The study guide for the course

University of Oviedo, Spain

1. Subject Identification

NAME	Software Architecture	***CODE***	
DEGREE	Software Engineering	***CENTER***	Computer Science Engineering School
TYPE	Compulsory	***E.C.T.S.***	6
PERIOD	Semester	***LANGUAGE***	English

COORDINATOR LECTURER	***PNONE/EMAIL***	***ADDRESS***
Jose Emilio Labra Gayo	labra@uniovi.es	Office. 206 – Edif. Fac. Ciencias Oviedo
OTHER LECTURERS	***PHONE/EMAIL***	***ADDRESS***
Aquilino Adolfo Juan Fuente	aajuan@uniovi.es	Office 197 – 3rd floor – Fac. Ciencias Oviedo
Begoña Cristina Pelayo García-Bustelo	crispelayo@uniovi.es	Office 198 – 3rd floor – Fac. Ciencias Oviedo
Jordán Pascual Espada	pascualjordan@uniovi.es	OOTLAB – 3rd floor – Fac. Ciencias Oviedo
Vicente García Díaz	garciavicente@uniovi.es	Office 196 – 3rd floor – Fac. Ciencias Oviedo

2. ContextDegree

This course is part of the Software Engineering degree. It is also part of the specific training module titled Software engineering. Other related subjects are *Software Process Engineering*; *Software Design*; *Software Quality, validation and Verifying*; *Requirements Engineering* and *Software Engineering Modelling*.

The subject is compulsory and is taught during the second semester of the third year of the Degree in Software Engineering.

It has 6 ECTS credits, which represent a total of 150 hours, 60 on-campus activities hours and 90 self-study hours.

From the organizational point of view, the subject will have two hours of lectures a week (for a total of 21 hours), one hour seminar (for a total of 7 hours), two hours of laboratory practice (for a total 28 hours), and 2 group tutorials hours.

3. Requirements

To take this course, it is recommended to have successfully acquired the learning objectives established in the subjects *Software Process Engineering* and *Software Design* of the first semester of the third year.

4. Competencies and Learning Outcomes

According to the *Verification Report* for the *Degree in Software Engineering* from the University of Oviedo, general competencies that students will acquire after studying the subject of Software Architecture are the following (the first column of the table shows the notation of each competency in the verification report):

General Competencies

CG-1	Ability to design solutions to human complex problems

Specific Competencies – Common to Computer Science

Com.1	Ability to design, develop, select and evaluate applications and systems, ensuring their reliability, safety and quality, according to ethical principles, laws and regulations.
Com.8	Ability to analyse, design, build and maintain applications in a robust, secure and efficient way, and choosing the most suitable paradigms and programming languages.
Com.11	Knowledge and application of features, functionality and structure of distributed systems, computer networks and the Internet, and to design and implement applications based on them.

Specific Competencies – Software Engineering Technology

ISW.1	Ability to develop, maintain and evaluate software systems and services that match all user requirements and behave reliably and efficiently, being affordable to develop and maintain and accomplishing quality standards, applying the theories, principles, methods and Software Engineering good practices.

ISW.3	Ability to solve integration problems in terms of strategies, standards and available technologies.
ISW.4	Ability to identify and analyse problems and to design, develop, implement, verify and document software solutions based on adequate knowledge of the theories, models and techniques.

Learning Outcomes

The learning outcomes that students will achieve after completing the course as stated in the *Verification Report* for the *Degree in Software Engineering*, are as follows (the first column of the table shows the notation of each learning outcome in the verification report):

RA.IS-1.	Making complex Software Engineering Projects that provide solutions to real problems and to solve them using techniques and technologies related to manufacturing processes, including software frameworks, architectural patterns, design and integration patterns, and quality software development
RA.IS-3.	To apply different construction techniques in designing low level software
RA.IS-4.	Develop design and object-oriented programming with a high level of competence
RA.IS-5.	To evolve and refactor existing designs to afford changing requirements
RA.IS-6.	Determining the degree of maintainability, reliability and efficiency of software designs
RA.IS-7	To design and implement software using different middleware technologies
RA.IS-9	To design and to carry out checks and efficient and effective inspections about validation, verification, quality and test plans
RA.IS-10	Statistically analysing the density of defects and failure probability
RA.IS-11	Evaluating the quality of a software process from the point of view of product quality

5. Syllabus

1. Software Architecture definition and basic concepts
    a. Software Architecture Introduction
    b. Describing Architectures
    c. Modelling Architectures

   d. Elements Related to Documentation
2. Software Architecture Taxonomies
   a. Introduction to Architecture Taxonomies
   b. Allocation: Building, deployment and distribution
   c. Modularity
   d. Behaviour: Components and connectors
   e. Integration
   f. Business architectures
3. Software Architecture based on Models
   a. Introduction to Architecture based on Models, Models and Metamodels
   b. Model Driven Architectures
   c. Specific domain and metamodel languages

6. Working plan and methodology

Course schedule:

Content (topics)	Study week	Lectures	Seminars / Laboratory
1. Software Architecture definition and basic concepts			
a. Software Architecture Introduction	Week 1	Online lecture 16th September 12:00 - 13:30	CASE Tool (*Seminar-Documentation*)
b. Describing Architectures	Week 2	Online lecture 23th September 12:00 - 13:30	UML (*Seminar-Video*)
c. Modelling Architectures	Week 3	Online lecture 30th September 12:00 - 13:30	Case of example (*Laboratory-Online*)
d. Elements Related to Documentation	Week 4	Online lecture 7th October 12:00 - 13:30	Building Architecture Documentation (*Seminar-Video*)
2. Software Architecture Taxonomies			
a. Introduction to Architecture Taxonomies	Week 5	Online lecture 14th October 12:00 - 13:30	Practical Task (*Laboratory-Online*)
b. Allocation: Building, deployment and distribution	Week 6	Online lecture 21th October 12:00 - 13:30	Practical Task (*Laboratory-Foro-Chat*)
c. Modularity	Week 7	Online lecture 28th October 12:00 - 13:30	Practical Task (*Laboratory-Foro-Chat*)

Content (topics)	Study week	Lectures	Seminars / Laboratory
d. Behaviour: Components and connectors	Week 8	Online lecture 4th November 12:00 - 13:30	Practical Task (*Laboratory-Foro-Chat*)
e. Integration	Week 9	Online lecture 11th November 12:00 - 13:30	Practical Task (*Laboratory-Foro-Chat*)
f. Business architectures	Week 10	Online lecture 18th November 12:00 - 13:30	Practical Task (*Laboratory-Online*)
3. Software Architecture based on Models			
a. Introduction to Architecture based on Models, Models and Metamodels c. Model Driven Architectures	Week 11	Online lecture 25th November 12:00 - 13:30	Practical Task (*Laboratory-Foro-Chat*)
d. Specific domain and metamodel languages	Week 12	Online lecture 2th December 12:00 - 13:30	BPM (*Video*) Practical Task (*Laboratory-Foro-Chat*)
4. Practical task			
a. Ending and documenting the work b. Building the team presentation	Week 13	Online 9th December 12:00 - 13:30	Practical Task (*Laboratory-Online*)
5. Assessment			
Practical Work, oral presentation	Week 14	Online 16th December 12:00 - 15:00	
Theoretical test	Week 14	Virtual Campus	

As stated in the IEEE, students will carry out face-to-face and self-study works and teachers will supervise these activities.

Teaching activities will be of one of these five types:

- Lectures, where to establish fundamental contents and where the student will be guide for their self-study activities.
- Workshops and seminars, to drive the student through active and collaborative learning, integrating lectures and virtual campus work.
- Laboratory practices, where to make different projects to solve any proposed problems. Individual and team projects will be done, requiring for students self-study work.
- Evaluation Sessions, examinations will be carried on in order to assess student acquisition of knowledge.

7. Learning material

Learning material consists of:

- scientific literature;
- additional interesting and useful literature;
- records of presentations and online consultations in virtual learning environment;
- practical tasks;
- real time chat, discussion forums and reflection blogs;
- learning guide for Software Architecture;
- video seminars and presentations.

Methodology

The student will have access to the following contents:

1. **Learning Guide**: For Software Architecture. It describes the basic theoretical elements that will be dealt with in the online lectures.
2. **Learning Objects**: Basically, the learning objects are the contents of the guide, along with some additional elements (videos, audio, etc.) and the self-assessment tools. These objects will be available in the Virtual Campus.
3. **Additional books and readings**: These are references to other additional learning contents that expand those of the Guide and the Learning Objects.

The working method of this subject is described below:

1. Before a theoretical online lecture (24 to 48 hours prior), the student will have to revise the theoretical contents in the Learning Guide or in the Learning Objects in the Virtual Campus. They may expand their knowledge (approx. 1 hour).
2. During the theoretical online lecture, these contents will be once again revised and related cases of usage will be analysed. The contents will also be expanded, based on the commented references (1 hour).
3. After the theoretical class, the student will have to revise again this document or the learning objects, and the additional references that have been marked as compulsory reading (the others are left to the judgment of the student). When a reading is compulsory, it will be marked with “COMP” in the Learning Guide. This revision will have to be completed before the first 76 hours after the theoretical class (approx. 4 hours).
4. Lastly, a 3-day period will be opened so that the student is able to take the corresponding self-assessment tests on the Virtual Campus (Half an hour).

Any student who doesn’t pass the test will have to start over this process from point 3. The first positive mark (over 5) will be their final mark on that topic, and all the successive positive evaluations will be deleted.

8. Learning Assessment

Assessment will be divided in two different aspects:

1. **Theoretical Assessment:** several exams (40% of final mark). It will take into account so the self-assessment tests (50%) as the final test (50%).
2. **Laboratory Assessment:** Individual and team projects (60% of final mark).

Final mark:

Mark = Theoretical Assessment * 0.40 + Laboratory Assessment * 0.60

9. Resources, bibliography and complementary documentation

Bibliography of Compulsory Reading

Albin, Stephen T. 2003. *The Art of Software Architecture: Design Methods and Techniques .* s.l. : John Wiley & Sons , 2003. ISBN:0471228869.

Bass, Len, Clements, Paul y Kazman, Rick. 2013. *Software Architecture in Practice.* 3. s.l. : Software Engineering Institute, Carnegie Mellon, 2013.

Clements, Paul, y otros. 2010. *Documenting Software Architecture.* Boston : Pearson Education, Inc., 2010. ISBN-13: 978-0-321-55268-6.

Garland, Jeff y Anthony, Richard. 2003. *Large-Scale Software Architecture: A Practical Guide using UML.* s.l. : John Wiley & Sons, LTD, 2003. ISBN: 0 470 84849 9.

Gorton, Ian. 2006. *Essential Software Architecture.* New York : Springer Berlin Heidelberg, 2006. ISBN-13 978-3-540-28713-1.

Additional interesting and useful literature

Albin, Stephen T. 2003. *The Art of Software Architecture: Design Methods and Techniques .* s.l. : John Wiley & Sons , 2003. ISBN:0471228869.

Ambler, Scott W. 2004. *The Object Primer: Agile Model-Driven Development with UML 2.0.* New York : Cambridge University Press, 2004. ISBN 0-521-54018-6.

ANSI/IEEE 1471. 2000. *Recommended Practice for Architectural Description of Software-Intensive Systems.* s.l. : ANSI/IEEE, 2000. ANSI/IEEE Std 1471-2000.

Architecture, Design, Implementation. **Eden, Amnon H. y Kazman, Rick. 2003.** Portland, OR : IEEE Computer Society Washington, DC, USA, 2003, ICSE'03 25th International Conference on Software Engineering , págs. 149-159. ISBN 0-7695-1877-X.

Bachmann, Felix, y otros. 2000. *Software ArchitectureDocumentation in Practice: Documenting Architectural Layers.* s.l. : Carnegie Mellon University, 2000. CMU/SEI-2000-SR-004.

Bass, Len, Clements, Paul y Kazman, Rick. 2013. *Software Architecture in Practice.* 3. s.l. : Software Engineering Institute, Carnegie Mellon, 2013.

—. 2003. *Software Architecture in Practice, Second Edition.* Boston : Addison Wesley, 2003. ISBN: 0-321-15495-9.

Beck, Kent. 1999. *Extreme Programming Explained: embrace change.* s.l. : Addison-Wesley Professional, 1999.

Big Ball of Mud. **Foote, Brian y Yoder, Joseph. 1997.** Monticello, Illinois : s.n., 1997. Vol. Fourth Conference on Patterns Languages of Programs. http://www.laputan.org/mud/.

Billy Reynoso, Carlos. 2004. *Introducción a la Arquitectura de Software.* Buenos Aires : Universidad de Buenos Aires, 2004.

Buschman, Frank, Henney, Kevlin y Schmidt, Douglas C. 2007. *Pattern Oriented Software Architecture: A pattern language for distributed computing.* s.l. : Wiley, 2007. Vol. 4.

Buschman, Frank, y otros. 2001. *Pattern Oriented Software Architecture: A system of patterns.* s.l. : Wiley, 2001. Vol. 1.

Camacho, Erika, Cardoso, Fabio y Núñez, Gabriel. 2004. *ARQUITECTURAS DE SOFTWARE - Guía de Estudio.* Caracas : Universidad Simón Bolívar, 2004.

Carmegie Mellon. 2012. Software Engineering Institute (SEI). *Duties, Skills, & Knowledge of a Software Architect.* [En línea] 2012. http://www.sei.cmu.edu/architecture/research/previousresearch/duties.cfm.

Clements, Paul, y otros. 2000. A practical method for documenting software architectures. [En línea] Carnegie Mellon University, 2000. http://www-2.cs.cmu.edu/afs/cs/project/able/ftp/icse03-dsa/submitted.pdf.

Clements, Paul, y otros. 2010. *Documenting Software Architecture.* Boston : Pearson Education, Inc., 2010. ISBN-13: 978-0-321-55268-6.

Clements, Paul, y otros. 2011. *Documenting software architectures.* 2011.

Cockburn, Alistair. 2005. Hexagonal Architecture. [En línea] 2005. http://alistair.cockburn.us/Hexagonal+architecture.

Cooper, Alan, Reinman, Robert y Cronin, David. 2009. *About Face 3: The Essentials of User Interface Design.* s.l. : Wiley, 2009. ASIN: B008NC0XR2.

Cooper, I. y Dilley, J. 2001. *HTTP Proxy/Caching Problems.* s.l. : IETF, 2001. Request for Comments. 3143.

Coplien, James A. 2010. *Lean Architecture: for Agile Software Development.* s.l. : Wiley, 2010.

Coplien, James O. y Bjornvig, Gertrud. 2010. *Lean Architecture for Agile Software Development.* s.l. : Wiley, 2010.

Erl, Thomas, y otros. 2012. *SOA with REST.* s.l. : Prentice Hall, 2012.

Etzion, Opher y Niblett, Peter. 2011. *Event Processing in Action.* s.l. : Manning, 2011.

Evans, Eric. 2003. *Domain-Driven Design: Tackling Complexity in the Heart of Software.* s.l. : Addison-Wesley, 2003.

Fairbanks, George. 2010. *Just enough Software Architecture.* s.l. : Marshall & Brainerd, 2010.

Fielding, Roy Thomas. 2000. *Architectural Styles and the design of network based architectures.* Tesis doctoral. s.l. : Universidad de California, Irvine, 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Fowler, Martin. 2010. *Domain-Specific Languages.* s.l. : Addison-Wesley Professional, 2010.

—. 2002. *Patterns of Enterprise Application Architecture.* s.l. : Addison-Wesley, 2002.

Frank, Buschmann, y otros. 1996. *Pattern Oriented Software Architecture.* Chichester, New York, Brisbane, Toronto, Singapore : John'Wiley & Sons Ltd., 1996. ISBN 0 471 95889 7.

Frankel, David S. 2003. *Model Driven Architecture. Applying MDA to Enterprise Computing.* Indeanapolis : Willey Publishing, Inc, 2003.

Garlan, David y Shaw, Mary. 1993. An Introduction to Software Architecture. [ed.] V. Ambriola y G. Tortora. *Advances in Software Engineering and Knowledge Engineering.* New Jersey : World Scientific Publishing Company, 1993, Vol. 1. http://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/intro_softarch.pdf.

Garland, Jeff y Anthony, Richard. 2003. *Large-Scale Software Architecture: A Practical Guide using UML.* s.l. : John Wiley & Sons, LTD, 2003. ISBN: 0 470 84849 9.

Gorton, Ian. 2006. *Essential Software Architecture.* New York : Springer Berlin Heidelberg, 2006. ISBN-13 978-3-540-28713-1.

Greer, Derek. 2007. Interactive Application Architecture. [En línea] 2007. http://aspiringcraftsman.com/2007/08/25/interactive-application-architecture/.

Hanmer, Robert. 2013. *Pattern Oriented Software Architecture for Dummies.* s.l. : For Dummies, 2013.

Hohmann, Luke. 1997. *Journey of the Software Professional: The Sociology of Software Development.* s.l. : Prentice Hall, 1997.

Hohpe, Gregor y Woolf, Bobby. 2004. *Enterprise Integration Patterns: Designing, building and deploying messaging solutions.* s.l. : Addison Wesley, 2004.

Humble, Jez y Farley, David. 2010. *Continuous Delivery.* s.l. : Addison-Wesley, 2010.

Ibsen, Claus y Anstey, Jonathan. 2011. *Camel in Action.* s.l. : Manning, 2011.

ISO. 1994. *ISO. Reference Model of Open Distributed Processing (RM-ODP).* s.l. : International Organization for Standardization, 1994. Technical Report 10746.

Kelly, Steven y Tolvanen, Juha-Pekka. 2008. *Domain-Specific Modeling. Enablign full code generation.* New Jersey, EEUU : John Willey & Sons, 2008.

Knoernschild, Kirk. 2012. *Java Application Architecture: modularity patterns with examples using OSGi.* s.l. : Prentice Hall, 2012.

Kruchten, P. 1995. *Architectural Blueprints - The "4+1" View Model of Software Architecture.* s.l. : IEEE, 1995. IEEE Software 12 (6).

Martin, Robert C. 2003. *Agile Software Devlopment: Principles, Patterns and Practices.* s.l. : Pearson Education, 2003.

—. 2008. *Clean Code: A Handbook of Agile Software Craftsmanship.* s.l. : Prentice Hall, 2008.

May, Nicholas. 2005. *A Survey of Software Architecture Viewpoint Models.* Melbourne, Australia : Swinburne University of Technology, Pages: 13–24, 2005.

Mellor, Stephen J, y otros. 2004. *MDA Distilled. Principles of Model-Driven Architecture.* Boston, EEUU : Pearson Education, Inc., 2004.

Norris, D. 2004. *Communicating Complex Architectures with UML and the Rational ADS.* s.l. : IBM, 2004. Proceedings of the IBM Rational Software Development User Conference.

OMG. Object Management Group. [En línea] http://www.omg.org/.

OMG-MDA. 2013. The Architecture of Choice for a Changing World. *OMG Model Driven Architecture.* [En línea] 2013. http://www.omg.org/mda/.

OMG-UML. 2012. Documents Associated with Unified Modeling Language (UML) Version 2.5. *"In Process version".* [En línea] Octubre de 2012. http://www.omg.org/spec/UML/2.5/Beta1/.

—. 2011. Documents associated with Unified Modeling Language (UML), v2.4.1. *OMG-UML.* [En línea] OMG-UML, Agosto de 2011. http://www.omg.org/spec/UML/2.4.1/.

—. 2005. OMG (Object Management Group). *Unified Modeling Language™ (UML®).* [En línea] OMG, 2005. http://www.omg.org/spec/UML/.

—. 1997. UML® Resource Page. [En línea] OMG - Object Management Group, 1997. http://www.uml.org/.

On the criteria to be used in decomposing systems into modules. **Parnas, David. 1972.** 12, 1972, Communications of the ACM, Vol. 15. http://www.cs.umd.edu/class/spring2003/cmsc838p/Design/criteria.pdf.

Qian, Kai, y otros. 2010. *Software architecture and design Illuminated.* s.l. : Jones and Batlet Illuminated series, 2010.

Qian, Kai, y otros. 2008. *Software Architecture and Design Illuminated.* s.l. : Jones and Bartlet Publishers, 2008.

RAE. Real Academia de la Lengua. [En línea] http://www.rae.es/rae.html.

Rasmusson, Jonathan. 2010. *The Agile Samurai.* s.l. : The Pragmatic Programmer, 2010.

Reynoso, Carlos y Kicillof, Nicolás. 2004. *Lenguajes de Descripción de Arquitectura (ADL).* Buenos Aires : Universidad de Buenos Aires, 2004.

Rotem-Gal-Oz, Arnon. 2006. Fallacies of Distributed Computing Explained. [En línea] 2006. http://www.rgoarchitects.com/Files/fallacies.pdf.

Rottem-Gal-Oz, Arnon. 2013. *SOA Patterns.* s.l. : Manning, 2013.

Soni, Dilip, Nerd, Robert L. y Hofmeister, Christine. Software Architecture in Industrial Applications. [En línea] http://www.users.abo.fi/lpetre/SA10/paper95.pdf.

Taylor, Richard N., Medvidovic, Nenad y Dashofy, Eric M. 2010. *Software Architecture. Foundations, Theory and Practice.* s.l. : Wiley, 2010.

Torre LLorente, César Luis, y otros. 2010. *Guía de arquitectura N-Capas orientada al dominio con .Net 4.0.* s.l. : Krasis Press, 2010.

uml-diagrams.org. 2010. UML Profile Diagrams. [En línea] 2010. http://www.uml-diagrams.org/profile-diagrams.html.

Vernon, Vaughn. 2013. *Implementing Domain Driven Design.* s.l. : Addison-Wesley Professional, 2013.

Wilder, Bill. 2012. *Cloud Architecture Patterns.* s.l. : O'Reilly, 2012.

Wojcik, Rob, y otros. 2006. *Attribute-Driven Design (ADD), Version 2.0.* s.l. : Software Engineering Institute, 2006. CMU/SEI-2006-TR-023 / ESC-TR-2006-023.