Edad		Peso/kg		Estatura/cm	
Coeficiente de asimetría	1,89375412	Coeficiente de asimetría	0,7241226	Coeficiente de asimetría	0,26662081
Rango	42	Rango	69	Rango	48
Mínimo	15	Mínimo	31	Mínimo	142
Máximo	57	Máximo	100	Máximo	190
Suma	23219	Suma	61385	Suma	167273
Cuenta	1083	Cuenta	1014	Cuenta	1011

En una entrega posterior se presentarán ejemplos resultantes de la aplicación de otras rutas que permiten, entre otras cosas, el agrupamiento de datos, el cálculo de estudios de correlación entre dos variables, y la aplicación de análisis de varianza para uno y dos factores (Hopkins, Hopkins, & Glass, 1997).

LISTA DE REFERENCIAS

Hopkins, K. D., Hopkins, B., & y Glass, G. (1997). *Estadística básica.* México: Prentice-Hall.

Microsoft. (2007). *Excel bajo Windows.*

Spiegel, M. R. (1993). *Estadística.* Madrid: Mc Graw Hill.

Walpole, R. E. (1999). *Probabilidad y estadística para ingenieros*. México: Prentice-Hall.

ARTÍCULO IV

AVANCES EN LA GENERACIÓN AUTOMÁTICA DE CÓDIGO A PARTIR DE ESQUEMAS PRECONCEPTUALES

Carlos Mario Zapata Jaramillo, Ph. D.
Líder del Grupo de Investigación
en Lenguajes Computacionales
Escuela de Sistemas
Universidad Nacional de Colombia Sede Medellín

RESUMEN

Pese al desarrollo actual de las herramientas *Computer-Aided Software Engineering* (CASE) y a la existencia de múltiples métodos de desarrollo de software, la generación automática de código ejecutable y funcional sigue siendo una de las promesas incumplidas de la Ingeniería de Software. A las dificultades normales que exhiben las herramientas CASE, que generan sólo parcialmente el código desde esquemas conceptuales difíciles de entender para los interesados, se suma el hecho de que estos comunican de manera vaga e imprecisa sus necesidades y expectativas. Por las razones anteriores, en este artículo se hace un compendio de los avances en la generación automática de software tomando como punto de partida los denominados esquemas preconceptuales, que son representaciones gráficas del discurso del interesado para facilitar la comunicación y la validación del dominio del problema desde fases preliminares del ciclo de vida del software.

Palabras clave: Esquemas preconceptuales, Herramientas CASE, Reglas heurísticas, Generación automática de código fuente.

INTRODUCCIÓN

Las herramientas CASE se crearon, principalmente, para asistir a los analistas en labores como el trazado de diagramas, la ingeniería inversa y la generación de código. En esta última tarea, herramientas de tipo comercial como Rational Rose® (IBM Corporation, 2010), Together™ (BORLAND Software Corporation, 2010) y Poseidon® (Gentleware, 2010) y otras de software libre, como ArgoUML® (Tigris.org, 2010) suelen generar sólo una parte del código, que generalmente se asocia con la definición de clases, atributos y el encabezado de los métodos en varios lenguajes de programación. Si bien esta actividad reemplaza otras de tipo manual que deben realizar los analistas en ausencia de herramientas CASE, los esfuerzos aún son insuficientes y la promesa de generación de código completamente funcional todavía sigue sin tener respuestas concretas. Por otra parte, los diferentes métodos de desarrollo de software, ya sean basados en planes o ágiles, presentan la misma limitación en cuanto a la generación automática de código, con el agravante de que muchas de ellas, incluso, no poseen un esquema de consistencia que permita realizar traducciones sucesivas de los modelos de requisitos hasta llegar al código ejecutable.

La situación se agrava aún más cuando se toma en consideración que los interesados, principal fuente de información para el levantamiento de los requisitos de una aplicación, suelen expresar sus necesidades y expectativas en lenguaje ambiguo y con problemas de precisión. Esta situación la reflejan Zapata y Olaya (2007) en forma de cómic, como se muestra en la figura 12.

Figura 12. Parodia de la expresión de necesidades y expectativas del interesado (tomado de Zapata & Olaya, 2007).

A esta situación, los desarrolladores responden con una necesidad adicional, que también muestran en forma de cómic Zapata y Olaya (2007) en la figura 13. En este caso, es el deseo de los desarrolladores de que el software sea como un kit prediseñado y listo para usar, sin mediar proceso alguno.

Figura 13. Parodia del deseo de los desarrolladores de una aplicación (tomado de Zapata & Olaya, 2007).

Finalmente, la visión de los vendedores de software, expresada de forma similar en los cómic de Zapata y Olaya (2007), se muestra en la figura 14. En ella, el vendedor de software presenta unos argumentos, pero otra es la realidad que rodea el desarrollo de la aplicación, ausente de propuestas metodológicas y con un bajo uso de estándares en las diferentes fases del ciclo de vida del software.

Figura 14. Parodia de la opinión de los vendedores de software en relación con la estandarización y los procesos de calidad (tomado de Zapata & Olaya, 2007)

La problemática esbozada constituye la motivación para que, en este artículo, se presente un recuento de los avances en la generación automática de código fuente para el software, fundamentando la propuesta en los denominados esquemas preconceptuales. Estos avances corresponden al trabajo que, en esta materia, viene realizando el Grupo de Investigación en Lenguajes Computacionales de la Universidad Nacional de Colombia, sede Medellín. Los esquemas preconceptuales constituyen un punto de partida para esta aproximación, puesto que por su cercanía con el discurso del interesado, posibilitan la interacción entre este y el analista para facilitar la validación de sus conceptos y relaciones.

El resto de este artículo se estructura de la siguiente manera: en la sección 2 se presenta una conceptualización de la problemática, que incluye el trabajo previo del Grupo de Investigación en Lenguajes Computacionales relacionado con este tema; en la sección 3 se presenta el compendio de los avances que rodean la generación automática de código, desde la óptica del Grupo de Investigación en Lenguajes Computacionales; finalmente, en la sección 4 se discuten las conclusiones y el trabajo futuro.

1. CONCEPTUALIZACIÓN DE LA PROBLEMÁTICA

Se denomina educción de requisitos al proceso que se realiza para capturar las necesidades y expectativas de los interesados para traducirlas, posteriormente, en especificaciones formales o semiformales de software (Leite, 1987). La educción de requisitos es un proceso complejo, porque se basa en la interacción humana y el nivel de comprensión que puedan alcanzar los actores del proceso sobre el dominio del problema.

En la figura 15 se esquematizan las principales dificultades del proceso (denotadas por símbolos de interrogación en las líneas discontinuas) que se resumen en la baja comprensión que tienen los analistas en relación con el discurso del dominio y el exiguo conocimiento que exhiben los interesados para entender esquemas conceptuales o representaciones sintácticas o semánticas del discurso (Zapata & Olaya, 2007).

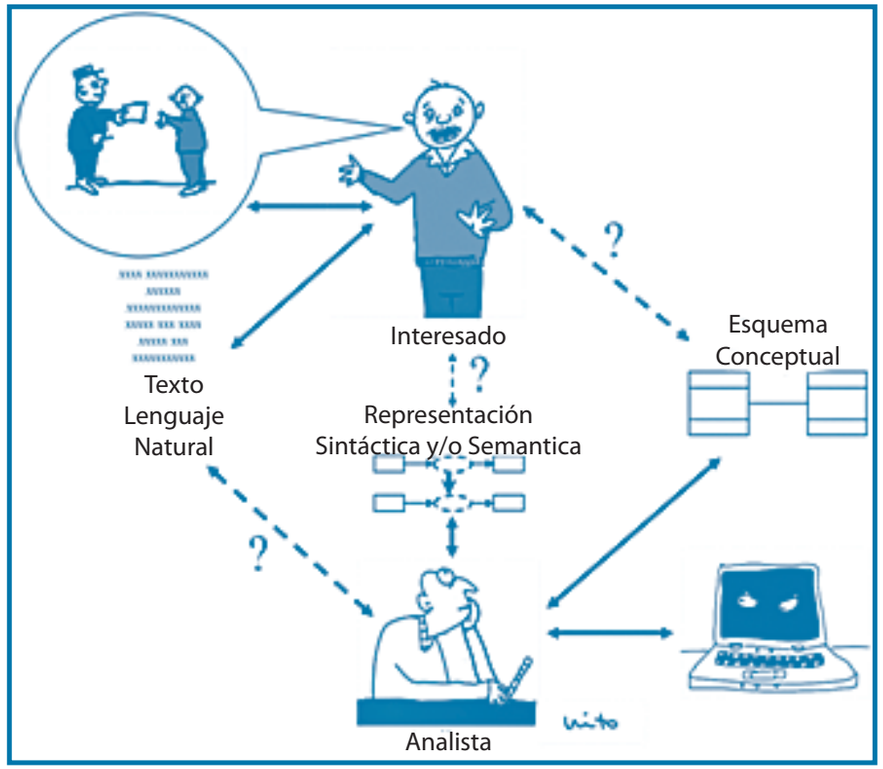

Figura 15. Representación de las dificultades de la educción de requisitos (tomado de Zapata & Olaya, 2007).

Los principales esfuerzos que se vienen realizando en educción de requisitos se suelen concentrar en la conversión automática de esquemas conceptuales en código fuente (cuya representación es el computador en la figura 4). Esto se suele hacer con herramientas CASE convencionales. Sin embargo, hacerlo de esta manera presenta tres problemas principales: el desconocimiento de los interesados en relación con los esquemas conceptuales, el descuido en automatización de los elementos iniciales del proceso (en la figura 4, el texto en lenguaje natural y su representación sintáctica y/o semántica) y los proble-

mas de completitud que tiene el código generado, pues se suele extraer de diagramas de clases y secuencias, generando tan sólo una mínima parte del código necesario para una aplicación ejecutable.

En la figura 16 se presenta otra posible solución al problema, que compendia los esfuerzos del Grupo de Investigación en Lenguajes Computacionales de la Universidad Nacional de Colombia, sede Medellín. Allí, se muestran los denominados esquemas preconceptuales (Zapata *et al.*, 2006) como una representación intermedia del dominio del problema, que se origina desde un discurso en el lenguaje controlado UN-LENCEP (Zapata *et al.*, 2008), que se combinan en el entorno UNC-Diagramador para generar esquemas conceptuales de UML desde el lenguaje controlado. El uso de estos elementos y el resultado que se genera se puede apreciar en la figura 17, donde el dominio del problema se representa gráficamente (por facilidad, aunque se supone que hay un equivalente en lenguaje natural del mismo). Luego, se define el discurso en UN-Lencep, del cual se obtiene el esquema preconceptual para, finalmente, generar los diferentes diagramas de UML (Clases, Comunicación y Máquina de Estados). Sin embargo, en este proceso no se genera código fuente, sino únicamente los diagramas que especifican el dominio del problema, lo que deja aún en manos de los desarrolladores la traducción de los esquemas conceptuales en la aplicación de software que sirve para solucionar los diferentes problemas inherentes al dominio.

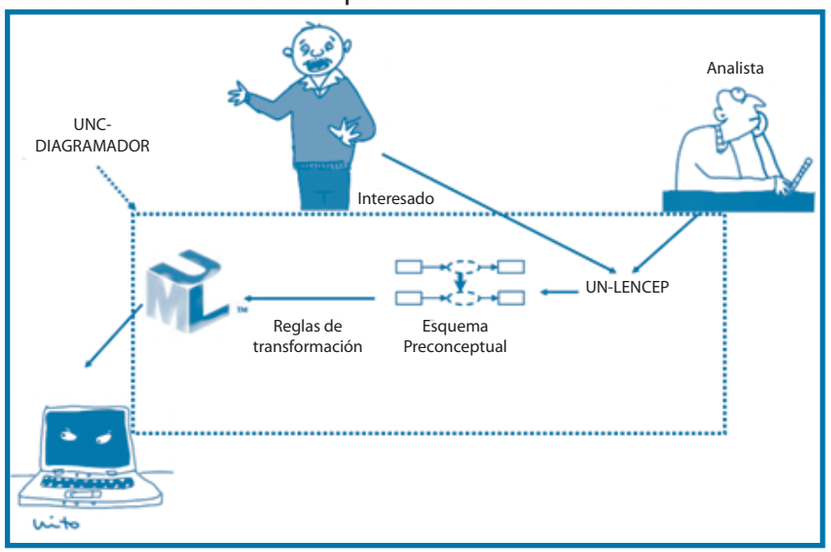

Figura 16. Compendio de la solución que propone el Grupo de Investigación en Lenguajes Computacionales (Tomado de Zapata & Olaya, 2007)

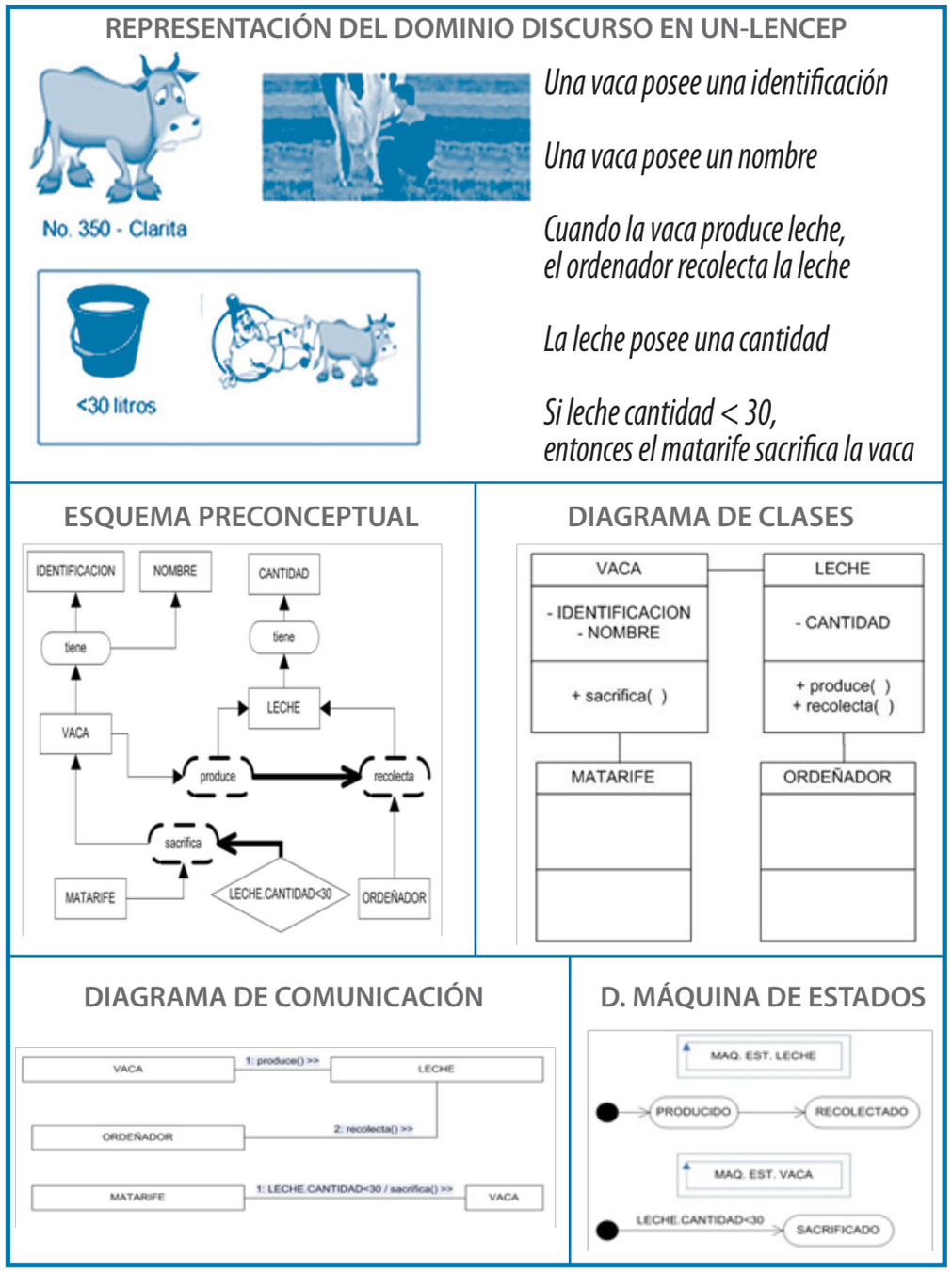

Figura 17. Representación del proceso desde la perspectiva del Grupo en Lenguajes Computacionales (tomado de Zapata & Olaya, 2007).

2. AVANCES EN GENERACIÓN AUTOMÁTICA DE CÓDIGO DESDE LENGUAJE CONTROLADO: UNA VISIÓN DEL GRUPO DE INVESTIGACIÓN EN LENGUAJES COMPUTACIONALES

Zapata y Chaverra (2010) emplean el trabajo previo del grupo en Lenguajes Computacionales y lo complementan para generar código fuente correspondiente a las interfaces gráficas de usuario de una aplicación ejecutable. Definen un conjunto de plantillas que permiten obtener código en el lenguaje Java para trazar automáticamente las interfaces gráficas de usuario. Así, es posible generar campos (figuras 18 y 19), listas de valores (figura 20), botones radio o listas desplegables (figura 21) y cajas de chequeo o listas de selección múltiple (figura 22).

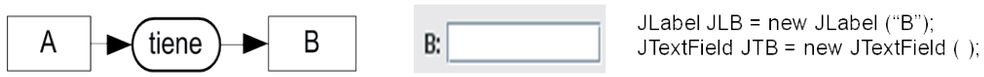

Figura 18. Generación del código de un campo sencillo a partir de un concepto hoja (tomado de Zapata & Chaverra, 2010).

Figura 19. Generación de varios campos a partir de conceptos hoja de diferentes conceptos (tomado de Zapata & Chaverra, 2010).

Figura 20. Generación de listas de valores a partir de conceptos únicos (tomado de Zapata & Chaverra, 2010).

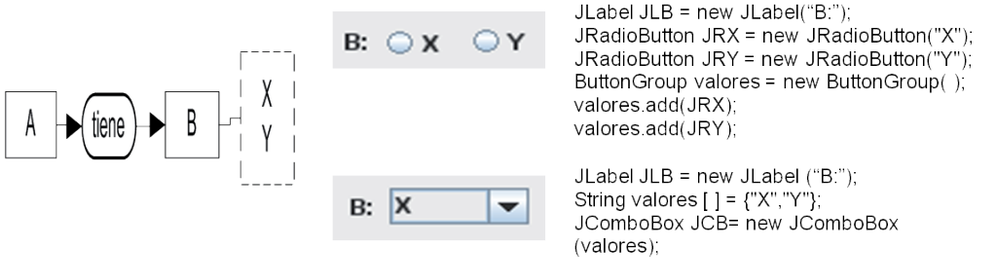

Figura 21. Generación de botones radio o listas desplegables a partir de posibles valores (excluyentes) de un concepto (tomado de Zapata & Chaverra, 2010).

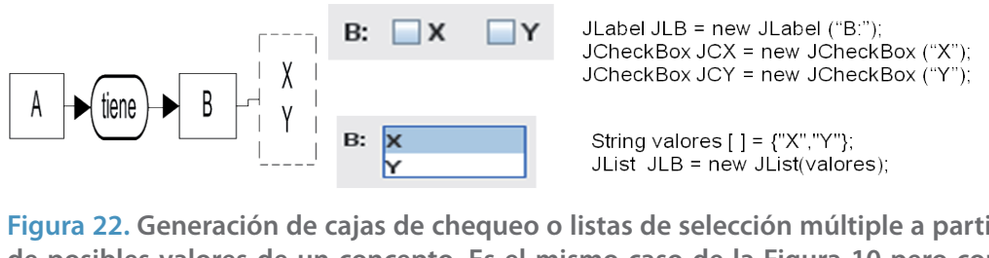

Figura 22. Generación de cajas de chequeo o listas de selección múltiple a partir de posibles valores de un concepto. Es el mismo caso de la Figura 10 pero con posibles valores no excluyentes (tomado de Zapata & Chaverra, 2010).

Con fines de ejemplificación de las reglas mencionadas, se presenta el esquema preconceptual de la figura 23. Las interfaces que se pueden generar, incluyendo la porción del esquema que las origina, se pueden apreciar en las figuras 24, 25 y 26.

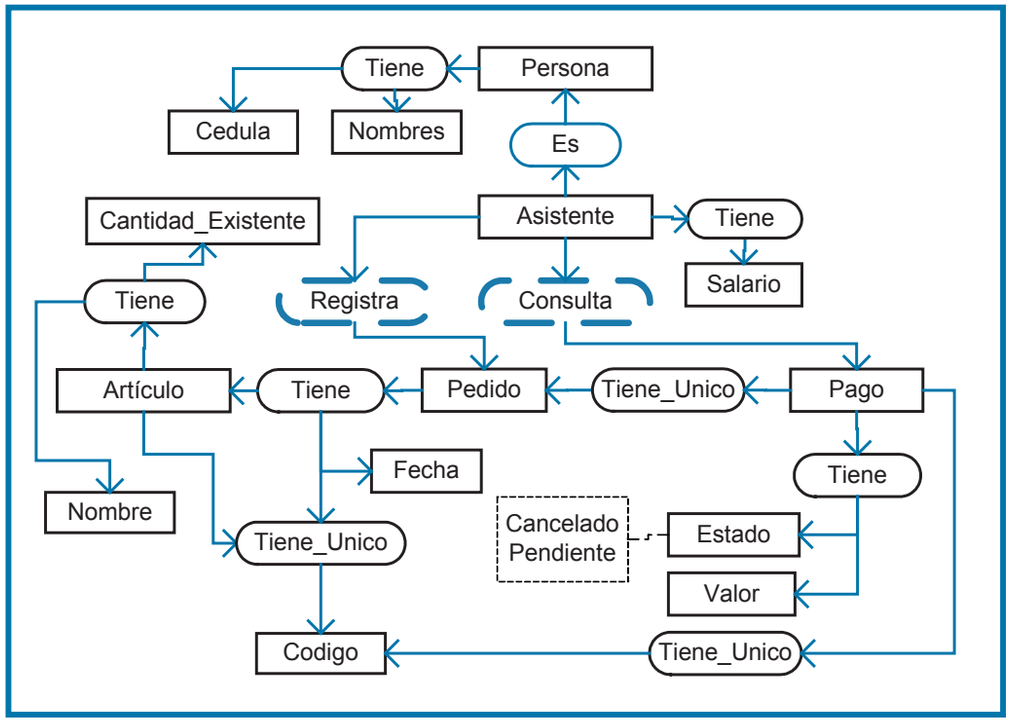

Figura 23. Ejemplo de un esquema preconceptual (tomado de Zapata y Chaverra, 2010).

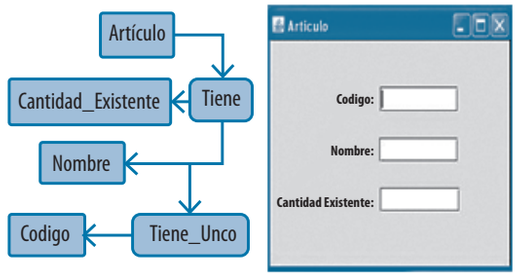

Figura 24. Generación de campos de texto en el ejemplo (Zapata & Chaverra, 2010).

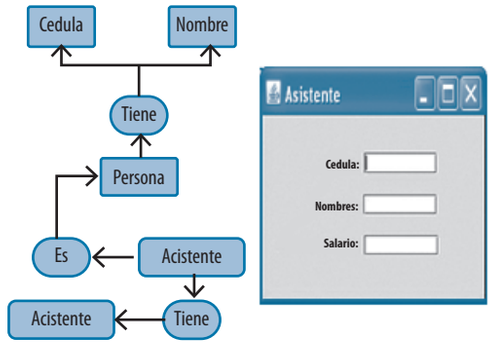

Figura 25. Generación de campos de texto desde estructuras más complejas (tomado de Zapata & Chaverra, 2010)

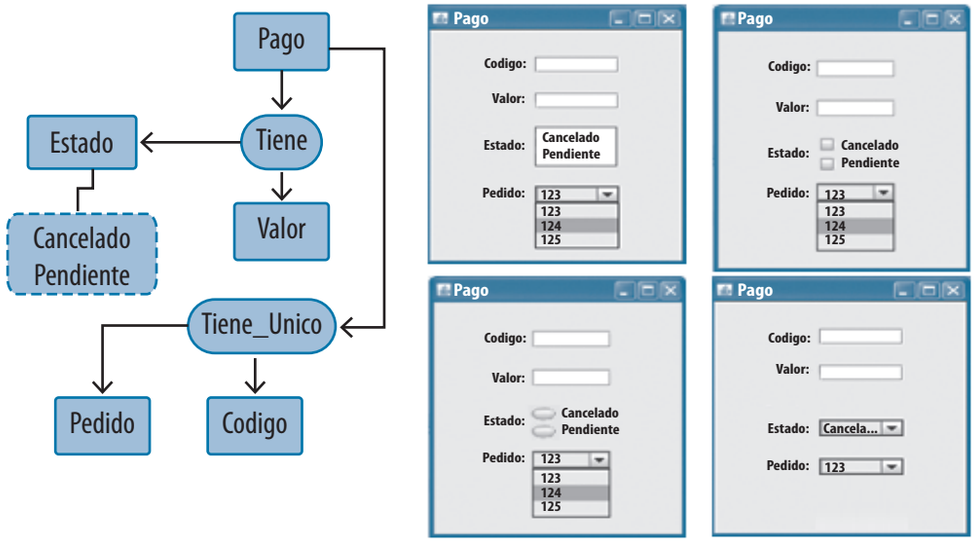

Figura 26. Generación de diferentes opciones de diseño para posibles valores (tomado de Zapata & Chaverra, 2010).

3. CONCLUSIONES Y TRABAJO FUTURO

El proceso de educción de requisitos se apoya, tradicionalmente, en herramientas CASE convencionales, que posibilitan el trazado de los diferentes diagramas que especifican el dominio del problema y luego permiten realizar su traducción a código fuente. El resultado de ese proceso es, todavía, código incompleto, puesto que se generan sólo porciones de algunos elementos del código sin que se pueda considerar aún ejecutable. Además, el punto de partida carece de la validación que puede suministrar el interesado, puesto que los lenguajes de partida de las herramientas CASE son esquemas conceptuales, lejanos del conocimiento y entendimiento del interesado.

El Grupo de Investigación en Lenguajes Computacionales, consciente de estas falencias, viene desarrollando una estrategia que permita resolver parcialmente estos dos problemas. Así, se generó, en principio, un entorno para capturar el dominio del interesado para luego representarlo en esquemas preconceptuales y, finalmente, transformarlo en esquemas conceptuales de UML. Posteriormente, se generó un conjunto de reglas heurísticas que permite la generación de las interfaces gráficas de usuario, en lenguaje Java, completamente ejecutables y funcionales. Algunas de esas reglas permiten diferentes opciones de diseño, dependiendo de las características de los datos y de las preferencias del interesado.

De esta manera, se involucra completamente al interesado en el proceso de creación de la aplicación, a la vez que se logra su intervención en la validación desde etapas tempranas del desarrollo. El código generado es consistente con el discurso representado en el esquema preconceptual y con los diferentes diagramas de UML que se pueden generar, dado que provienen todos del mismo discurso en el lenguaje controlado UN-Lencep. Se espera que esta solución se pueda traducir en mayor celeridad para desarrollar aplicaciones de software y, por ende, en costos de desarrollo inferiores para este proceso.

Las líneas de trabajo futuro que se pueden derivar de este trabajo son las siguientes:

- Adición de nuevos elementos a los esquemas preconceptuales que, sin sacrificar la sintaxis sencilla de dichos esquemas, permita una especificación más precisa, lo que se traduciría en más elementos para la transformación a código fuente.

- Determinación de equivalencias del UN-Lencep en lenguaje natural. Aún los discursos en UN-Lencep son muy controlados y eso, en ocasiones, puede dejar por fuera de la especificación elementos que se mencionan en el discurso en lenguaje natural.
- Definición de nuevas reglas heurísticas que permitan tomar decisiones de diseño en relación con las interfaces gráficas de usuario. Entre más se acerquen estas decisiones a las preferencias del interesado, mayor probabilidad de éxito tendrá la aplicación que se desarrolle.
- Definición de reglas heurísticas que tomen como punto de partida otros diagramas de UML que se puedan generar desde los esquemas preconceptuales. Además, se requieren otras reglas heurísticas para mejorar la especificación de la aplicación que se expresa en el código fuente.
- Definición de un esquema de persistencia de los datos que posibilite la interacción de las interfaces así generadas con sistemas gestores de bases de datos. Esta línea de trabajo futuro permitiría cerrar la brecha entre los esquemas preconceptuales y los lenguajes de programación, de forma que el trabajo de los desarrolladores se especializaría y se alejaría de las tareas simples y repetitivas que se encuentran en la elaboración de aplicaciones.
- Incorporación del entorno completo en las herramientas CASE, de modo que los analistas lo puedan emplear en la solución de los problemas en diferentes dominios.

AGRADECIMIENTOS

Una gran parte de este trabajo se realizó en el marco de dos proyectos de investigación:

- "TRANSFORMACIÓN SEMIAUTOMÁTICA DE LOS ESQUEMAS CONCEPTUALES, GENERADOS EN UNC-DIAGRAMADOR, EN PROTOTIPOS FUNCIONALES", bajo la financiación de la Vicerrectoría de Investigación de la Universidad Nacional de Colombia.
- "UN MODELO DE DIÁLOGO PARA GENERACIÓN AUTOMÁTICA DE ESPECIFICACIONES EN UN-LENCEP", bajo la financiación de la Dirección de Investigación de la Sede Medellín de la Universidad Nacional de Colombia.

LISTA DE REFERENCIAS

BORLAND Software Corporation. Borland Together™ Architect. En: http://www.borland.com/us/products/together/index.html. [Consultado 10 de Agosto de 2010].

Gentleware Business to model. Poseidon® http://www.gentleware.com/. [Consultado 10 de Agosto de 2010].

IBM Corporation. Rational Rose Architect™. En: http://www-306.ibm.com/software/awdtools/architect/swarchitect/index.html. [Consultado 10 de Agosto de 2010].

Leite, J. (1987). A survey on requirements analysis, Advanced Software Engineering Project. Technical Report RTP-071, Department of Information and Computer Science, University of California at Irvine.

Tigris.org. ArgoUML®. http://argouml.tigris.org/. [Consultado 10 de Agosto de 2010].

Zapata, C. M. & Chaverra, J. (2010). *Generación automática de interfaces gráficas de usuario a partir de esquemas preconceptuales*. Memorias del Quinto Congreso Colombiano de Computación (5CCC), Cartagena.

Zapata, C. M., Gelbukh, A. & Arango, F. (2006). Pre-conceptual Schemas: A Conceptual-Graph-Like Knowledge Representation for Requirements Elicitation. *Lecture Notes in Computer Science*, Vol. 4293, pp. 17–27.

Zapata, C. M., Gelbukh, A. & Arango, F. (2008). UN-LENCEP: A Controlled Language for Pre-conceptual Schema Specification. Memorias de las VII Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento, Guayaquil, pp. 269-276.

Zapata, C. M. & Olaya, Y. (2007). *Ingeniería de Software para analistas*. Medellín: C. Zapata (Ed.).