Scientia Et Technica
ISSN: 0122-1701
scientia@utp.edu.co
Universidad Tecnológica de Pereira
Colombia

SERNA M., EDGAR
LA IMPORTANCIA DE LA ABSTRACCIÓN EN LA INFORMÁTICA
Scientia Et Technica, vol. XVI, núm. 48, agosto, 2011, pp. 122-126
Universidad Tecnológica de Pereira
Pereira, Colombia

Disponible en: http://www.redalyc.org/articulo.oa?id=84922622022

Cómo citar el artículo
Número completo
Más información del artículo
Página de la revista en redalyc.org

Sistema de Información Científica
Red de Revistas Científicas de América Latina, el Caribe, España y Portugal
Proyecto académico sin fines de lucro, desarrollado bajo la iniciativa de acceso abierto

LA IMPORTANCIA DE LA ABSTRACCIÓN EN LA INFORMÁTICA

The importance of abstraction in the informatics

RESUMEN
Es un hecho que algunos ingenieros de software y científicos computacionales son capaces de producir diseños y programas claros y elegantes, mientras que otros no pueden. Acaso, ¿será esto cuestión de inteligencia? ¿Será posible mejorar en los estudiantes estas aptitudes y habilidades mediante formación y entrenamiento? En este trabajo se exploran respuestas a estas preguntas y se argumenta que para los profesionales y estudiantes de informática es crucial que posean una buena comprensión de la abstracción.

PALABRAS CLAVES: Abstracción, Ciencias Computacionales, informática, Ingeniería de Software.

EDGAR SERNA M.
Ingeniero de Sistemas, Ms.C
Profesor auxiliar
Facultad de Ingenierías
Fundación Universitaria Luis Amigó
edgar.sernamo@amigo.edu.co

ABSTRACT
It is a fact that some software engineers and computer scientists can develop clear and elegant software programs and designs, in contrast to those who are unable to do that. Could it be a matter of intelligence? It would be possible to improve these skills and aptitudes in the students by means of training and education? In this work we explore answers to these questions and we propose that a good comprehension level on abstraction is crucial for both professional and students belonging to computer science.

KEYWORDS: *Abstraction, computing, Computer Science, Software Engineering.*

1. INTRODUCCIÓN

Por más de veinte años, he estado involucrado en la investigación, formación y difusión de las ciencias computacionales y la ingeniería de software. Mi experiencia en formación abarca cursos como principios de la programación, arquitectura de sistemas, arquitectura de software, ingeniería de software, sistemas de información, algoritmos distribuidos y diseño de software. Cursos que requieren de los estudiantes habilidades para analizar, conceptualizar, modelar y resolver problemas. La experiencia en estos años me ha permitido concluir que los estudiantes que sobresalen en informática son capaces de manejar la complejidad de los problemas para producir modelos y diseños elegantes, y de hacer frente a la complejidad de los algoritmos distribuidos, a la aplicabilidad de diversas notaciones de modelado, y otras cuestiones importantes en esta área.

Por otro lado, existen otros estudiantes que no sobresalen tanto como aquellos: ya que encuentran que los algoritmos distribuidos son muy difíciles, no aprecian la utilidad del modelado, tienen dificultades para identificar lo que es importante en un problema, y producen soluciones complicadas que replican las complejidades mismas del problema. ¿Por qué? ¿Qué es lo que hace que algunos estudiantes puedan? ¿Qué les falta a los que no pueden? ¿Es alguna cuestión de inteligencia? Creemos que la clave está en la abstracción, es decir, en la capacidad para realizar y aplicar pensamiento abstracto y poseer habilidades en abstracción.

En este artículo se explora esta hipótesis y se formulan recomendaciones para trabajos futuros. En primer lugar, se discute qué es la abstracción y su papel en la informática y otras disciplinas; se utilizan los resultados del desarrollo cognitivo, y se analizan los factores que afectan la capacidad de los estudiantes para hacer frente a la abstracción y para aplicarla. Luego, se discute si la abstracción es o no enseñable; y, finalmente, se sugieren los pasos necesarios para poner a prueba las habilidades en abstracción, como medio para validar la hipótesis planteada, revisar la actuales técnicas de enseñanza, e incluso, tal vez, seleccionar los estudiantes mejor capacitados.

2. DEFINICIÓN E IMPORTANCIA DE LA ABSTRACCIÓN

De las diversas definiciones de abstracción [1], nos centramos en dos aspectos particularmente pertinentes [2]: el primero hace hincapié en el proceso de *eliminar detalles* para simplificar y concentrar la atención con base en:

Fecha de Recepción: 13 de Mayo de 2011
Fecha de Aceptación: 22 de Julio de 2011

- El acto de retirar o remover algo
- El acto o proceso de no considerar una o más propiedades de un objeto complejo a fin de atender las demás.

El segundo hace hincapié en el proceso de *generalización* para identificar el núcleo común o esencial con base en:

- El proceso de formulación general de conceptos para abstraer propiedades comunes de las instancias
- Un concepto general formado por la extracción de características comunes a partir de ejemplos específicos.

La abstracción es ampliamente utilizada en otras disciplinas como el arte, la escritura y la música. Por ejemplo, es famosa la pintura de Henri Matisse, "*Naked blue IV*", en la que logra representar con claridad la esencia de su tema, una mujer desnuda, utilizando sólo líneas simples o recortes. Su representación elimina todos los detalles, pero transmite mucho; del mismo modo, Katsushika Hokusai en su pintura "*South Wind, Clear Sky*", utiliza un equilibrio perfecto de color y composición representando una forma abstracta de la montaña para capturar su esencia.

Otro ejemplo es el jazz, en el que los músicos identifican las melodías esenciales o el corazón de la pieza de música en particular, e improvisan a su alrededor, de tal forma que proporcionan sus propios adornos. De acuerdo con los músicos de jazz, es fácil hacer complejo a un sonido simple, pero es más difícil hacer simple a un sonido complejo. Esta dificultad es un claro ejemplo del desafío en la aplicación de la abstracción para eliminar detalles superfluos.

Un maravilloso ejemplo de la utilidad de la abstracción lo proporciona Harry Beck [3], en su mapa del metro de Londres: en 1928 el mapa era esencialmente una superposición de la red del metro sobre un mapa geográfico convencional de Londres; se mostraban las curvas de las líneas de tren y las del río Támesis, y las distancias relativas entre las estaciones. En 1931, Beck produjo la primera representación abstracta y esquemática: simplificó las curvas a sólo líneas horizontales, verticales y diagonales, donde las distancias entre las estaciones ya no eran proporcionales a las distancias geográficas. Esta forma de representación simplificada, o abstracción, resultó ser tan adecuada para navegar alrededor del metro de Londres que todavía se utiliza, y se ha utilizado para sistemas de transporte en muchos otros países. El nivel de abstracción utilizado fue cuidadosamente seleccionado a fin de incluir sólo los detalles necesarios, pero abandonando los innecesarios: si es demasiado abstracto, el mapa no proporciona suficiente información para este propósito; y demasiado detallado, se vuelve confuso e incomprensible. Al igual que cualquier abstracción, puede ser engañosa si se utiliza para otros propósitos: este mapa del metro a veces es mal utilizado por los turistas, ya que lo malinterpretan como a un verdadero mapa geográfico de Londres. El nivel, los beneficios y el valor de una abstracción en particular dependen de sus propósitos.

¿Por qué es importante la abstracción en las Ciencias Computacionales y en la Ingeniería de Software? El software en sí ciertamente es abstracto, y la disciplina de desarrollo de software requiere habilidades de abstracción. Devlin [4] señala de manera clara y concisa: "*Una vez que te das cuenta de que la informática tiene que ver con la construcción, manipulación y razonamiento acerca de abstracciones, se hace evidente que un pre-requisito importante para la buena escritura de programas de computador es la capacidad para manejar abstracciones de manera precisa*". Wing [5] confirma la importancia de la abstracción en el pensamiento computacional, haciendo hincapié en la necesidad de pensar en múltiples niveles de abstracción. Ghezzi *et al*. [6] identifican a la abstracción como uno de los principios fundamentales de la ingeniería de software, para poder dominar la complejidad. Autores como Hazzan [7], también han discutido la abstracción como un pilar básico para las matemáticas y la computación. La eliminación de detalles innecesarios es evidente en la ingeniería de requisitos y en el diseño de software.

La elicitación de requisitos consiste en identificar los aspectos críticos del entorno que requiere el sistema, mientras se abandonan los irrelevantes. El diseño requiere que se evite la implementación innecesaria de restricciones. Por ejemplo, en el diseño de un compilador a menudo se emplea una sintaxis abstracta para centrarse en las características esenciales de la construcción del lenguaje, y se diseña el compilador para producir código intermedio para una máquina abstracta idealizada, para mantener la flexibilidad y evitar la innecesaria dependencia de la misma. El aspecto de generalización de la abstracción se puede ver claramente en el desarrollo, en el uso de abstracciones de datos y de clases en la programación orientada por objetos. La interpretación abstracta para analizar el programa es otro ejemplo de generalización, donde el dominio del programa concreto se asigna a un dominio abstracto para capturar la semántica computacional para analizar el programa.

Las habilidades de la abstracción son esenciales en la construcción de modelos, diseños e implementaciones apropiadas, que son aptas para el propósito particular que nos ocupa. El pensamiento abstracto es fundamental para manipular y razonar sobre abstracciones, ya sean modelos formales para el análisis o programas en un lenguaje de programación. De hecho, la abstracción es fundamental para las matemáticas y la ingeniería en general, jugando un papel crítico en la producción de modelos para el análisis y en la producción de soluciones de ingeniería de sonido.

3. LAS CAPACIDADES DE LOS ESTUDIANTES DE INFORMÁTICA

¿De qué competencias en abstracción dependen nuestros estudiantes para su desarrollo cognitivo? ¿Podemos mejorar sus capacidades y, en caso afirmativo, cómo? ¿Es posible enseñar las habilidades del pensamiento abstracto y la abstracción?

Jean Piaget [8, 9] sentó las bases para una comprensión del desarrollo cognitivo de los niños, desde que son bebés hasta la edad adulta. Con base en estudios de caso, derivó cuatro etapas para el desarrollo: senso-motriz, pre-operacional, operacional concreta y operacional formal. Las primeras dos fases van desde la infancia hasta la primera infancia, cerca de los siete años, y es donde indica, más o menos, la inteligencia mediante actividades motrices, y luego con el lenguaje y la manipulación temprana de símbolos. La tercera es la etapa operacional concreta, entre los siete y doce años, y es donde indica, más o menos, la inteligencia mediante una comprensión de la conservación de la materia, de la causalidad, y una habilidad para clasificar objetos concretos. La cuarta es la etapa operacional formal, alrededor de los doce años hasta la edad adulta, y es donde los individuos indican una habilidad para pensar abstracta, sistemática e hipotéticamente, y utilizan símbolos relacionados con conceptos abstractos. Esta es la etapa crucial en la que el individuo es capaz de pensar abstracta y científicamente.

Aunque existen algunas críticas acerca de la forma como Piaget realizó sus investigaciones y derivó su teoría, existe un apoyo general para sus ideas fundamentales. Además, estudios y evidencias experimentales apoyan la hipótesis de Piaget de que los niños progresan a través de las tres primeras etapas de desarrollo; sin embargo, parece que no todos los adolescentes progresan hasta la etapa operacional formal a medida que maduran. El desarrollo biológico puede ser un pre-requisito, pero pruebas realizadas en poblaciones de adultos indican que sólo entre el 30 y el 35% de los adultos alcanzan la etapa operacional formal [10], y que condiciones particulares del medio ambiente y de formación pueden ser necesarias para que tanto adolescentes como adultos alcancen esta etapa.

4. LA ENSEÑANZA DE LA ABSTRACCIÓN

A pesar de que el nivel de exigencia para alcanzar la fase operacional formal de Piaget es bajo, puede más bien ser decepcionante, ya que no parece haber alguna esperanza de poder mejorar el rendimiento de los estudiantes mediante la creación de un ambiente educativo adecuado. Por ejemplo, Huitt and Hummel [9] basados en Woolfolk and McCune-Nicolich [11], recomiendan para adolescentes utilizar técnicas de enseñanza como la de darles la oportunidad de explorar muchas cuestiones hipotéticas: animarlos a explicar cómo resuelven problemas, y enseñarles conceptos generales en lugar de sólo hechos.

¿Qué pasa con los contenidos curriculares y los planes de estudio? Es recomendable que, en cualquier pregrados en informática, se ofrezcan módulos de cursos diferentes, entre los que se incluya un número adecuado de especialización, que sean opcionales. Ninguno de esos cursos debe ser sobre abstracción, sin embargo, ¡todos deben depender de o utilizar la abstracción para explicar, modelar, especificar, razonar o resolver problemas! Esto podría confirmar que la abstracción es un aspecto esencial de la informática, pero que tiene que enseñarse indirectamente, a través de otras temáticas.

Nuestra experiencia es que las matemáticas son un excelente vehículo para enseñar el pensamiento abstracto. En los primeros años de algunos pregrados, cuando los planes de estudio tienen menos contenido matemático, parece que a los estudiantes les hicieran falta habilidades de abstracción y son menos capaces de lidiar con problemas complejos. Devlin [4] confirma esta tesis al subrayar: "*El principal beneficio de aprender y utilizar matemáticas no son los contenidos específicos, sino el hecho de que se desarrolla la capacidad para razonar precisa y analíticamente acerca de estructuras abstractas definidas formalmente*". El movimiento en favor de un tratamiento matemático de la informática y la inclusión de tópicos matemáticos en el currículo es muy fuerte. Sin embargo, en informática, no sólo es fundamental que los estudiantes sean capaces de manipular formalismos simbólicos y numéricos, también es necesario que tengan habilidades para pasar del mundo real, informal y complicado, a un modelo abstracto simplificado.

El currículo en informática de la ACM [12], reconoce la importancia de la abstracción mediante la inclusión de aspectos como encapsulamiento, niveles de abstracción, generalización y clases de abstracción; sin embargo, es el modelado y el análisis del software los que reciben mayor atención.

La modelización y el análisis formal son un poderoso medio para la práctica del pensamiento abstracto y la consolidación de la capacidad de los estudiantes para aplicar la abstracción. El modelado es la técnica de ingeniería más importante: los modelos nos ayudan a comprender y analizar los problemas grandes y complejos. Dado que los modelos son una simplificación de la realidad con el propósito de promover la comprensión y el razonamiento, los estudiantes deben ejercitar todas sus capacidades de abstracción para construir modelos que sean idóneos para un propósito. También deben ser capaces de trazar mapas entre la realidad y la abstracción, a fin de que puedan apreciar las limitaciones de ésta última, y de interpretar las implicaciones del análisis del modelo. La motivación del estudiante se puede mejorar mediante la presentación matemática de los formalismos del modelado, pero de

forma problematizadora; de tal forma que puedan beneficiarse de la disposición de herramientas de soporte ─como las de comprobación de modelos─ para el razonamiento y el análisis.

La experiencia personal me ha demostrado que la construcción de modelos y análisis como parte de un curso en concurrencia [13], pueden ser muy alentadoras. Dado un modelo, los estudiantes manifiestan mayor interés por clarificar los aspectos importantes del problema, y por razonar acerca de sus propiedades y comportamiento. Sin embargo, a algunos todavía les resulta extremadamente difícil construir los modelos desde el principio. No es suficiente pensar en lo que desean modelar, necesitan pensar acerca de cómo ─el propósito─ van a utilizar ese modelo. Aunque son capaces de pensar y razonar en abstracto, estos estudiantes parecen carecer de las habilidades necesarias para aplicar la abstracción.

5. UNA SOLUCIÓN

Si la abstracción es una habilidad clave para la informática, ¿en qué se deberían centrar los procesos de enseñanza en informática para asegurar que sea efectiva, y para que los profesionales tengan adecuados conocimientos en abstracción?

Hasta el momento hemos presentado situaciones principalmente anecdóticas, con alguna evidencia soportada en la literatura, pero ¿cómo sustentar esto con una base más científica para mejorar nuestra comprensión de la situación? Como en todas las actividades científicas e ingenieriles, antes de que podamos controlar o efectuar, primero debemos medir. El objetivo es reunir los siguientes datos:

1. Mientras estén en la universidad, medir anualmente las capacidades en abstracción de los estudiantes. Esto se podría utilizar para comprobar si su habilidad se correlaciona con sus calificaciones, y relacionarlo con los resultados de sus compañeros de semestre. Suponiendo que nuestras técnicas de clasificación convencional ─trabajos de cursos, trabajos de laboratorio y evaluaciones─ son indicativas de la capacidad de un estudiante en informática, esto ayudaría a ganar confianza en que la abstracción es un indicador clave de capacidad. Un segundo objetivo de estas pruebas es que proporcionarían un medio alternativo para revisar las capacidades del estudiante. Por último, también podría ayudar a evaluar la eficacia de nuestras técnicas de enseñanza, y a asegurar que las capacidades de todos los estudiantes mejoren a medida que progresan en su carrera.

2. Medir las capacidades en abstracción de los estudiantes al momento que tramiten su inscripción para estudiar informática. Actualmente, el ingreso a las universidades de los estudiantes se sustenta casi que exclusivamente en las calificaciones escolares previas. La capacidad en abstracción podría potencialmente utilizarse para ayudar a detectar a aquellos estudiantes que no son competentes o que tienen menor probabilidad de lograr un adecuado desempeño en sus estudios de informática, y seleccionar a aquellos que, además de ser académicamente competentes, también tienen una aptitud real para la informática y la ingeniería de software.

La realización de estos experimentos y recolección de estos datos depende de la disponibilidad de *buenas pruebas en abstracción* para medir el pensamiento y las capacidades de los estudiantes acerca de la misma. Desafortunadamente, no hemos podido detectar la existencia de pruebas apropiadas. Las pruebas para la etapa operacional formal se enfocan principalmente en el razonamiento lógico, y no son apropiadas para probar capacidades en abstracción ni para distinguir entre las capacidades de los estudiantes de un nivel universitario. Dubinsky and Hazzan [14] recomendaron diseñar un conjunto específico de preguntas de prueba, incluyendo suficientes y diferentes tipos de tareas y descripciones, soportadas en la recolección de datos cuantitativos y cualitativos e incluyendo preguntas y entrevistas no limitadas de antemano. Estas pruebas deben examinar las diferentes formas y niveles de abstracción y los diferentes propósitos para esas abstracciones. Este debe ser nuestro siguiente paso. Sólo entonces podremos tener datos definitivos en cuanto a la criticidad de la abstracción en la informática, y de nuestra habilidad para enseñarla. Por ejemplo, deberíamos ser capaces de confirmar o refutar que un curso en particular, tales como el modelado y el análisis formal, es realmente un medio efectivo para enseñar abstracción.

6. CONCLUSIONES

Al igual que muchos, creemos que la abstracción es una habilidad clave para la informática; que es esencial, en la ingeniería de requisitos, para elicitar los aspectos críticos del entorno que requiere el sistema, mientras descuidamos los no importantes. En el diseño, necesitamos articular la arquitectura del software y la funcionalidad de los componentes, de tal forma que satisfagan los requisitos funcionales y no funcionales, mientras evitamos innecesarias restricciones de implementación. Incluso, en la fase de implementación, utilizamos la abstracción de datos y de clases con el fin de generalizar las soluciones.

En este trabajo hemos propuesto que la razón por la que algunos ingenieros de software y científicos computacionales son capaces de producir diseños y programas claros y elegantes, mientras que otros no son capaces, se puede atribuir a sus capacidades de abstracción. Argumentamos que una buena comprensión del concepto de la abstracción y su importancia en la

ingeniería de software, y la forma de enseñar y examinar las capacidades en abstracción, es crucial para el futuro de las Ciencias Computacionales. Lo primero que se necesita es un conjunto de pruebas en abstracción para revisar el progreso del estudiante, que permitan comprobar nuestras técnicas de enseñanza, y potencializarlas como una ayuda para seleccionar estudiantes en los procesos de admisión.

AGRADECIMIENTOS

El autor desea manifestar sus agradecimientos a la Fundación Universitaria Luis Amigó por su colaboración para la realización de la investigación en la que se originó este trabajo.

REFERENCIAS

[1] Webster, *Third New International Dictionary of the English Language*. London: Merriam-Webster, 2002.

[2] P. Frorer, O. Hazzan, and M. Manes, "Revealing the faces of abstraction", *International Journal of Computers for Mathematical Learning*, No. 2, pp. 217-228, Jul. 1997.

[3] BBC News (2006). Top three "iconic" designs named. [Online]. Available: http://news.bbc.co.uk/2/hi/uk_news/england/london/4769060.stm

[4] K. Devlin, "Why universities require computer science students to take math", *Communications of ACM*, Vol. 46, No. 9, pp. 37-39, Sept. 2003.

[5] J. M. Wing, "Computational thinking", *Communications of ACM*, Vol. 49, No. 3, pp. 33-35, Mar. 2006.

[6] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*. New Jersey: Pearson International, 2003.

[7] O. Hazzan, "Reducing abstraction level when learning abstract algebra concepts", *Educational Studies in Mathematics*, No. 40, pp.71-90, May 1999.

[8] J. Piaget, and B. Inhelder, *The Psychology of the Child*, London: Routledge & Kegan Paul, 1969.

[9] W. Huitt, and J. Hummel, *Piaget's theory of cognitive development: Educational Psychology Interactive*. Valdosta USA: Valdosta State University, 2003.

[10] D. Kuhn, J. Langer, L. Kohlberg, and N. S. Haan, "The development of formal operations in logical and moral judgment", *Genetic Psychology Monographs*, No. 95, pp. 97-188, Feb. 1977.

[11] A. E. Woolfolk, and L. McCune-Nicolich, *Educational Psychology for Teachers*. New Jersey: Prentice-Hall, 1984.

[12] ACM, AIS, IEEE-CS, *The ACM/IEEE Computing Curricula*, Washington: ACM & IEEE Press, Sept. 2005.

[13] J. Magee, and J. Kramer, *Concurrency - State Models and Java Programs*, Chichester UK: John Wiley & Sons, 2006.

[14] Y. Dubinsky, and O. Hazzan, "Shaping of a teaching framework for software development methods at higher education", Proceedings of The International Workshop on Learning and Assessment in Science, Engineering & Management in Higher Education. Technion - Israel Institute of Technology, Haifa, Israel, pp. 59, Oct. 2004.