SmallHelp: un sistema de ayuda para el entorno Smalltalk

J. M. Gómez Hidalgo, M. de las Mercedes Gómez Albarrán, Ana María Fernández-Pampillón Cesteros
{jmgomez,albarran,apampi@eucmax.sim.dia.es}
Departamento de Informática y Automática
Universidad Complutense de Madrid

Los entornos de programación orientada a objetos (POO) ofrecen varias ventajas, entre las que cabe destacar la posibilidad de reutilizar trabajo previo. Sin embargo, la tarea de desarrollar programas en la POO no es sencilla, y es importante proporcionar al programador herramientas que faciliten dicha tarea. SmallHelp es un sistema de ayuda basado en técnicas de inteligencia artificial, que facilita al usuario la localización de métodos del lenguaje Smalltalk que realicen funciones determinadas. Hemos seguido la línea tradicional de los sistemas de ayuda inteligentes, simplificando sus objetivos para disminuir el esfuerzo de desarrollo de nuestro sistema. Asimismo, SmallHelp es fácilmente adaptable a otras áreas de aplicación.

1. Introducción

El paradigma de la programación orientada a objetos se ha hecho muy popular en los últimos años. Los ingenieros de software ven numerosas ventajas en esta aproximación, entre las cuales se pueden destacar la mejora de la calidad de los programas, la reutilización extensiva de los objetos software y la facilidad de mantenimiento gracias al robusto encapsulamiento de código [Boo 86, Kru 92, Mil 95]. Ahora bien, estas ventajas no están exentas de inconvenientes. Por ejemplo, los mecanismos de herencia y polimorfismo que proporcionan fuerza a los lenguajes orientados a objetos generan la aparición de un gran número de dependencias entre las componentes; la vinculación dinámica aumenta el número de implementaciones que hay que examinar para localizar aquello que se necesita, y la dispersión de funcionalidad entre los diferentes componentes del entorno hace que su comprensión global sea difícil [Wil 92]. Además, la curva de aprendizaje en estos lenguajes de programación tiene un escalón inicial muy pronunciado [Nie 89], es decir, se requiere gran cantidad de tiempo para tener un conocimiento adecuado de la metodología y familiarizarse con las diferentes clases del sistema. Debido a estos problemas sería deseable la existencia de herramientas que den soporte a los programadores, paliando alguna de estas dificultades [Fer 95], p.e. que ayuden al programador a localizar el objeto software (clase o método) que más se aproxime a sus necesidades. Esta ha sido la motivación de nuestro trabajo: el desarrollo de un sistema de ayuda para un entorno de programación orientada a objetos.

El objetivo de un sistema de ayuda es proporcionar la información necesaria al usuario para que pueda continuar con su trabajo. Es decir, cuando el usuario se encuentra con un problema que no sabe resolver por sí solo, puede ejecutar el sistema de ayuda que le proporciona o facilita el acceso a la información requerida para solucionarlo. La construcción de estos sistemas de ayuda se ha abordado desde enfoques muy diversos, p.e. con la utilización de hipertextos o ayudas dependientes de contexto [Kea 88]. Nuestra aproximación es diferente ya que se basa en la aplicación de técnicas de inteligencia artificial, como el procesamiento del lenguaje natural. Para esto tomamos como punto de partida la experiencia de los tutores inteligentes (Intelligent Tutoring Systems, ITS) [Wen 87], reali-

zando simplificaciones a su arquitectura que permitan simplificar su coste y complejidad de desarrollo [Win 92]. Nuestro sistema interactivo de ayuda SmallHelp, sigue la línea de trabajo de otros sistemas como ARGOS, ARES [Fer 94, Bue 95] y ROSA [Gir 93, Gir 94] que se han aplicado con éxito en el dominio de la programación en el sistema operativo Unix.

SmallHelp es un sistema que ofrece ayuda a los programadores en el lenguaje de programación orientada a objetos Smalltalk. El objetivo es proporcionar respuesta a las solicitudes que plantea un programador cuando le surge una duda sobre cual es el método a utilizar en una determinada situación. Los aspectos fundamentales de nuestro sistema son que permite la interacción en lenguaje natural con el usuario, de modo que simplifica su uso, y que se realiza la construcción automática del modelo del dominio, con lo que se reduce su coste de producción. En el desarrollo de SmallHelp hemos tenido en cuenta dos criterios fundamentales: minimizar el esfuerzo de desarrollo y facilitar su transportabilidad a otros dominios o áreas de aplicación.

A continuación presentamos el sistema SmallHelp, describiendo las técnicas básicas empleadas para su desarrollo. Posteriormente describimos su interfaz y explicamos su funcionamiento. Finalmente presentamos nuestras conclusiones y las líneas de trabajo futuro.

2. Descripción del sistema Smallhelp

Es frecuente que los programadores en lenguajes de POO ignoren qué método pueden emplear para realizar una tarea. En esta situación pueden encontrar útil consultar un sistema de ayuda interactivo para obtener el método que realiza esa tarea o la más parecida en el sistema. De esta forma, no es preciso que el usuario cree el método, pudiendo reutilizar uno ya existente.

El sistema SmallHelp proporciona esa clase de ayuda para los programadores en el lenguaje Smalltalk. El programador en Smalltalk puede describir la función que desee de un método empleando el lenguaje natural. El sistema analiza la solicitud del programador y le presenta aquellos métodos que realizan la función deseada u otra semejante. El usuario puede examinar dentro del entorno los métodos que se le ofrecen.

SmallHelp se basa en la comunicación por medio del lenguaje natural y en la construcción automática del modelo del dominio.

Interacción en lenguaje natural: El sistema SmallHelp permite al usuario emplear el lenguaje natural para expresar la función que necesita realizar. El lenguaje natural no es el único método posible para interaccionar con los sistemas de ayuda o los tutores inteligentes: algunos sistemas como STEAMER permiten la interacción por medio de interfaces gráficos [Wen 87]. Sin embargo, numerosos sistemas clásicos de ITS emplean el lenguaje natural para interaccionar con el usuario o estudiante. Algunos ejemplos clásicos son SCHOLAR, SOPHIE y GUINDON [Wen 87]. El lenguaje natural como medio de comunicación posee características muy útiles: evita el aprendizaje de un lenguaje formal, facilita la expresión de consultas complejas y posibilita un tratamiento del discurso (elipsis, referencias con pronombres, etc.) que facilita la comunicación con el usuario [And 95].

Algunos de los sistemas de más reciente desarrollo que permiten efectuar consultas expresadas en lenguaje natural, como ARGOS o ARES, están basados en técnicas de tratamiento estadístico de textos provenientes del campo de la Recuperación de la Información [Sal 89]. Estas técnicas se complementan con otras basadas en el procesamiento del lenguaje natural (PLN) con el fin de mejorar la precisión de los resultados. Otros sistemas recientes, como ROSA, realizan una procesamiento más basado en el conocimiento, pretendiendo lograr una comprensión más profunda del significado de las frases. Nuestro sistema SmallHelp sigue la línea representada por el sistema ROSA. Esperamos que SmallHelp sirva para estimar la efectividad de la utilización únicamente de técnicas de PLN, tanto a nivel de la eficacia del sistema como a nivel de esfuerzo de desarrollo y transportabilidad.

Construcción automática del modelo del dominio: En el dominio del software existe gran cantidad de información funcional almacenada en términos de documentación textual [Maa 91] en formato electrónico. Por ejemplo, el Manual de Unix [Sun 89] contiene, entre otras cosas, descripciones de los comandos del sistema operativo Unix y de las funciones de la biblioteca del lenguaje C que provee esa versión del sistema operativo, y la Enciclopedia de Clases del lenguaje Smalltalk [Dig 92] contiene descripciones de los métodos o funciones que ofrecen las clases del sistema. Toda esta información puede ser reutilizada en la construcción del modelo del

dominio.

Nosotros empleamos las descripciones de los métodos de Smalltalk para automatizar el proceso de construcción. Esto lleva a dos consideraciones:

- La información está expresada en un lenguaje natural muy próximo al que el usuario podría emplear para describir sus necesidades. Por ello, el lenguaje natural permite en este caso una interacción más simple y *natural*.
- La construcción automática limita la complejidad del modelo del dominio, y esto influye en la elección del método de representación de este modelo. Hemos elegido representar los métodos de Smalltalk por medio de registros de una base de datos relacional [Ull 88]. Esta base, denominada SmallCom, contiene información acerca de la función que realiza cada método, la clase a la que pertenece y el tipo de método de que se trata (método de clase o de instancia). De este modo, la interfaz al sistema se convierte en una interfaz en lenguaje natural a una base de datos [And 95], que traduce el lenguaje natural al lenguaje de acceso a bases de datos relacionales SQL [Gom 95].

En los siguientes puntos describimos los aspectos más interesantes de nuestro sistema SmallHelp. Comenzamos introduciendo las técnicas empleadas en su desarrollo, a continuación mostramos cómo está organizado el sistema. Finalizamos este punto presentando cada componente: en primer lugar describimos la componente de procesamiento de la consulta y en segundo lugar la de creación de la base de datos SmallCom.

2.1. Técnicas empleadas

Para minimizar el esfuerzo de desarrollo hemos usado un formalismo gramatical basado en restricciones [Shi 89, Shi 92] como entorno de representación del conocimiento lingüístico, y una base de datos léxica ya existente, WordNet [Mil 93]. El formalismo gramatical elegido ha sido la Gramática de Cláusulas Definidas (Definite Clause Grammar, DCG) [Per 80].

- Las gramáticas basadas en restricciones permiten expresar y depurar de una manera rápida el conocimiento lingüístico. Este conocimiento lingüístico es de carácter general, frente al enfoque de las gramáticas semánticas empleadas en sistemas como SOPHIE. De esta forma se facilita la adaptación a otros dominios.
- Las bases de datos léxicas, accesibles electrónicamente, contienen gran cantidad de información reutilizable que contribuye a hacer menor el esfuerzo de desarrollo de sistemas que precisan un tratamiento del lenguaje natural. En este momento empleamos WordNet para averiguar la categoría sintáctica de las palabras.

Por otra parte, hemos empleado la técnica clásica de la teoría de casos para describir el significado de las expresiones en lenguaje natural, tanto de las consultas de usuario como de las descripciones de los métodos de Smalltalk. La gramática de casos es uno de los enfoques mejor conocidos para el procesamiento semántico del lenguaje natural [Win 83], y ha sido empleado en diversos sistemas inteligentes, desde sistemas de consejo médico hasta sistemas de comprensión del discurso.

La noción lingüística tradicional de caso hace referencia a la clasificación de los nombres con respecto al papel sintáctico que desempeñan en una oración, determinado por la declinación o sufijo. Estos casos se suelen llamar *casos superficiales* o *sintácticos*, y son característicos de lenguajes naturales como el latín, ruso o finés.

Alternativamente, se propone otro concepto de caso, *profundo* o *semántico*, que se define como la categorización de los sintagmas nominales de acuerdo con los papeles conceptuales que juegan en la acción representada por una frase. Se trata de representar el significado de la oración por medio de la acción que en ella se describe, en la que los sintagmas nominales representan características de dicha acción. Los papeles conceptuales son independientes del verbo particular que aparece en la frase. Por ejemplo, el caso AGENTE es la generalización de numerosas ideas: lector, profesor, reloj, etc.: alguien o algo que realiza una acción.

Una teoría de casos consiste en la identificación de un conjunto limitado de papeles conceptuales que pueden ser utilizados para describir el significado de una frase expresada en cualquier lenguaje natural. Dado que son los casos y no las palabras los que representan el significado de la frase, el conjunto o sistema de casos debería ser independiente del idioma. Históricamente se han propuesto diversos sistemas generales de casos, entre los que cabe destacar los de Fillmore, Simmons, Schank, Chafe o Grimes [Win 83].

Recientemente se han presentado sistemas de casos dependientes tanto del idioma empleado como del dominio en el que se trabaja. El sistema ROSA emplea un sistema de casos específico para el idioma inglés utilizado en las descripciones de los comandos del sistema operativo Unix, que aparecen en el Manual de dicho sistema.

2.2. Organización del sistema

El sistema que presentamos consta de dos partes principales, una interactiva y otra de preprocesamiento. La primera de ellas está agrupada tras la interfaz del sistema, mientras que la segunda ha sido ejecutada una sola vez para crear la base de datos SmallCom, que contiene la información correspondiente a los métodos de Smalltalk. En la Figura 1 se muestra un gráfico que representa la estructura del sistema SmallHelp.

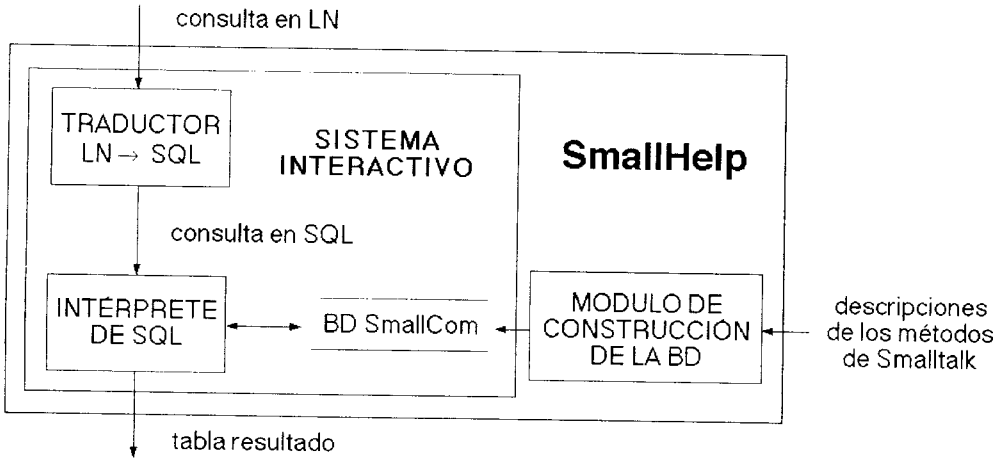

Figura 1. Estructura del sistema SmallHelp.

2.3. Traducción de las consultas a SQL

La componente de traducción de lenguaje natural a SQL realiza sucesivamente las tareas siguientes: análisis sintáctico, análisis semántico y traducción. Describimos cada tarea de modo breve en los próximos apartados.

2.3.1. Análisis sintáctico

El analizador sintáctico recibe como entrada una frase perteneciente a un subconjunto del lenguaje natural y devuelve un árbol de análisis sintáctico que representa la estructura de la frase.

El subconjunto del lenguaje natural admitido por nuestro sistema consiste en una parte de las frases interrogativas del inglés, particularmente las que involucran pronombres interrogativos como *what* o *which*. Los verbos admiten varios complementos, cuyo número y clase están limitados por restricciones relativas a preposiciones y a la transitividad. El sistema es capaz de admitir preguntas como "Which method can add two numbers?"

2.3.2. Análisis semántico

El analizador semántico recibe como entrada el árbol de análisis de la frase introducida. Haciendo uso de unas correspondencias definidas desde los árboles de análisis a las listas de casos, construye una lista que representa el significado de la frase. Las correspondencias se obtienen por medio de estructuras que relacionan algunos marcadores superficiales con casos semánticos para cada verbo.

Se ha propuesto el siguiente sistema de casos para representar el significado de una frase independientemente del dominio:

AGENT	el instigador o causante último de una acción
ATRIBUTE	representa una característica del agente
OBJECT	la entidad sobre la que se realiza una acción
LOCATION	lugar donde tiene lugar una acción
TIME	momento temporal en que se realiza una acción
BENEFICIARY	destinatario paciente de una acción
INSTRUMENT	objeto o causa inmediata física de una acción
DESTINATION	lugar de destino de una acción
SOURCE	lugar de origen de una acción

El caso ATRIBUTE está muy relacionado con el de AGENT, y aparece debido a la dificultad de tratar el verbo *to be*.

Los marcadores superficiales más característicos son la categoría sintáctica, las preposiciones que acompañan a los sintagmas preposicionales y los pronombres interrogativos, que introducen información sobre el caso deseado.

El análisis semántico permite además eliminar algunas frases que son sintáctica pero no

semánticamente correctas. El ejemplo característico se produce cuando se inicia una frase con el pronombre *where*, en la que sólo se puede solicitar un único caso de entre SOURCE y DESTINATION.

2.3.3. Traducción

La fase de traducción recibe como entrada una lista de casos semánticos y la transforma en una consulta expresada en el lenguaje SQL. Para interrogar a una base de datos por medio del lenguaje de consulta SQL, es suficiente utilizar la cláusula SELECT. Por ello, el subconjunto del lenguaje SQL generado por esta fase del sistema que presentamos es el descrito en las siguientes reglas independientes del contexto:

```
cláusula ::=SELECT tabla . (atributo | *)
           FROM tabla ( , tabla)*
           WHERE condición ( AND condición)*
condición ::=tabla , atributo =
              (tabla . atributo | valor)
```

donde los elementos terminales son "SELECT", ".", "FROM", "WHERE", "AND", ";" "=", mientras que "tabla", "atributo" y "valor" son clases de terminales que representan elementos SQL [Ull 88].

En esta fase se emplea conocimiento relativo al lenguaje SQL y la base de datos concreta que se esté tratando. Esta fase depende en parte de la base de datos, y es preciso adaptarla a cada base específica, aunque esta tarea conlleva poco esfuerzo.

La asignación de elementos SQL a los casos se realiza por medio de un proceso de normalización. Este proceso consiste en la búsqueda del sinónimo más apropiado entre los nombres de tablas, atributos y valores.

2.4 LA BASE DE DATOS SMALLCOM

A continuación pasamos a describir el proceso de construcción de la base de datos SmallCom, así como la estructura de la misma.

2.4.1. Creación de la base de datos

La base de datos almacena la información de los métodos del lenguaje de programación orientada a objetos Smalltalk. Dicha información constituye el modelo del experto o modelo del dominio.

Este modelo del dominio se construye a partir del manual de clases del lenguaje Smalltalk que existe en formato electrónico. El manual está organizado por clases y para cada clase se describen sus distintos métodos, separados en métodos de clase y de instancia. Cada método tiene asociada una breve descripción en inglés que describe su funcionalidad. La descripción de los métodos tiene un formato fijo: la primera línea contiene el nombre del método; las siguientes líneas, que van entrecomilladas, corresponden a la descripción propiamente dicha o comentario asociado al método. Los comentarios son frases imperativas consistentes en un verbo (representando la acción del método), posiblemente seguido por una frase nominal (representando el objeto de la acción) y quizás sintagmas preposicionales (representando el resto de complementos de la acción).

Veamos un ejemplo de descripción:

```
addFirst :anObject
"Add anObject before the first element of
the receiver"
```

La construcción del modelo del dominio se ha realizado de forma automática. A ello ha contribuido la ya citada estructura fija de la documentación de la que hacemos uso.

Para construir nuestra base de datos hacemos uso de dos módulos:

Un preprocesador cuya función consiste exclusivamente en adaptar el formato de las descripciones al que necesita el siguiente módulo. El preprocesador está implementado en Lex.

Un analizador que constituye la pieza fundamental del proceso de automatización. Está implementado en Prolog y está constituido por una gramática de casos expresada con el formalismo de las DCGs. La aplicación del analizador a la descripción de un método genera la entrada correspondiente a dicho método en nuestra base de datos relacional SmallCom.

El desarrollo del analizador fue precedido por un estudio del dominio del lenguaje natural que aparece en los comentarios de los métodos. Como fruto de este estudio proponemos el siguiente sistema de casos:

AGENT	el que realiza la acción (método)
ACTION	acción realizada
OBJECT	entidad afectada por la acción
FACTITIVE	resultado de la acción
SOURCE	fuente de la que se sirve para realizar la acción
DESTINATION	destino del objeto
INSTRUMENT	entidad con la que se realiza la acción
MANNER	modo en que se realiza la acción
REPETITION	número de veces que se realiza la acción
PURPOSE	causa por la que se realiza la acción
LOCATION	lugar donde se produce la acción exactamente
CONDITION	premisa bajo la que se realiza la acción

Este sistema de casos ha sido el utilizado y ha servido para dar forma al registro de la base de datos que pasamos a describir a continuación.

2.4.2. Estructura de la base de datos

Cada registro de la base de datos contiene los siguientes campos:

- Un campo que representa al propio método: el campo AGENT.
- Otros dos campos que nos dan información acerca de la clase y el tipo de método (método de clase o de instancia). Son los campos CLASS y TYPE, respectivamente.
- Diez campos en los que queda reflejada la información funcional. Dichos campos coinciden con los casos semánticos ACTION, OBJECT, FACTITIVE, DESTINATION, INSTRUMENT, MANNER, REPETITION, PURPOSE, LOCATION y CONDITION.

De este modo, el esquema de la base de datos es:

```
SmallCom (AGENT, CLASS, TYPE, ACTION,
OBJECT, FACTITIVE, SOURCE, DESTINATION,
INSTRUMENT, MANNER, REPETITION, LOCATION,
CONDITION)
```

SmallCom				
AGENT	add :anObject	add :anObject	add :anObject	addFirst :anObject
CLASS	Collection	Set	SortedCollection	SortedCollection
TYPE	Instance	Instance	Instance	Instance
ACTION	add	add	add	add
OBJECT	an object	an object	an object	an object
FACTITIVE	—	—	—	—
SOURCE	—	—	—	—
DESTINATION	receiver collection	receiver	receiver	receiver
INSTRUMENT	—	—	—	—
MANNER	—	—	sorted position	—
REPETITION	—	—	—	—
PURPOSE	—	—	—	—
LOCATION	—	—	—	before the first element
CONDITION	—	receiver doesnít contain it	—	—

Tabla 1. Algunos contenidos de la base de datos SmallCom.

Los cuatro primeros campos se completan para todos los métodos mientras que el resto de los campos pueden ser o no rellenados. Los campos se rellenan con cadenas de caracteres extraídas directamente de la documentación.

En la *Tabla 1* presentamos (por columnas) varios registros de la base de datos. Estos registros corresponden a cuatro métodos de instancia: los métodos add :anObject de las clases Collection, Set y SortedCollection y el método addFirst :anObject de la clase SortedCollection.

3. Funcionamiento del sistema Smallhelp

El sistema SmallHelp se comunica con sus usuarios por medio de una interfaz constituida por un sistema basado en ventanas relativamente simple. En la ventana de la interfaz se muestran tres paneles:

- El panel Query se emplea para introducir la consulta en lenguaje natural.
- El panel List of Relevants muestra la lista de métodos candidatos a ser utilizados por el usuario.
- El panel View permite visualizar tanto la descripción como el código del método que se seleccione en el panel List of Relevants.

Por medio del siguiente ejemplo de consulta mostraremos el funcionamiento del sistema. Supongamos que el programador necesita introducir un elemento nuevo en una colección, es decir en una instancia de la clase Collection. El usuario introduce su consulta en el panel Query: "Which method puts an element in the receiver collection?" (1) y pulsa el botón Search.

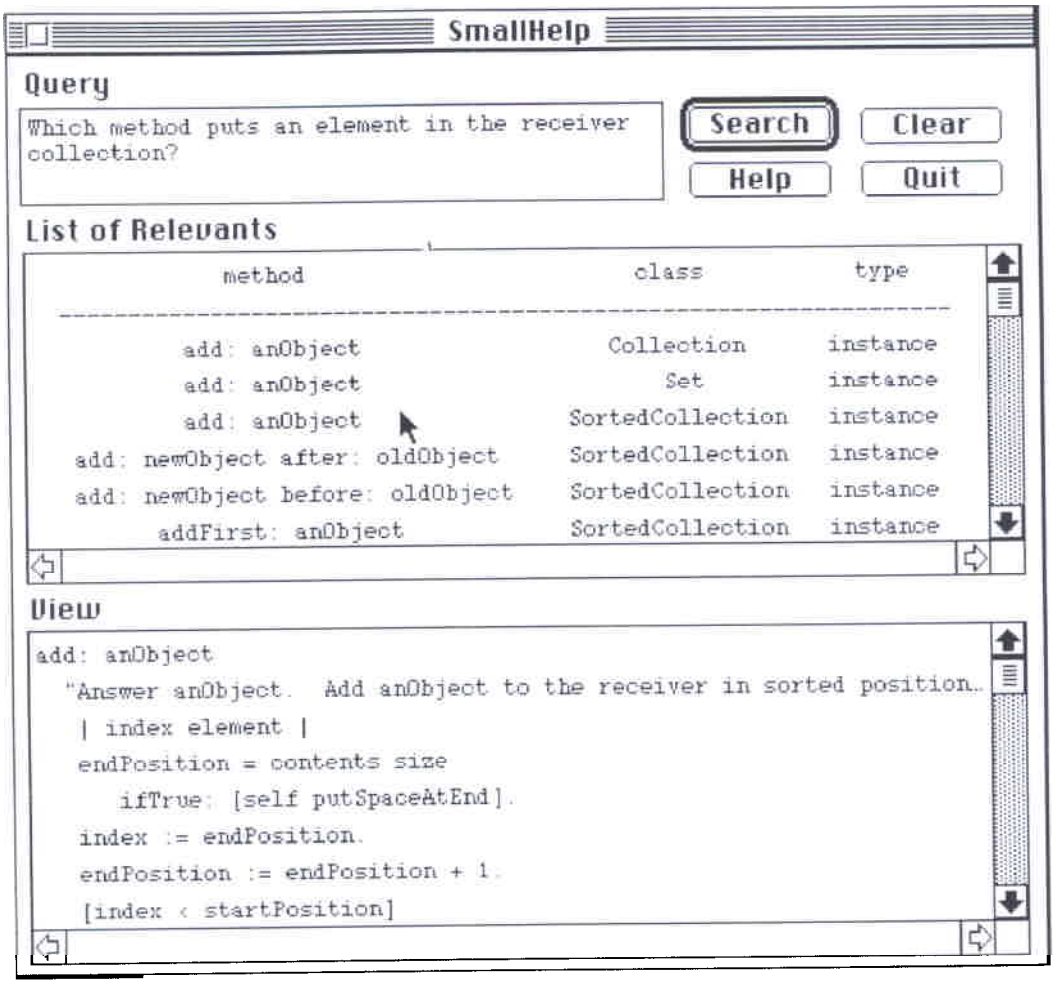

Figura 2. Respuesta del sistema a la consulta (1).

El sistema toma la consulta (1), la analiza y obtiene la sentencia SQL que responde a esa petición. La sentencia SQL correspondiente a la consulta (1) es la siguiente:

```
SELECT method, class, type
FROM   smallcom
WHERE  object="an element",
       destination="receiver collection";
```

Un intérprete de SQL realiza el acceso a la base de datos SmallCom y obtiene la tabla resultado que se muestra al usuario en el panel List of Relevants (Fig. 2).

El usuario puede seleccionar cualquiera de los métodos del panel List of Relevants para obtener más información sobre el mismo. Esta información, que aparece en el panel View, está constituida por un comentario descriptivo de la funcionalidad del método y su código.

Es posible que ninguno de los métodos que se muestra resuelva el problema del programador. En este caso, el programador tiene dos posibilidades: o bien implementar uno nuevo o bien modificar alguno de los candidatos. Esta es la razón de presentar como parte de la información el código del método.

La lista mostrada en el panel List of Relevants puede contener varios métodos candidatos, uno o ninguno. Si el usuario no realiza consultas muy específicas, el sistema probablemente devuelva varios candidatos, como ha ocurrido con la consulta (1). En una consulta muy específica como "Which method inserts an element before the first element in a sorted collection?" (2), SmallHelp encuentra exactamente el método que implementa la función deseada (Fig. 3). Finalmente, es posible que el sistema no localice ningún método que se adecue a la consulta, en cuyo caso muestra una tabla vacía.

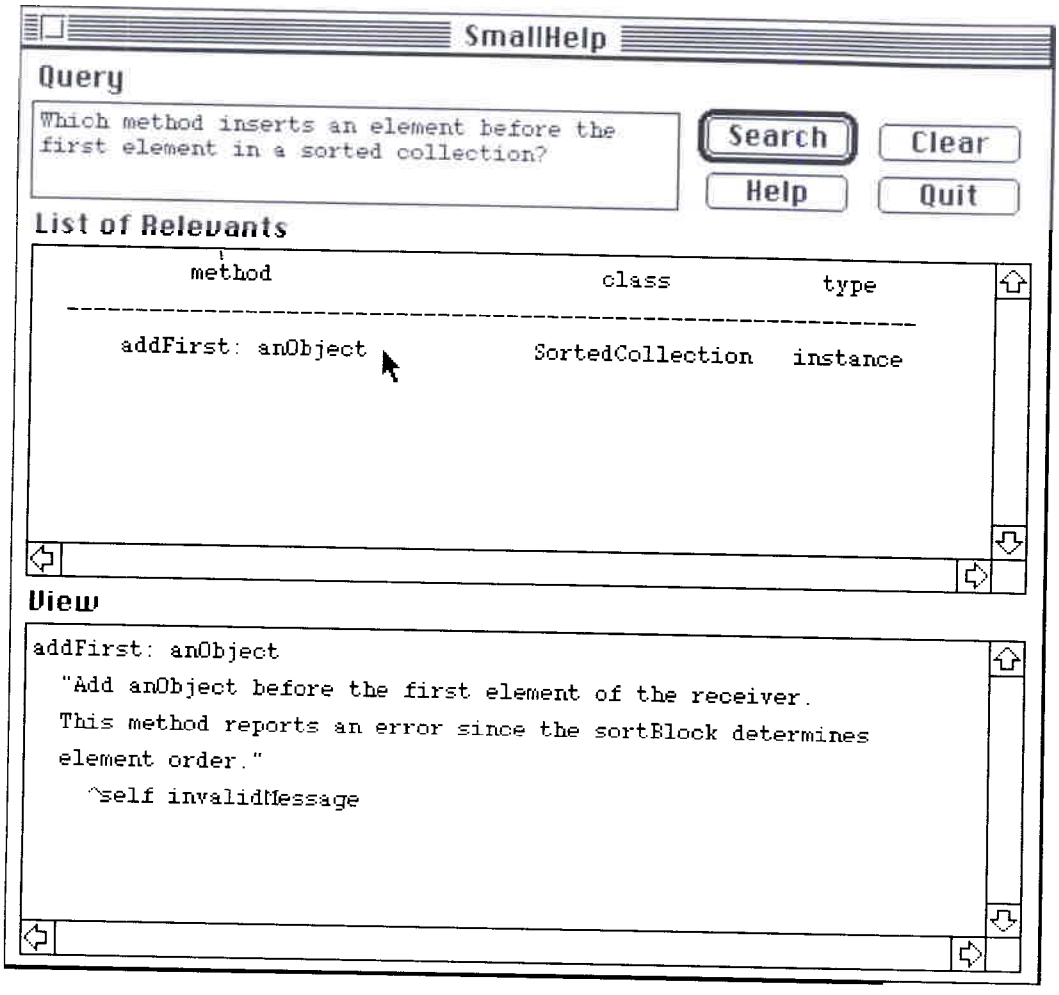

Figura 3. Respuesta del sistema a la consulta (2).

Aunque el funcionamiento e interfaz de SmallHelp son muy sencillos, siempre se puede obtener ayuda sobre su uso pulsando el botón Help.

Cuando el usuario no necesita realizar más consultas, o bien sale del sistema con el botón Quit, o bien deja abierta la ventana de SmallHelp en previsión de nuevos problemas. Para realizar una nueva consulta, el usuario pulsa el botón Clear, que borra el panel Query, e introduce una nueva consulta.

4. Conclusiones y trabajo futuro

En este trabajo hemos presentado el sistema interactivo SmallHelp, que proporciona ayuda a los programadores en el entorno de POO Smalltalk. El sistema SmallHelp facilita al usuario el empleo del lenguaje natural para comunicarse con él, y realiza una construcción automática del modelo del dominio. Estas dos facetas se han desarrollado con la ayuda de técnicas basadas en conocimiento propias de la inteligencia artificial.

El sistema SmallHelp ha sido construido minimizando su coste de desarrollo. Además, el sistema es transportable a otros dominios: esperamos realizar versiones del mismo para los entornos del sistema operativo Unix y para las bibliotecas del lenguaje de programación C.

Actualmente estamos realizando la implementación definitiva de SmallHelp, implementación que usaremos para evaluar los resultados de su funcionamiento. Las versiones para Unix y C pueden ser especialmente útiles para realizar dicha evaluación, dado el gran número de programadores en este lenguaje.

REFERENCIAS

[And 95] Androutsopoulos, I., Ritchie, G.D., Thanisch, P, 1995. ÒNatural Language Interfaces to Data Bases.Ó En Boguraev, B., Garigliano, R., Tait, John. (eds.) *Natural Language Engineering*, vol. 1, Part 1. Cambridge University Press.

[Boo 86] Booch, G., 1986. "Object-oriented development." *IEEE Transactions on Software Engineering*, vol. SE-12, pp. 211-221.

[Bue 95] Buenaga, M., Fernández-Manjón, B., Fernández-Valmayor, A., 1995. "Information Overload in the Information Age", en Collis, B., Davies, G., (eds.) *Innovative Adult Learning with Innovative Technologies*, North-Holland.

[Dig 92] Digitalk 1992. *Smalltalk/V for Windows. Encyclopedia of Clases*. Digitalk Inc., California.

[Fer 94] Fernández-Manjon, B., Buenaga, M., Fernández-Valmayor, A., 1994. "De los entornos de ayuda inteligentes a los tutores inteligentes: el sistema Argos", *II Congreso Ibero-Americano de Informática na Educaçao*, Lisboa, Portugal.

[Fer 95] Fenández Chamizo, C., González Calero, P.A., Gómez Albarrán, M. 1995. "Promoting Sofware Reuse through Explicit Knowledge Representation. " En Pinto Ferreira, C., Mamede, N. (eds.) *Progress in Artifical Intelligence, Lecture Notes in Artificial Intelligence*, 990; Springer Verlag (Capítulo), pp. 387-396.

[Gir 93] Girardi, M.R., Ibrahim, B. 1993. "A Software Reuse System Based on Natural Language Specifications.íí En *Proceedings of the 5th International Conference on Computing and Information*, Abou-Rabia, O., Chang, C.K., Koczkodaj, W.W., (eds.) pg 507-511. IEEE Computer Society Press, Sudbury, Ontario, Canada, Mayo 27-29.

[Gir 94] Girardi, M.R., Ibrahim, B. 1994. "Automatic indexing of software artifacts.íí En *Proceedings of the 3rd Int. Conf. on Software Reusability*, Rio de Janeiro, Brasil, Nov. 1-4.

[Gom 95] Gomez, J.M. 1995. "Un sistema de traducción del lenguaje natural a SQL." Informa Técnico No. 7-95, Depto. de Informática y Automática, Universidad Complutense de Madrid.

[Kea 88] Kearsley, G. 1988. *Online Help Systems. Design and Implementation*. Ablex Publishing Corporation, New Jersey.

[Kru 92] Krueger, C., 1992. "Software Reuse." *ACM Computing Surveys*, vol.24, No.2.

[Maa 91] Maarek, Y.S., Berry, D.M., Kaiser, G.E. ,1991. "An Information Retrieval Approach for Automatically Constructing Software Libraries." *IEEE Transactions on Software Engineering*, vol. 17, 8.

[Mil 93] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K., 1993. "Five Papers on WordNet." Cognitive Science Laboratory, Princeton University, CSL Report 43.

[Mil 95] Mili, H., Mili, F., Mili, A., 1995. "Reusing Software: Issues and Research Directions." *IEEE Software*, vol. 21, 6.

[Nie 89] Nielsen, J., Richards, J.T., 1989. "The experience of Learning and Using Smalltalk." *IEEE Software*, vol. 6, 3, pp. 73-77.

[Per 80] Pereira, F., Warren, H., 1980. "Definite clause grammars for language analysis, a survey of the formalism and a comparison with augmented transition network." *Artificial Intelligence*, No.13.

[Sal 89] Salton, G., 1989. *Automating Text Processing. The Transformations Analysis and Retrieval of Information by Computer*. Reading, MA: Addison-Wesley.

[Shi 89] Shieber, S., 1989. *Introducción a los formalismos gramaticales de unificación*, Editorial Teide, Barcelona.

[Shi 92] Shieber, S., 1992. *Constraint-Based Grammar Formalisms*, MIT Press, Cambridge, MA.

[Sle 82] Sleeman, D., Brown, J.S., 1982. *Intelligent Tutoring Systems*. Academic Press, New York.

[Sun 89] Sun 1989. *SunOs Reference Manual v.4.1*. Sun Microsystems Europe.

[Ull 88] Ullman, J.D., 1988 *Principles of Database and Knowledge-Base Systems*, Computer Science Press, Inc. USA.

[Wen 87] Wenger, E., 1987. *Artificial Intelligence and Tutoring Systems*, Morgan Kaufman Publishers Inc., Los Altos, CA.

[Wil 92] Wilde, N., Huitt, R., 1992. "Maintenance Support for Object-Oriented Programs." *IEEE Transactions on Software Engineering*, vol. 18, 12.

[Win 83] Winograd, T., 1983. *Language as a congnitive Process vol.1. Syntax*, Addison Wesley Publishing Company, Reading, Massachusetts.