Título: "Generación automática de documentación que facilite la depuración de metaprogramas EFL"

Breve Descripción

Muchos autores consideran que el desarrollo de familias de productos, frente a la construcción individual de productos aislados, es un paso decisivo hacia la reutilización sistemática de software y la obtención de economía de alcance. En el Departamento de Ingeniería de Software y Sistemas Informáticos existe un grupo de investigación que se adscribe a esta corriente y propone un nuevo proceso de desarrollo de familias de productos, denominado EDD (*Exemplar Driven Development*), que aprovecha la similitud entre los productos de una familia para construirlos por analogía.

La primera actividad de EDD es la realización de un producto concreto de una familia. A continuación, se busca cómo flexibilizar este ejemplar para que satisfaga los requisitos del resto de los productos. Es decir, se trata de definir formalmente una relación de analogía que permita derivar del ejemplar los demás productos de forma automática. Por último, se obtienen los productos de la familia parametrizando la flexibilización del ejemplar.

Entre las aportaciones de EDD, cabe destacar:

- Abordar el desarrollo y el mantenimiento de una familia de productos mediante una estrategia sistemática e iterativa. Lo primero que se construye es un ejemplar que satisface los requisitos fijos de la familia. Después, se incorporan progresivamente capas de flexibilización que implementan los requisitos variables.

- Los requisitos fijos de una familia de productos suelen ser más estables que los requisitos variables. EDD separa la implementación de los requisitos fijos (el ejemplar) de la implementación de los requisitos variables (los módulos que flexibilizan el ejemplar).

- La decisión de elaborar una familia a menudo se toma al detectar trabajo repetitivo en el desarrollo aislado de varios productos de un dominio o al identificar oportunidades de negocio en la ampliación de las prestaciones de un producto de éxito. EDD reconoce esta situación y trata de aprovecharla mediante la reutilización íntegra de un ejemplar.

Actualmente, se han evaluado distintas maneras de flexibilizar un ejemplar aplicando las técnicas más comunes de generalización de código (herencia, genericidad, plantillas de código…). Lamentablemente, se ha comprobado que estas técnicas padecen limitaciones que impiden flexibilizaciones satisfactorias (que sean modulares, no invasivas, aplicables a cualquier producto software…). Para superar estas limitaciones, se ha propuesto el nuevo lenguaje EFL (*Exemplar Flexibilization Language*), que, de momento, está implementado como una librería del lenguaje orientado a objetos Ruby.

Con el fin de ilustrar la potencia y versatilidad de EDD y EFL, se han desarrollado diversos ejemplos de desarrollo de programas escritos en Java y C++; de procedimientos almacenados escritos en TRANSACT SQL; de juegos de prueba escritos en Java y Modula-2; y de documentación escrita en HTML y Javadoc.

Para facilitar la depuración de los metaprogramas escritos en EFL, sería deseable dotar a la actual implementación de EFL de la capacidad de generar documentación acerca de la ejecución de los metaprogramas.

Objetivos

El propósito de este proyecto fin de carrera es 1) determinar qué tipo de documentación sería deseable obtener y 2) ampliar la implementación de EFL en el lenguaje orientado a objetos Ruby para que sea capaz de generar automáticamente dicha documentación.

Método de desarrollo, fases de trabajo y fechas de realización

El desarrollo del proyecto fin de carrera durará un año y comprenderá las siguientes actividades:

1. **Análisis**. Identificación del tipo de documentación que sería deseable obtener para depurar metaprogramas EFL.
2. **Diseño**. Comprensión del diseño de la actual implementación de EFL y ampliación del mismo para que incorpore la capacidad de generar la documentación.

Los modelos de análisis y diseño resultantes de las actividades 1 y 2 seguirán el estándar *de facto* UML.

3. **Codificación**. Esta tarea se desarrollará en el lenguaje Ruby.
4. **Integración y Pruebas**. Las pruebas de unidades se automatizarán en la medida de lo posible, utilizando algún framework como *Ruby Test Unit*.
5. **Desarrollo de la memoria del proyecto fin de carrera**.

Medios a utilizar y breve justificación de la pertinencia de los mismos

El proyecto fin de carrera se realizará con herramientas de código abierto o gratuitas (Ruby, Jude para dibujar diagramas UML…). El resultado del proyecto será código abierto (con una licencia tipo BSD, *Berkeley Software Distribution*).

Información adicional

En http://www.issi.uned.es/miembros/pagpersonales/ruben_heradio/rheradio.html

puede encontrarse:

- Un video introductorio sobre la metodología EDD y el lenguaje EFL.

- La tesis “Metodología de desarrollo de software basada en el paradigma generativo. Realización mediante la transformación de ejemplares”, donde se documenta EDD y EFL.

- La implementación de EFL y diversos ejemplos de aplicación.