COURSE GUIDE:Model-Driven Software Development (MDE)

Academic year: 2015-2016

Program: Master's program in Research and Innovation in Information and Communication Technologies (I^2-CIT)
Center: Escuela Politécnica Superior
University: Universidad Autónoma de Madrid

Last modified: 2015/18/05
Status: Approved June 8th 2015

1. ASIGNATURA / COURSE (ID)

Desarrollo de Software Dirigido por Modelos
Model-Driven Software Development (MDE)

1.1. Programa / program

Máster Universitario en Investigación e Innovación en Tecnologías de la Información y las Comunicaciones (I^2-TIC)

Master in Research and Innovation in Information and Communication Technologies (I^2-CIT) [Officially certified]

1.2. Course code

32432

1.3. Course areas

Software Engineering, Software Development, Software Modelling and Design, Programming Languages, Visual Languages, End user development

1.4. Tipo de asignatura / Course type

Optativa	[itinerario: Software Centrado en el Usuario]
Elective	[itinerary: Human-centered software development]

1.5. Semester

First semester

1.6. Credits

6 ETCS

1.7. Language of instruction

The lecture notes are in English. The lectures are mostly in Spanish. Some of the lectures and seminars can be in English. All the students' work can be presented in either Spanish or English.

1.8. Recommendations / Related subjects

Knowledge of the principles of object orientation, proficiency in object oriented programming languages like Java, development IDEs like Eclipse, and basic knowledge of modelling languages like UML are a prerequisite.

Related subjects are:

- Interacción Persona Ordenador [Human-Computer Interaction]
- Sistemas adaptativos y modelado de usuario [Adaptive systems and user modelling]

1.9. Lecturers

Add @uam.es to all email addresses below.

Lectures and labs:

Dr. Juan de Lara (Coordinator)
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Office: B-430
Tel.: +34 914972277
e-mail: Juan.deLara
Web: http://www.eps.uam.es/~jlara

Dra. Esther Guerra
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Office: B-419
Tel.: +34 914977476
e-mail: esther.guerra
Web: http://www.eps.uam.es/~eguerra

Dr. Jesús Sánchez Cuadrado
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Office: B-318
Tel.: +34 914972251
e-mail: jesus.sanchez.cuadrado
Web: http://sanchezcuadrado.es/

1.10. Objetivos de la asignatura / Course objectives

El desarrollo dirigido por modelos (DDM) permite la creación de aplicaciones y sistemas software a partir de modelos de alto nivel, mediante generación de código. En esta asignatura estudiaremos los fundamentos, técnicas, herramientas para la aplicación práctica del DDM. Estas técnicas incluyen el desarrollo de editores visuales y textuales para lenguajes de modelado, la transformación de modelos, y la generación de código.

Model-Driven Development (MDD) allows the construction of software systems from high-level models, by means of code generation techniques. In this subject, we will study the foundations, techniques and tools for the practical application of MDD. These techniques include those for the development of visual and textual editors for modelling languages, model transformations and code generation.

At the end of each unit, the student should be able to:

UNIT BY UNIT SPECIFIC OBJECTIVES	
UNIT 1.- Introduction to Model-Driven Software Development	
1.1.	To understand the usefulness of MDE, and its applicability scenarios.
1.2.	To understand the main concepts and principles of MDE.
1.3.	To know other software automation approaches, like Software Product Lines.
UNIT 2.- Software Modelling and Meta-Modelling	
2.1.	To review the basics of Software Modelling: structure and behavior.
2.2.	To review modelling notations, like the UML and OCL.
2.3.	To understand the concepts and techniques of meta-modelling, and to be able to apply them in practice using available technologies.
2.4.	To introduce advanced meta-modelling concepts, like multi-level meta-modelling.
UNIT 3.- Domain-Specific Modelling Languages	
3.1.	To understand the role of Domain-Specific Modelling languages in MDE, and the different techniques for Software Language Engineering.
3.2.	To know the different techniques for defining a visual concrete syntax to a language, and be able to apply them in practice, using available technologies.
3.3.	To know the different techniques for defining a textual concrete syntax to a language, and be able to apply them in practice, using available technologies.
UNIT 4.- Code Generation and Reverse Engineering	
4.1.	To understand the code generation principles, and how model-to-text transformations are realized.
4.2.	To know current frameworks and languages for code generation, and to apply them in practice.
4.3.	To understand the importance of software reengineering, and the different techniques, technologies and standards involved.
UNIT 5.- Model Transformations	
5.1.	To understand the importance and role of model transformations in MDE, and to be able to apply them in practice using current technologies.
5.2.	To understand and apply in-place and model-to-model transformation, to learn and apply specific languages for transformations.
5.3.	To learn the basics of a particular approach to transformation: graph transformation.

1.11. Course contents

1. Introduction to Model-Driven Software Development.
    a. Concepts.
    b. Technologies.
    c. Applications and Scenarios.

2. Software Modelling and Meta-Modelling
    a. UML.
    b. OCL.
    c. Meta-modelling.
    d. Multi-level meta-modelling.

3. Domain-Specific Modelling Languages
    a. Graphical Concrete Syntax
    b. Textual Concrete Syntax
    c. Defining DSMLs in practice: xText, Sirius.

4. Code Generation and Reverse Engineering
    a. Code generation principles: Model-to-text transformations.
    b. Code generation languages and tools.
    c. Reverse Engineering and Model-based modernization.

5. Model Transformations
    a. In-place Transformations.
    b. Model-to-Model Transformations.
    c. Transformation Languages and Tools in practice.
    d. Graph Transformation.

1.12. Course bibliography

- Bettini, L. "*Implementing Domain-Specific Languages with Xtext and Xtend*" (https://www.packtpub.com/application-development/implementing-domain specific-languages-xtext-and-xtend). Packt Publishing (2013).
- Brambilla, M., Cabot, J., Wimmer, M. "*Model-Driven Software Engineering in Practice*". Synthesis Lectures on Software Engineering, Morgan & Claypool Publishers 2012.
- Clayberg, E., Rubel, D. "*Eclipse Plug-ins*". Addison-Wesley Professional, 3rd Edition (2008).
- Ehrig, H., Ehrig, K., Prange, U., Taentzer, G. 2006. "*Fundamentals of Algebraic Graph Transformation*". Springer.
- Kelly, S., Tolvanen, J.-P. "*Domain-Specific Modeling. Enabling Full Code Generation*". IEEE CS, 2008.
- Raistrick, C. "*Model driven architecture with executable UML*". Cambridge University Press. 2004.
- Stahl, T., Völter, M. "*Model-Driven Software Development*". Wiley, 2006

- Steinberg, D., Budinsky, F., Paternostro, M., Merks, E. “*EMF: Eclipse Modeling Framework*”. Addison-Wesley Professional, 2nd Edition (2008).
- Völter, M. “*DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*”. dslbook.org 2013.

1.13. Coursework and evaluation

The course involves lectures with (at least) one assignment per unit (but not more than one per week), lab sessions and a practical research project. The assignments are to be solved individually. The research project is to be done in teams of two people, and involves the use of the studied technologies to solve a practical problem (of the student choice or proposed by the professors). The project is to be presented by the members of the team, who should also deliver a short document (in the form of a research paper) and the code of the project.

- In the ordinary exam period, the evaluation will be made according to the following scheme:
    - 40 % exercises.
    - 60 % final project.

- Both parts need to be passed separately (i.e., a grade >= 5 is needed in every part). If the student fails the exercises in the ordinary period, in the extraordinary period, the student has the opportunity to hand in all the exercises with corrections. However, an extension of the exercises will be requested.

- If the student fails the final project in the ordinary period, in the extraordinary period, the student has the opportunity to hand in an extended version of the project. The extension will be proposed by the professors.

- There are two modes of evaluation: continuous and non-continuous. The former implies attending classes regularly (≥70%), and implies the evaluation method just described. In non-continuous evaluation, the student is not required to attend classes. However, he still has to deliver the exercises and the project on the scheduled dates. In addition, he has to make the project individually.

- In both the exercises and the project, the students are expected to make original work. In particular, it is completely forbidden to copy part or all exercises from other students. In case of such behavior, the professors will apply the current regulations of the school regarding cheating.