas de los paradigmas y las tendencias en las ciencias de la complejidad planteando ejemplos en los que podrían ser aplicados dentro del mundo de los sistemas de información. Creemos que la ingeniería del software podría ir acomodándose a estos paradigmas y no quedar al margen de ellos. Es por eso que en este artículo proponemos el uso de la filosofía de la ciencia contemporanea para la construcción de sistemas de información.

En [7] se presenta como la física moderna esta demostrando ahora lo que la filosofía oriental viene proclamando desde hace miles de años. Es este trabajo el que nos ha inspirado para plantear que la ingeniería del software podría evitar el seguir el camino de la física clásica, tropezandose con el pensar determinista y newtoniano y que podría coger un atajo hacia la ciencia contemporánea que le evitara el tener que renacer cuándo sus métodos quedasen obsoletos.

Como trabajos futuros planteamos la necesidad de incorporar, dentro del trabajo de tesis que vamos a seguir, la visión moderna de la ciencia de modo que en lugar de ser orientada hacía los paradigmas de la ciencia clásica, nuestra tesis pueda ser orientada hacia los paradigmas de la ciencia contemporanea dentro de lo posible. En este aspecto nos estamos planteando la posibilidad de construir un modelo de calidad orientado al producto no determinista en donde las condiciones de calidad puedan variar a lo largo del tiempo y en donde se puedan definir ciclos de retroalimentación.

Bibliografía

[1] T. Abdel-Hamid and S. E. Madnick. *Software Project Dynamics: An Integrated Approach*. Prentice-Hall, 1991.

[2] F. Bacon. Novum Organum scientiarum sive indicia vera de interpretatione naturae, 1620.

[3] L. Von Bertalanffy. *General System Theory: Foundations, Development, Applications*. George Braziller, Marzo 1976.

[4] B Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby. Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*, 1: 57–94, 1995. Software Process and Product Measurement.

[5] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Reading, 1 edition, 1998.

[6] J. Briggs and D. Peat. *Seven Life Lessons of Chaos: Spiritual Wisdom from the Science of Change*. Harper Perennial, March 2000.

[7] F. Capra. *The Tao of Physics: An Exploration of the Parallels Between Modern Physics and Eastern Mysticism*. Shambhala Publications, New York, 4th edition, Enero 2000.

[8] A. Einstein. *The Meaning of Relativity*. London, 1950.

[9] W. Heisenberg. *Physics and Philosophy: The Revolution in Modern Science (Great Minds Series)*. Prometheus Books, reprint edition edition, May 1999.

[10] B. Mandelbrot. *Fractals: Form, chance, and dimension*. W. H. Freeman, New York, 1978.

[11] B. Meyer. *Object-Oriented Software Construction*. Prentice-Hall, Englewood Cliffs, second edition, 1997.

[12] K. R. Popper. Indeterminism in quantum physics and classical physics. *British J. for the Philosophy of science*, 1: 179–188, 1951.

[13] J. Rifkin. *Entropy: A New World View*. Bantam Books, 1981.

[14] D. Ruelle. *Chaotic evolution and strange attractors: The statistical analysis of time series for deterministic nonlinear systems*. University Press, Cambridge, Septiembre 1989.

[15] Varios. *Kolmogorov in Perspective (History of Mathematics, V. 20)*. American Mathematical Society, Septiembre 2000.

[16] N. Wiener. *Cybernetics, or Control and Communication in the Animal and the Machine*. MIT Press, New York, 1948.

[17] J. Benavides. *Ideas y teorías acerca de la ciudad contemporánea*. Universidad de Sevilla, 2001.

[18] I. Ramos. *Un nuevo enfoque en la gestión de proyectos de desarrollo de software*. Tesis doctoral, Universidad de Sevilla, 1999.