Conciencia de Modelos como Instrumentos en Ingeniería de Software. Una Aproximación desde las Ciencias Naturales y Sociales.

José Miguel Cañete Valdeón*, Francisco José Galán Morillo, Miguel Toro Bonilla
E.T.S. de Ingeniería Informática. Universidad de Sevilla.

Resumen

El papel que desempeñan los modelos en la Ciencia moderna ha sido tradicionalmente un tema de gran interés para la Filosofía. Una reciente investigación realizada sobre múltiples casos de estudio tomados de la Física, la Química y las Ciencias Económicas (Morgan y Morrison, 1999) nos ofrece una perspectiva pragmática de los modelos científicos, descubriendo en ellos verdaderos *instrumentos de investigación*, útiles tanto en la elaboración de teorías como en la comprensión del mundo. Este papel consolidado y eminentemente práctico ha despertado nuestra curiosidad y nos ha motivado a estudiar los objetos que en Ingeniería de Software se denominan "modelos", provistos en nuestro estudio con el conocimiento que ofrece este nuevo ensayo sobre los modelos de la Ciencia. Nos hemos planteado las siguientes preguntas:

a) ¿Existe un concepto consolidado de "modelo" en Ingeniería de Software?
b) ¿Se utilizan modelos científicos en esta disciplina?
c) ¿Existe algún paralelismo entre los modelos de Ingeniería de Software y los de la Ciencia?
d) ¿Pueden ser considerados "instrumentos" los modelos de esta ingeniería, al igual que ocurre en la Ciencia? En caso afirmativo, ¿cuál es la funcionalidad de los mismos?

En este artículo exponemos las conclusiones a las que hemos llegado. Nuestro método de trabajo ha sido el mismo que el seguido por la investigación referida anteriormente, es decir, el estudio de distintos casos.

Palabras clave

Modelo, modelo como instrumento, modelo científico, lenguaje de modelado, Ingeniería de Software, Filosofía de la Ciencia.

1. Modelos como instrumentos de investigación sobre la realidad

Comenzamos nuestra exposición reflexionando sobre aquellos objetos encontrados en la Ingeniería de Software cuyas características coinciden con las de los modelos como instrumentos de investigación sobre la realidad descritos por Mary Morgan y Margaret Morrison (1999). En la siguiente sección estudiamos los modelos que funcionan como instrumentos para el diseño y la resolución de problemas.

Resumamos el planteamiento de Morgan y Morrison. Según estas autoras, los modelos científicos son *instrumentos de investigación* cuya utilidad se hace patente durante su uso, adoptando por tanto el carácter de una tecnología. El papel de un modelo como herramienta de investigación procede de su capacidad para representar la teoría, el mundo o ambos, lo cual nos permite *aprender* sobre lo representado. Por ello un modelo es, al mismo tiempo, fuente de conocimiento y medio para alcanzar conocimiento. Las autoras distinguen tres usos de los modelos científicos como instrumentos de investigación: (1) ayudar tanto en la construcción de nuevas teorías como en la exploración de teorías existentes; (2) permitir estructurar y mostrar mediciones sobre el mundo, así como servir de herramientas de medición y predicción; y (3) ayudar en el diseño de nuevas tecnologías y de mecanismos de intervención en el mundo.

Con este planteamiento, iniciamos nuestra búsqueda de modelos no desde la identificación de teorías científicas en Ingeniería de Software, sino desde la reflexión acerca de qué *realidades* existen en esta disciplina, susceptibles del estudio científico. A partir de esas realidades buscaremos modelos que las representen.

* Para exponer comentarios sobre este artículo, sírvase de escribir a la dirección jmcv@us.es .

Podemos hallar al menos dos, ambas de carácter muy heterogéneo. Por un lado, la realidad constituida por el entorno que rodea al software como producto: sobre la que éste debe obtener y/o tratar información, o que debe controlar, o simplemente con la que interacciona, tanto directa como indirectamente. Por otro lado, la realidad constituida por el propio proceso de desarrollo de software. Nótese que en este trabajo no consideramos el software cuando ya es una realidad existente y por tanto susceptible del estudio científico, sino como una realidad que aparece cuando se aplica un proceso de ingeniería. Trataremos esto en la siguiente sección.

Los modelos que representan la realidad constituida por el entorno que rodea al software son objetos *conceptuales* que describen ciertos aspectos de la misma, recogidos con unos determinados grados de abstracción y aproximación. Existen al menos dos formas de elaborar estos modelos: (1) a través de ontologías[1] que constituyen lenguajes, y (2) tomando estructuras matemáticas como analogías de la realidad. Un ejemplo paradigmático del primer tipo son los modelos elaborados a partir de la ontología propuesta por Peter Chen (1976) constituida, entre otros, por los elementos "Entidad" y "Relación". Dicha ontología está equipada con una notación gráfico-textual, lo que permite que los modelos como objetos conceptuales puedan ser físicamente plasmados en forma diagramática. Dichos modelos representan el aspecto estático de la realidad. Michael Jackson (2000) recoge varios ejemplos de modelos de ambos tipos; destacamos uno del segundo, consistente en un grafo de colas dinámicas que representa la realidad formada por los paquetes postales mientras discurren a través de una red de tubos y conmutadores que los conducen hacia una de las cubetas destino.

Estos modelos se comportan como instrumentos en dos sentidos. El primero coincide con lo señalado por Morgan y Morrison, ya que el modelo sirve al ingeniero para *comprender* la realidad, especialmente al comienzo del proceso de desarrollo, cuando se está analizando y estructurando el problema a resolver mediante software (Jackson, 2000, Wieringa, 2003). Desde esta perspectiva destaca el carácter simplificador de estos modelos así como su carácter predictivo. Por ejemplo, el analista podría utilizar el modelo de colas del ejemplo anterior para calcular el tiempo medio de tránsito de un paquete postal a través de la red de tuberías.

Pero hemos de notar que estos modelos también se comportan como un tipo muy distinto de instrumentos. A menudo los ingenieros los incorporan al propio software; el propósito es mantener una "copia virtual" de lo que está aconteciendo en la realidad, sincronizada con el estado de la misma, con la finalidad de obtener un acceso más rápido o conveniente a información del mundo, o para que el software pueda conocer qué está ocurriendo en el mundo (Jackson, 2000).

Consideremos ahora la realidad constituida por el proceso de desarrollo software. En este caso encontramos modelos como instrumentos en el sentido de Morgan y Morrison, siendo el objetivo *predecir* distintos aspectos de dicho proceso. Un ejemplo paradigmático son los llamados "modelos de estimación empírica", que tienen como objetivo predecir el esfuerzo que va a ser necesario en cada etapa del proyecto antes de que la etapa comience (Pressman, 2000). Otro ejemplo, el denominado "*Capability Maturity Model*" (Paulk y otros, 1993), trata de predecir el grado de madurez del proceso de desarrollo de software de una empresa; además, el modelo propone actividades a realizar de acuerdo con el grado de madurez pronosticado.

Como vemos, los objetivos de los modelos estudiados hasta ahora coinciden con la función señalada por Morgan y Morrison de servir de instrumentos de ayuda en el

[1] Esperanza Marcos y Alfredo Marcos (2001) realizan una interesante reflexión acerca de si las ontologías utilizadas en el área de las bases de datos (los llamados "modelos de datos") pueden ser consideradas como modelos científicos, desde las perspectivas de las Ciencias Empíricas y de la Lógica Matemática.

diseño de tecnología y en el diseño de mecanismos de intervención en el mundo, al aportar al ingeniero el tipo de información que necesita: en el primer caso, sobre el mundo donde va a intervenir introduciendo tecnología en forma de software; en el segundo, sobre la realidad que constituye el proceso de software y donde va a intervenir mediante decisiones estratégicas de planificación.

2. Modelos como instrumentos para el diseño y la resolución de problemas

Un tipo diferente de modelos, empleados en la Ciencia y, con más frecuencia, en toda ingeniería (Giere, 1997) son los *modelos a escala*. En esta sección nos referiremos a estos modelos en el sentido de la ingeniería, es decir, como representaciones de un objeto que aún no existe pero que se desea construir. El objetivo de los mismos no es explicar la realidad, sino predecir las propiedades que tendrá un producto antes de comenzar a construirlo. Por ello se elaboran durante la etapa de diseño.

Los productos que se diseñan en la Ingeniería de Software son de naturaleza heterogénea y conceptual: los requisitos, la arquitectura, los componentes, el plan de pruebas, etc. En este artículo nos referiremos a los mismos genéricamente como "el producto". Debido al carácter conceptual de éste, el equivalente en Ingeniería de Software a los modelos a escala lo encontramos en aquellos objetos *conceptuales* que constituyen *aproximaciones* al producto, con distintos grados de incertidumbre, diversos niveles de abstracción, y desde distintas perspectivas. Por lo tanto, podemos afirmar que el producto *es* aquel modelo con un grado de aproximación máximo: mínimas incertidumbre y abstracción, y que abarca todas las perspectivas. Al igual que los modelos a escala, los modelos como aproximaciones permiten predecir propiedades del producto, tengan el grado de aproximación que tengan. Roel Wieringa (2003) destaca ésta sobre otras utilidades de estos modelos. Las predicciones que se obtienen son asertos sobre propiedades del producto, derivables a partir de las decisiones de diseño implícitas en el modelo. Wieringa señala varios esquemas de razonamiento para realizar esta derivación de propiedades. Queremos resaltar, además de la capacidad predictiva de estos objetos, su utilidad como instrumentos de aproximación gradual hacia el producto deseado, gracias a sus capacidades de enfoque y abstracción, y de servir como contenedores de incertidumbre. Ambas cualidades convierten a estos modelos en herramientas indispensables en cualquier ingeniería.

Al igual que ocurre con algunos de los modelos descritos en la sección anterior, los modelos como aproximaciones a un producto se elaboran a través de ontologías conceptuales que constituyen lenguajes de modelado. Éstos tienen una amplísima tradición en Ingeniería de Software, pero destacamos, por su popularidad en la comunidad de desarrolladores, la propuesta unificadora realizada por James Rumbaugh, Ivar Jacobson y Grady Booch (1999) denominada *Unified Modeling Language* (UML).

Como vemos, los "modelos a escala" en Ingeniería de Software son objetos conceptuales a modo de aproximaciones hacia los productos a diseñar, que a su vez son también objetos conceptuales (como indicábamos antes son los requisitos, la arquitectura, etc). Sin embargo, hemos hallado ciertos modelos de este tipo cuyo objetivo principal no es el de aproximarse a un producto. De hecho, a veces el producto conceptual que aproximan ni siquiera tiene interés por sí mismo. Lo común de estos modelos es que, junto con otros elementos, constituyen *entidades* conceptuales superiores cuyo objetivo es *aconsejar* al ingeniero acerca de cómo acometer un diseño o cómo resolver un misterio. Como ejemplo, consideremos los modelos que se elaboran a partir de una de las ontologías del lenguaje UML, denominados "modelos de casos de uso". Contemplados de manera aislada, son modelos aproximativos hacia un producto:

los requisitos funcionales del sistema informático. Pero, junto con una porción de la metodología denominada *Unified Process* (UP –Jacobson, Booch y Rumbaugh, 1999), constituyen una entidad pensada para aconsejar al ingeniero a diseñar la arquitectura esencial del sistema. La metodología contiene guías para diseñar la arquitectura a partir de la estructura del modelo de casos de uso. La ontología a partir de la que se elabora el modelo ha sido especialmente pensada para que éste sirva como *instrumento de creación*, ya que en él se basan los razonamientos implícitos en la metodología para proponer consejos de diseño arquitectural. La lógica seguida es sencilla, y se basa en la experiencia de los autores de UP; se puede resumir así: (1) en la ingeniería de sistemas informáticos conocidos con modelos de casos de uso de estructura *M* fue útil aplicar las decisiones *A* sobre diseño arquitectural; (2) como en este caso particular ha aparecido un modelo con estructura *M*, entonces (3) una posible arquitectura para un sistema informático que cumpla los requisitos probablemente resulte de aplicar las decisiones *A*.

Un ejemplo más complejo se encuentra en la propuesta de Jackson (2000). El objetivo es aconsejar al ingeniero en la tarea de analizar un problema de desarrollo software, incluyendo, entre otros aspectos, descubrir de cuántos subproblemas consta, diseñar subproblemas adicionales cuya introducción puede ser útil para aliviar otros, y prever las dificultades con las que se va a encontrar en el estudio de los subproblemas. La entidad conceptual propuesta por Jackson para este fin es heterogénea; entre sus elementos, incluye una ontología para elaborar modelos llamados "*problem diagrams*". El objetivo de los mismos es permitir al ingeniero capturar los elementos clave del problema, a modo de repositorio de "pistas", sobre las que operan los demás componentes de la entidad. Uno de los usos de estos modelos es servir de base a unos esquemas de razonamiento (denominados "argumentos de corrección") que sirven para verificar si la comprensión del problema por parte del ingeniero es correcta.

3. Breve conclusión

En Ingeniería de Software existen tanto modelos científicos para investigar la realidad que le concierne, como modelos que se aproximan a un producto a diseñar, al tiempo que predicen sus propiedades y, que, a veces, forman parte de entidades conceptuales superiores que aconsejan en el diseño y la resolución de problemas. Identificando estos conceptos, hemos tratado de contribuir a crear una conciencia de modelos como instrumentos en esta disciplina.

Referencias

Chen, P. (1976). The Entity-Relationship Model – Toward a Unified View of Data. *ACM Transactions on Database Systems*, Vol. 1, No. 1. Marzo 1976, pgs. 9-36.

Giere, R. (1997). *Understanding Scientific Reasoning*. Cuarta edición. Harcourt Brace College Publishers.

Jackson, M. (2000). *Problem Frames. Analyzing and structuring software development problems.* ACM Press, Addison-Wesley.

Jacobson,I., Booch,B., Rumbaugh,J.(1999). *The Unified Software Development Process*.Addison-Wesley.

Marcos, E. y Marcos, A. (2001). A Philosophical Approach to the Concept of Data Model: Is a Data Model, in Fact, a Model? *Information Systems Frontiers*, 3:2, 267-274.

Morgan, M. y Morrison, M. (1999). Models as mediating instruments. En "*Models as Mediators. Perspectives on Natural and Social Science*", pgs. 10–37. Morgan, M. y Morrison, M. (Editores). Cambridge University Press.

Paulk, M., Curtis, B., Chrissis, M.B., y Weber, C. (1993). Capability Maturity Model for Software, Version 1.1. Technical Report CMU/SEI-93-TR-024, ESC-TR-93-177.

Pressman, R. (2000). *Software Engineering. A Practitioner's Approach.* McGraw Hill.

Rumbaugh, J., Jacobson, I., Booch, G. (1999). *The Unified Modeling Language Reference Manual*. Addison-Wesley.

Wieringa, R. (2003). *Design Methods for Reactive Systems. Yourdon, STATEMATE and the UML.* Morgan Kaufmann.