Agentes Inteligentes para el Soporte a la Reutilización de Software

Alvaro Ortigosa[†] - Marcelo Campo

ISISTAN - Fac. Ciencias Exactas - UNICEN
[†] también Universidad Autónoma de Madrid, E.T.S. de Informática
E-mail: {ortigosa; mcampo}@exa.unicen.edu.ar

1. Introducción

A pesar de los beneficios ofrecidos por los *frameworks* orientados a objetos [4, 5], comenzar a utilizar un *framework* no es una tarea fácil. Debido a que los *frameworks* implementan estructuras de diseño muy flexibles, son muy complicados y difíciles de entender. Esta situación es agravada por la inversión de control característica de los *frameworks*: el código desarrollado por el programador de una aplicación es invocado por el código del *framework*, quien determina la estructura general y el flujo de control del sistema. Por estos motivos, dependiendo de la complejidad del *framework*, la reutilización es a menudo una tarea demasiado difícil para usuarios novatos [10]. Por ejemplo, alcanzar un alto grado de productividad utilizando un *framework* para construir interfaces de usuario usualmente lleva entre 6 y 12 meses [5]. Esta prolongada curva de aprendizaje es uno de los mayores obstáculos en la adopción de la tecnología de *frameworks* orientados a objetos

En los últimos años se han realizado diversos esfuerzos para desarrollar herramientas y técnicas de documentación que faciliten el proceso de entendimiento de un *framework*; sin embargo, ninguna de ellas consigue adaptarse adecuadamente a las necesidades de los distintos tipos de usuarios de un *framework*. Esto es especialmente cierto cuando se consideran las distintas necesidades de usuarios con diferente nivel de conocimientos y experiencia. Por un lado, los usuarios inexpertos necesitan que se les diga qué hacer, sin sobrecargarlos con detalles de diseño; estos usuarios normalmente adaptan el *framework* de acuerdo a lo que fue previsto en su diseño y documentación. Por otro lado, usuarios más expertos prefieren conocer los detalles de diseño y ser capaces de tomar sus propias decisiones; estos usuarios muchas veces adaptan el *framework* de formas no previstas originalmente.

Esta necesidad de flexibilidad lleva a pensar en la utilización de herramientas más inteligentes como un camino que permita asistir de manera efectiva en el proceso de instanciación. En este contexto, nosotros proponemos el uso de Agentes de Software Inteligentes [1] para proveer la asistencia y flexibilidad necesarias. Los Agentes Inteligentes proveen nuevas formas de analizar, diseñar e implementar sistemas de software. Un agente es una entidad computacional, capaz de percibir su entorno y reaccionar en consecuencia, con un comportamiento autónomo [3] y flexible[2]. Autonomía implica la capacidad de llevar a cabo la mayor parte de sus actividades sin intervención humana, mientras que flexibilidad significa que el agente es capaz de adaptarse a un entorno cambiante, de forma oportunista y guiado por sus objetivos [1].

En este trabajo presentamos *SmartBooks*, un método para documentar *frameworks* orientados a objetos. En base a este método, es posible diseñar un agente inteligente que guíe el proceso de instanciación, de acuerdo con la funcionalidad requerida para la aplicación que está siendo construida.

2. *SmartBooks*: Documentación Basada en Reglas

SmartBooks considera que el proceso de instanciación de *frameworks* está compuesto por un conjunto bien definido de actividades, como por ejemplo especialización de clases y reescritura de métodos; estas actividades se denominan *tareas de instanciación*. Si se cuenta con información adecuada, un agente puede derivar, a partir de una descripción de la funcionalidad requerida para una aplicación, la secuencia de tareas que debería ejecutar el usuario para implementar esa funcionalidad; esta secuencia de tareas es denominada *plan de instanciación*.

Con este objetivo, *SmartBooks* prescribe que el diseñador del *framework* describa la funcionalidad provista por el *framework* y cómo esta funcionalidad es implementada por los distintos componentes,

así como también provea reglas que acoten la especialización del *framework*. Esta documentación especial constituye las *reglas de instanciación*. Utilizando estas reglas, el agente provee información que guía al usuario del *framework* a través del proceso de desarrollo de su aplicación. Esta guía está enfocada en la funcionalidad pretendida para la nueva aplicación, así que el usuario es orientado a definir los requisitos funcionales de la nueva aplicación. En base a esta información, es elaborado un plan para la instanciación del *framework*, y el usuario debe ejecutar las tareas que componen el plan. Este plan es generado utilizando el algoritmo de planificación *PIT* (*Planning Instantiation Tasks*) [7], especialmente desarrollado para generar secuencias de tareas de instanciación en base al conjunto de objetivos funcionales que deben ser satisfechos por la nueva aplicación.

Cada tarea generada como parte del plan de instanciación es relacionada con su correspondiente documentación. Es decir, cuando el usuario está ejecutando una tarea dada, puede navegar a los documentos asociados, como por ejemplo la jerarquía de clases, los diagramas de colaboración, patrones de diseño, ejemplos, reglas, código relacionado, etc. Además, cuando se está ejecutando el plan de instanciación, el agente ofrece al usuario información sobre las restricciones aplicables el software que está siendo desarrollado, de acuerdo con la documentación del *framework*. La etapa de planificación se basa en una clase de reglas de instanciación denominada reglas funcionales; por otro lado, la asistencia durante el proceso de instanciación mismo está basada en reglas de consistencia.

Las reglas de instanciación utilizadas como entrada del algoritmo PIT pueden ser tanto reglas específicas del *framework* que se está instanciando, como reglas que describen situaciones generales que pueden encontrarse en la instanciación de distintos *frameworks*. La Figura 1 muestra un ejemplo de regla funcional, específica del *framework HotDraw* [6]. Esta regla ha sido definida utilizando la notación *TOON* [9], basada en *UML*. Los cuadrados blancos representan clases y los cuadrados con el nombre sobre un fondo negro representan tareas de instanciación. El rectángulo en la esquina superior derecha indica la funcionalidad que está siendo descrita por esa regla. La regla del diagrama establece que para implementar la animación interactiva (*InteractiveAnimation*) se deben ejecutar tres tareas de instanciación: crear una subclase de *Tool* e implementar dos métodos para esta nueva clase, *activate* y *deactivate*.

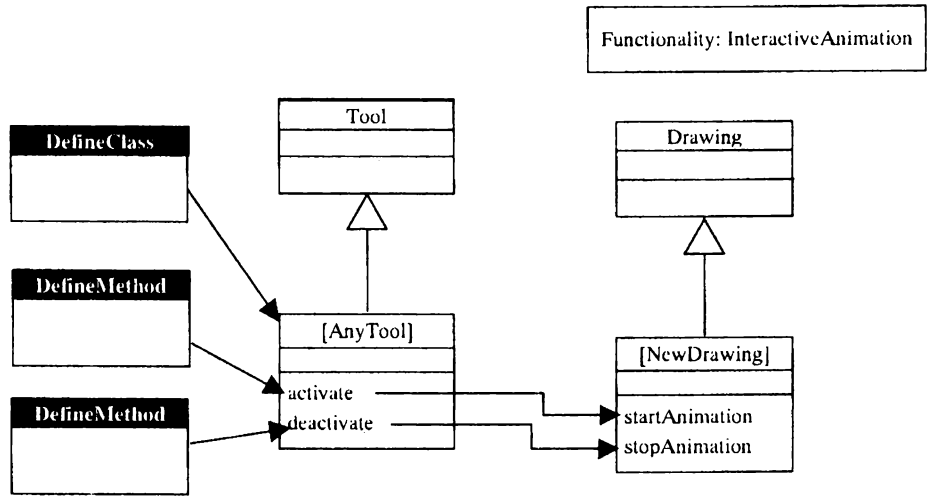

Figura 1. Descripción de reglas funcionales en *SmartBooks*

La descripción de la regla también puede proveer restricciones adicionales sobre el software a producir o modificar por cada tarea. Así, la regla de la figura establece que el método *activate* debe invocar al método *startAnimation*, mientras que el *deactivate* debe hacer lo propio con el *stopAnimation*, ambos de alguna subclase de *Drawing*.

3. Agente de Soporte a la Instanciación

En términos del planificador, la regla de la Figura 1 determina que si el usuario selecciona como parte de la funcionalidad requerida para la aplicación la animación interactiva de diagramas, tres tareas pendientes serán creadas y puestas en el plan de instanciación. Utilizando las reglas definidas para el

framework que se está instanciando y las reglas generales, el agente elabora la secuencia de tareas que deberían ser ejecutadas; estas tareas son presentadas al usuario a través de la interfaz de la Figura 2, denominado administrador de tareas.

A través del administrador de tareas el usuario puede ver tanto las tareas ya ejecutadas como las pendientes de ejecución. También puede navegar a través de la documentación relacionada con una tarea dada o seleccionar la próxima tarea para ejecutar, deshacer o cancelar. Al finalizar una tarea, el agente comprobará que se cumplan las restricciones impuestas por las reglas de instanciación. Esto muchas veces provocará la creación de nuevas tareas, necesarias para que el software desarrollado respete el diseño del *framework*.

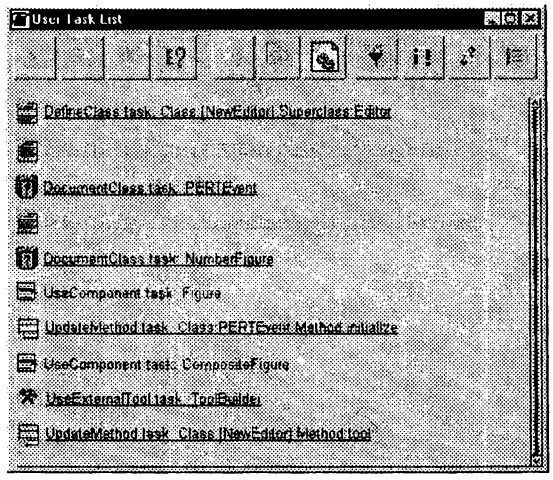

Figura 2. Interfaz de administración del plan de instanciación

4. Conclusiones

En este artículo es presentado un nuevo enfoque para asistir la instanciación de *frameworks* utilizando agentes inteligentes. Para poder utilizar los agentes, el *framework* debe ser documentado de acuerdo al método *SmartBooks*, basado en reglas de instanciación.

La principal contribución del trabajo es mostrar cómo la utilización de reglas de instanciación hacen posible la generación de la secuencia de tareas que deberían ser ejecutadas para implementar una funcionalidad dada. De esta forma, un agente inteligente puede guiar el proceso de instanciación, basado en la funcionalidad requerida para la aplicación que se está implementando.

Una de las limitaciones de este enfoque es que es altamente dependiente de la documentación disponible del *framework*: el agente no puede proveer asistencia sobre características no documentadas del *framework*. Para evitar sobrecargar de trabajo al diseñador y estimular la documentación de los *frameworks*, deben ser construidas herramientas y técnicas que faciliten el proceso de documentación. Algunos de los primeros pasos que se han dado en este sentido han sido el diseño de la notación *TOON*, así como la construcción del ambiente de documentación e instanciación *HiFi* [8].

5. Bibliografía

1. Amandi A., Zunino A., and Iturregui R. Multi-paradigm languages supporting multi-agent development. In Lecture Notes in Artificial Intelligence, MAAMAW'99. Springer-Verlag, 1999.
2. Bradshaw, J. Software Agents. AAAI Press, Menlo Park, USA. 1997
3. Demazeau, Y., Müller, J. (eds.). Decentralized AI – Proceedings of the First European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'89). Elsevier Science B.V. Amsterdam, Netherlands. 1990
4. Deuch, P. Framework and Reuse in the Smalltalk-80 System. En Software Reusability: Applications and Experience. Biggerstaf, Perlis (eds.) ACM Press, New York, 1989
5. Fayad M., Schmidt, D., Johnson R. Application Frameworks. En Building Application Frameworks. M.Fayad. D.Schmidt, R.Johnson (eds.) John Wiley and Sons, N.Y, 1999.
6. Johnson R. Documenting frameworks using patterns. In Proceedings of OOPSLA'92. ACM/SIGPLAN, New York, 1992.

7. Ortigosa A., Campo M. Using Incremental Planning to Foster Application Framework Reuse. To be published in Journal of Software Engineering and Knowledge Engineering (JSEKE). 2000.
8. Ortigosa A., Campo M. *SmartBooks*: A Step Beyond Active-Cookbooks to Aid in Framework Instantiation. Technology of Object-Oriented Languages and Systems 25, June 1999, IEEE Press. ISBN 0-7695-0275-X
9. Ortigosa A. Un Método para la Aplicación de Documentación Inteligente en la Instanciación de Frameworks Orientados a Objetos (in Spanish). PhD. thesis. Madrid, Spain, February 2000.
10. Soundarajan N. Understanding Frameworks. In Building Application Frameworks. M.Fayad, D.Schmidt, R.Johnson (Eds.) John Wiley and Sons, N.Y, 1999.