

DESARROLLO DE LÍNEAS DE PRODUCTO SOFTWARE MEDIANTE UN ENFOQUE GENERATIVO

Curso 2011/2012

(Código: 31105043)

1. PRESENTACIÓN

Aunque algunas estimaciones realizadas en los años 80, pronosticaban que el 60% de cualquier aplicación informática se desarrollaría ensamblando componentes reutilizables, el nivel de reutilización alcanzado hoy día es bastante inferior. Muchos autores consideran que este fracaso se debe a que la mayoría de los procesos de desarrollo de software, ya sean formales o ágiles, persiguen la construcción de productos aislados. Al no disponerse de contextos suficientemente amplios como para detectar con precisión qué elementos son reutilizables y cuáles son las situaciones donde puede sacarse más partido a la reutilización, se desemboca en una reutilización oportunista del software. Para que la reutilización del software fuera sistemática, los procesos de desarrollo deberían abordar la construcción colectiva de familias de productos relacionados por un dominio.

Otros autores han llegado a conclusiones similares al tratar de aplicar en la fabricación de software los principios de economía de escala y de alcance, comúnmente utilizados en la industria para reducir los costes y tiempos de fabricación y mejorar la calidad de los productos. La economía de escala se refiere a la fabricación de múltiples unidades de un mismo producto. Cuanto más se produce, menores son los costes. Se logra por diversas causas: reparto de los costes fijos entre más unidades producidas (disminución del coste medio), *rappel* sobre compras, mejora tecnológica, incremento de la racionalidad en el trabajo (especialización y división del trabajo)... La economía de alcance se da en la fabricación colectiva de productos similares. Se consigue principalmente porque los problemas comunes en la fabricación de los diversos productos se resuelven una sola vez. La fabricación industrial de un producto consta fundamentalmente de dos etapas: (i) la fase de desarrollo, donde se crean el diseño del producto y unos pocos prototipos para la validación del diseño; y (ii) la fase de producción, donde se crean de forma masiva instancias del producto. La economía de escala ocurre sobre todo durante la fase de producción. La naturaleza esencialmente lógica del software hace que los costes se concentren en la etapa de desarrollo (el coste de producir las copias de un sistema informático es despreciable comparado con el coste de desarrollo del sistema) y, por tanto, sea la economía de alcance el principio más aplicable en la fabricación de software.

En resumen, 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. Esta asignatura cubre las distintas fases de desarrollo de una familia de productos software.



2.CONTEXTUALIZACIÓN

La asignatura “desarrollo de familias de productos de software desde un enfoque generativo” se enmarca en el Máster Universitario en Investigación en Ingeniería de Software y Sistemas Informáticos, dentro del Módulo “Ingeniería de Software”. En particular, es una de las cinco asignaturas de la Materia “Ingeniería del desarrollo de Software”. A continuación se resume su relación con el resto de las asignaturas de dicha materia:

- a) La asignatura “Especificación de los Sistemas Software” trata la especificación formal de sistemas software. Los contenidos de esta asignatura se aplicarán en (i) el tema 2 de “desarrollo de familias de productos de software desde un enfoque generativo”, donde estudiaremos cómo especificar formalmente la variabilidad de una familia de productos, y (ii) el tema 3, donde se explicará como metamodelar un lenguaje específico de dominio.
- b) La asignatura “Arquitecturas para Sistemas Software” está íntimamente relacionada con “desarrollo de familias de productos de software desde un enfoque generativo”. De hecho, la arquitectura de una línea de productos software es la generalización de las distintas arquitecturas de cada producto particular.
- c) La asignatura “Generación Automática de Código” trata en profundidad la producción generativa de software. Dicha asignatura puede considerarse una ampliación de parte del contenido de los temas 4 y 5 del presente curso.
- d) La asignatura “Arquitecturas Orientadas a Servicios” presenta la reutilización de componentes software distribuidos en Internet. El principal objetivo de una línea de productos software es la reutilización sistemática de software. Por lo tanto, la relación entre ambas asignaturas es evidente.

3.REQUISITOS PREVIOS RECOMENDABLES

La formación previa que deberían tener los alumnos para el adecuado seguimiento de esta asignatura son los propios de ingreso al posgrado, haciendo especial recomendación en conocimientos de ingeniería de software, lenguajes de programación y compiladores.

4.RESULTADOS DE APRENDIZAJE

Los resultados de aprendizaje que se espera alcanzar con esta asignatura por parte del estudiante son:

- Comprender el impacto que tienen los conceptos de reutilización y abstracción en la producción de software.
- Aprender los principios metodológicos que guían el desarrollo de una línea de productos software, es decir, los fundamentos de la ingeniería de dominio e ingeniería de aplicación.
- Ser capaz de modelar mediante un diagrama de características el dominio de una línea de productos software.
- Conocer distintos enfoques para de desarrollar una familia de productos, distinguiendo (i) sus puntos fuertes y débiles, y (ii) las herramientas informáticas que los soportan.
- Ser capaz de implementar una línea de productos software mediante el lenguaje de programación Ruby.



5.CONTENIDOS DE LA ASIGNATURA

Tema 1: Introducción

Resumen: Este tema introduce el curso, presentando mediante un caso de estudio en qué consiste el desarrollo de una familia de productos software mediante un enfoque generativo y cuáles son sus ventajas respecto a la construcción de productos aislados.

Objetivos: (i) Comprender la importancia de los conceptos de reutilización y abstracción en la producción de software. (ii) Ofrecer una visión panorámica del curso.

1.1. Motivación:

1.1.a. Reutilización

- Familias de Programas
- Líneas de Productos Software (Software Product Lines, SPL)
- Factorías de Software (Microsoft)

1.1.b. Abstracción

- Programación Generativa (Generative Programming, GP)
- Desarrollo Dirigido por Modelos (Model Driven Development, MDD; Model Driven Architecture, MDA)

1.2. Ejemplo práctico de motivación: "Diccionarios en Java".

1.3. Organización del curso.

Tema 2: Aspectos metodológicos

Resumen: Este tema presenta los principios metodológicos que guían el desarrollo de una línea de productos software, es decir, los fundamentos de la ingeniería de dominio e ingeniería de aplicación.

Objetivos: (i) Distinguir las dos grandes fases de desarrollo de una línea de productos software. (ii) Aprender a modelar los aspectos comunes y variables de una familia de productos mediante diagramas de características.

2.1. Ingeniería de Dominio e Ingeniería de Aplicación.

2.2. Análisis de Dominio: Diagramas de características.

Tema 3: Lenguajes específicos de dominio

Resumen: Con el objetivo de facilitar la ingeniería de aplicación, la parametrización de una línea de productos suele realizarse mediante lenguajes específicos de dominio (DSL, Domain Specific Language). En este tema se estudiará cómo construir un DSL.

Objetivos: (i) Comprender las cualidades deseables de un DSL. (ii) Aprender cómo desarrollar un DSL.

3.1. Cualidades deseables de un Lenguaje Especifico de Dominio (Domain Specific Language, DSL).

3.2. Sintaxis y Semántica de un DSL:

3.2.a. Gramáticas BNF

3.2.b. Metamodelos

3.3. Cómo implementar un analizador para DSLs:

3.3.a. Metaparsers, Lenguajes de Transformación.

3.3.b. XML

3.3.c. DSLs embebidos en Lenguajes de Propósito General.



3.3.d. Herramientas de Metamodelado (DSL Tools de Microsoft, GME+EMF de Eclipse).

Tema 4: Implementación de la variabilidad de una línea de productos

Resumen: Durante el ciclo de vida de una línea de productos software, resulta vital gestionar convenientemente las características comunes y variables de todos los productos. Lamentablemente, no existe una técnica general para manejar la variabilidad de manera óptima, sino distintos mecanismos específicos para situaciones particulares.

Objetivos: Adquirir una visión panorámica de las distintas maneras de gestionar la variabilidad de software.

4.1. Motivación

- a. Cualidades deseables para una técnica de gestión de la variabilidad.
- b. Inexistencia de una técnica de "propósito general"

4.2. Técnicas internas: composición, herencia, polimorfismo, genericidad, orientación a aspectos.

4.3. Técnicas externas: generación de código.

Tema 5: Ruby, un lenguaje para la implementación de líneas de productos desde un enfoque generativo

Resumen: Ruby es un lenguaje que ofrece interesantes prestaciones generativas. Este tema introduce el uso de Ruby en metaprogramación.

Objetivos: Aprender a utilizar Ruby para desarrollar una familia de productos software mediante un enfoque generativo.

5.1. Justificación del uso de Ruby.

5.2. Análisis de DSLs con Ruby.

- 5.2.a. Metaparsers (racc, rookit)
- 5.2.b. XML
- 5.2.c. DSLs embebidos.

5.3. Generación de código con Ruby.

- 5.3.a. Librería de plantillas de texto ERB.

Tema 6: Enunciado del trabajo fin de curso

Resumen: Este tema presenta la estructura de los artículos científicos y propone el enunciado de un supuesto práctico que los alumnos deberán resolver aplicando los conceptos aprendidos en la asignatura.

Objetivos: Aplicar el temario del curso a un supuesto práctico que ejemplifique la bondad del desarrollo de líneas de productos software.

6.1. Enunciado.

6.2. Estructura y contenidos de un artículo científico.

6.3. Consulta de artículos desde la UNED.

6.EQUIPO DOCENTE

- [RUBEN HERADIO GIL](#)



7.METODOLOGÍA

La docencia de esta asignatura se impartirá a distancia, siguiendo el modelo educativo propio de la UNED adaptado al EEES. El principal instrumento docente será la web de la asignatura: <http://www.issi.uned.es/doctorado/generative/index.htm>, que incluye el siguiente material.

- Todas las referencias bibliográficas de la asignatura.
- La mayor parte de la bibliografía básica y complementaria de la asignatura.
- Videos introductorios de cada tema.
- Las diapositivas empleadas en los videos introductorios de cada tema.
- El código de los ejemplos utilizados en la asignatura.
- La plantilla de texto que deberán emplearse para redactar el trabajo del curso.

Además, el estudiante podrá realizar consultas al equipo docente a través del correo, teléfono y presencialmente en los horarios establecidos para estas actividades. También se podrán organizar videoconferencias si las necesidades docentes lo hicieran preciso.

8.BIBLIOGRAFÍA BÁSICA

Comentarios y anexos:

Tema 1:

- Heradio Gil, R. "Metodología de desarrollo de software basada en el paradigma generativo. Realización mediante la transformación de ejemplares". Ph. D. Thesis, Departamento de Ingeniería de Software y Sistemas Informáticos de la UNED, España, April 2007.

Tema 2:

- Capítulos 3 y 5 de Czarnecki, K. "Generative Programming Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models". Ph. D. Thesis, Department of Computer Science and Automation, Technical University of Ilmenau, October 1998.

Tema 3:

- Sección 3.2.2.2 de Heradio Gil, R. "Metodología de desarrollo de software basada en el paradigma generativo. Realización mediante la transformación de ejemplares". Ph. D. Thesis, Departamento de Ingeniería de Software y Sistemas Informáticos de la UNED, España, April 2007.
- Capítulo 8 de Greenfield, J.; Short, K. "Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools". Wiley, 2004.
- Fowler, M. "Language Workbenches: The Killer-App for Domain Specific Languages?" 12 Jun 2005. URL: <http://www.martinfowler.com/articles/languageWorkbench.html>

Tema 4:

- Capítulos 1, 2 y 3 de Pohl C. et al. "Survey of existing implementation techniques with respect to their support for the requirements identified in M3.2". AMPLE Consortium, Version 1.2, 7/30/2007. URL:<http://ample.holos.pt>.

Tema 5:



- Herrington, J. "Code Generation in Action". Manning, 2003.

Tema 6:

- R. Day. "How to Write and Publish a Scientific Paper". Cambridge University Press, 1989.

9. BIBLIOGRAFÍA COMPLEMENTARIA

Comentarios y anexos:

Tema 1:

- Czarnecki, K.; Eisenecker, U. "Generative Programming: Methods, Tools, and Applications". Addison-Wesley, 2000.
- Návrat, P. A closer look at programming expertise: critical survey of some methodological issues. In Information and Software Technology, no. 38, 1996, Elsevier, pp. 37-46.
- Greenfield, J.; Short, K. "Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools". Wiley, 2004.
- Stahl, T.; Voelter, M. "Model-Driven Software Development: Technology, Engineering, Management". Wiley, 2006.
- Mellor, S. J.; Scott, K.; Uhl, A.; Weise, D. "MDA Distilled. Principles of Model-Driven Architecture". Addison-Wesley, 2004.

Tema 2:

- Linden, F. J.; Schmid, K. Rommes, E. "Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering". Springer, 2007.

Tema 3:

- Aho A. V., Lam M. S., Sethi R.; Ullman J.D. "Compilers: Principles, Techniques, and Tools". Addison Wesley; 2nd edition (August 31, 2006).
- Parr, T. "The Definitive ANTLR Reference: Building Domain-Specific Languages". Pragmatic Bookshelf, May 17, 2007.
- Cook, S., Jones, G.; Kent, S. Wills, A. C. "Domain-Specific Development with Visual Studio DSL Tools". Addison-Wesley Professional (June 3, 2007).
- E. Visser. "Stratego: A language for program transformation based on rewriting strategies. System description of Stratego 0.5". In A. Middeldorp, editor, Rewriting Techniques and Applications (RTA'01), volume 2051 of Lecture Notes in Computer Science, pages 357-361. Springer-Verlag, May 2001. URL: <http://www.program-transformation.org/Stratego>

Tema 4:

- Capítulos 2 y 5 de Heradio Gil, R. "Metodología de desarrollo de software basada en el paradigma generativo. Realización mediante la transformación de ejemplares". Ph. D. Thesis, Departamento de Ingeniería de Software y Sistemas Informáticos de la UNED, España, April 2007.
- Heradio, R.; Cerrada, J. A. "Software Product Line Development by Analogy". Internacional Summer School, GTTSE (Generative and Transformational Techniques in Software Engineering). Braga, Portugal, July 2-7, 2007.
- Coplien J. O. "Multi-Paradigm Design for C++". Addison-Wesley, 1999
- Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. "Design Patterns: Elements of Reusable Object-Oriented Software". Addison Wesley, 1994.
- Alexandrescu, A. Modern C++ Design. Generic Programming and Design Patterns Applied. Addison-Wesley 2001.



- Batory, D. "A Tutorial on Feature Oriented Programming and the AHEAD Tool Suite (ATS)". September 8, 2004.

Tema 5:

Thomas, D.; Hunt, A. "Programming Ruby. The Pragmatic Programmers' Guide". Addison Wesley, 2nd edition (October 1, 2004).

10.RECURSOS DE APOYO AL ESTUDIO

Se ofrecerán las herramientas necesarias para que, tanto el equipo docente como el alumnado, encuentren la manera de compaginar el trabajo individual y el aprendizaje cooperativo (Skype, Moodle, Alf, etc).

11.TUTORIZACIÓN Y SEGUIMIENTO

La tutorización de los alumnos se llevará a cabo fundamentalmente a través de correo electrónico y consultas telefónicas los Jueves de 16:00 a 20:00:

- Rubén Heradio Gil: rheradio@issi.uned.es, 913988242
- José Antonio Cerrada Somolinos: jcerrada@issi.uned.es, 913986478

12.EVALUACIÓN DE LOS APRENDIZAJES

El conocimiento adquirido por los alumnos se evaluará mediante un trabajo práctico donde se aplicarán los conceptos estudiados en la signatura y cuyo enunciado aparecerá en el tema 6.

13.COLABORADORES DOCENTES

Véase equipo docente.

