Course title: Practical Software Workshop

Course No.: 17818

School/College: Escuela Politécnica Superior

Degree: Undergraduate Degree in Computer Science and Engineering

# PRACTICAL SOFTWARE WORKSHOP LEARNING GUIDE

This learning guide is for the Practical Software Workshop course, part of the Bologna-Compliant Undergraduate Degree in Computer Science and Engineering, approved for the 2013/2014 academic year by the School Board. The final version of this learning guide was published on the web site of the Escuela Politécnica Superior, Universidad Autónoma de Madrid. The Practical Software Workshop learning guide, approved and published before the enrolment period, constitutes a learning agreement with the student.

# COURSE TITLE

PRACTICAL SOFTWARE WORKSHOP (STI-S)

## 1.1. Course number

17818 of the Undergraduate Degree in Computer Science and Engineering

## 1.2. Content area

Practical Computer Workshop

## 1.3. Course type

Compulsory subject

## 1.4. Course level

Undergraduate

## 1.5. Year

1st

## 1.6. Semester

1st (Fall semester)

## 1.7. Credit allotment

3 ECTS credits (1.25 class attendance + 1.75 non-class attendance)

## 1.8. Prerequisites

**PREVIOUS KNOWLEDGE AND SKILLS REQUIRED TO TAKE THE SUBJECT**

For students to be able to satisfactorily learn the contents, abilities and skills taught as part of the course, they should have good Spanish and English reading comprehension, literature search and location and Spanish report writing skills. Another requirement is for students to be favourably disposed towards learning and the development of a personal study discipline, as well as to team work.

**RECOMMENDATIONS**

To successfully pass the Practical Software Workshop, students are advised to simultaneously enrol for the Programming I course, where they will learn an algorithmic problem-solving approach and acquire knowledge of C.

Additionally, students should regularly attend practical, group, problem-solving, tool use and software systems building sessions.

Finally, students should have a good enough level of English to be able to understand recommended reading.

**CO-REQUISITES**

The Practical Software Workshop and the Practical Hardware Workshop courses are both part of the Practical Computer Workshop content area of the Bologna-Compliant Undergraduate Degree in Computer Science and Engineering.

The Practical Software Workshop is horizontally related to the Programming I course, because builds upon the knowledge, skills and attitudes learned in Programming I. The Practical Software Workshop provides the groundwork for the knowledge, skills and attitudes to be learned in the second-term Programming II and Programming Project courses. This group of courses lead to common learning outcomes, competences and contents.

## 1.9. Minimum attendance requirement

There are two methods: continuous evaluation method (with compulsory class attendance) and non-continuous evaluation method (without compulsory class attendance). Students will have to decide on one option at the start of the course (first two weeks) and meet the different evaluation requirements set for each model.

**CONTINUOUS EVALUATION METHOD WITH COMPULSORY CLASS ATTENDANCE**

Attendance of at least 85% of face-to-face classes is compulsory.

**NON-CONTINUOUS EVALUATION METHOD WITHOUT COMPULSORY CLASS ATTENDANCE**

Attendance of at least 30% of face-to-face classes is compulsory. Students must attend all the classes where practical assignments are submitted for partial oral evaluation.

## 1.10. Faculty staff

Note: add @uam.es to all the listed email addresses.

**Dr. David Camacho Fernández (Coordinator)**
Department of Computer Science
Escuela Politécnica Superior
Office – Module: B-443 Building B – 4th Floor
Phone: +34 91 497 2288
Email: David.camacho
Web Page: http://aida.ii.uam.es/researchers/facultystaff/camacho-david
Office hours: By email appointment.

## 1.11. Course objectives

The **computing-specific competences** acquired by students through the Practical Software Workshop are:

> C13. Knowledge and application of the tools required to store, process and access information systems, including web-based systems.

The **learning outcomes** that students achieve through the Practical Software Workshop are:

- Knowledge of different operating systems at user (advanced) and administrator (intermediate) level
- Ability to effectively and proficiently use programming environments, including code editing, compilation, assembly and debugging
- Ability to design and test a program to check for correctness,

efficiency, usability and reliability
- Good programming practices
- Use of some specialized computer applications
- Use of project management, release management, collaborative work tools, etc.

The general and specific objectives of the Practical Software Workshop are specified below.

| **GENERAL OBJECTIVES** | |
|---|---|
| G1 | Characterize the computer science discipline. |
| G2 | Describe computer structure and operation according to their two main components: hardware and software. |
| G3 | Identify, describe and relate the tasks performed by an operating system. |
| G4 | The use of UNIX environments. |
| G5 | Design, codify, debug, test, execute and interpret programs in the C procedural programming language. |
| G6 | Apply modularization as a specific technique for structuring programs. |
| G7 | Use Linux-based programming environments. |
| G8 | Take an interest in and the initiative to search, organize and critically analyse key information as a means of learning and decision making to achieve the objectives specified during the implementation of the participatory techniques and group practical assignments. |
| G9 | Actively participate in group analyses and discussions, establish the thread of program development, cooperate with peers on the development of team work and communicate both in writing and orally. |

| SPECIFIC OBJECTIVES |
|---|
| **UNIT 1: INTRODUCTION TO COMPUTING** |
| • Define key computer science concepts.<br>• Describe computer structure and operation.<br>• Explain the role played by computer software. |
| **UNIT 2: BASIC CONCEPTS ABOUT OPERATING SYSTEMS** |
| • Define the concept of operating system<br>• Explain the evolution of operating systems<br>• Identify and list operating system tasks<br>• Describe the most common UNIX user commands<br>• Apply the most common UNIX environment user commands<br>• Use an editor/Emacs/Vi (editor) + gcc + make (compiler) + ddd (debugger) C programming environment running on Linux |
| **UNIT 3: GETTING STARTED WITH PROGRAMMING USING IDE** |
| • Definition of basic concepts about Visual and Interactive programming.<br>• Introduction to Integrated Development Environment (IDE).<br>• Programming application by the use of IDEs.<br>• Validating, debugging and executing of application using IDEs. |
| **UNIT 4: ADVANCED USE OF PROGRAMMING TOOLS** |
| • Interpret the operation of programs that include structures, types definition, functions, pointers, tables, dynamic memory management and organization of C code in a programming environment running on Linux<br>• Use dynamic memory management tools running on Linux<br>• Program, execute, interpret and test programs written in the C procedural programming language<br>• Document software systems written in the C programming language<br>• Apply structured and modular programming using the C programming language<br>• Develop key negotiating skills for developing software systems. |

## 1.12. Course contents

**PROGRAMME OUTLINE**

UNIT 1. INTRODUCTION TO COMPUTER SCIENCE.
UNIT 2. BASIC CONCEPTS ABOUT OPERATING SYSTEMS.
UNIT 3. GETTING STARTED WITH PROGRAMMING USING IDE.
UNIT 4. ADVANCED USE OF PROGRAMMING TOOLS.

**DETAILED PROGRAMME**

1. **INTRODUCTION TO COMPUTER SCIENCE**
   1.1. Basic concepts.
   1.2. Computer structure and operation.
   1.3. Computer software.

2. **BASIC CONCEPTS ABOUT OPERATING SYSTEMS**
   2.1. Operating system definition.
   2.2. Most popular operating systems (Windows, UNIX and Linux)
   2.3. Operating system tasks: processor management, memory management, I/O management and file management.
   2.4. UNIX and Linux environments.
   2.5. Programming environments running on Linux.

3. **GETTING STARTED WITH PROGRAMMING USING IDE**
   3.1. Definition of basic concepts about Visual and Interactive programming.
   3.2. Introduction to Integrated Development Environment (IDE).
   3.3. Programming application by the use of IDEs.
   3.4. Validating, debugging and executing of application using IDEs.

4. **ADVANCED USE OF PROGRAMMING TOOLS**
   4.1. Dynamic memory management tools.
   4.2. Program structuring and modular programming.

## 1.13. References

The learning resources of each unit are detailed below. We distinguish between basic and recommended references.

**UNIT 1: INTRODUCTION TO COMPUTER SCIENCE**

Basic References:

- A. PRIETO, A. LLORIS, J. C. TORRES. *Introducción a la Informática*. 4ª ed. (McGraw Hill, 2006). Cap. 1: Introducción.

Recommended References:

- J. G. BROOKSHEAR. *Computer Science: An Overview*. 9ª ed. (Pearson International Edition, 2007).

**UNIT 2: BASIC CONCEPTS ABOUT OPERATING SYSTEMS**

Basic References:

- E. ANGUIANO, D. CAMACHO, C. NAVARRETE. *LINUX, Guía de Aprendizaje*. Prentice Hall, 2008

Recommended References:

- J. GARCÍA de JALÓN, I. AGUINAGA, A. MORA. *Aprenda LINUX como Si Estuviera en Primero*. (Universidad de Navarra, 2000).

- B. W. KERNIGHAN. *El Entorno de Programación UNIX*. (Prentice Hall Hispanoamericana, 1987).

**UNIT 3: *GETTING STARTED WITH PROGRAMMING USING IDE***

Basic References:

- Netbeans. C/C++ Application Learning Trail. http://netbeans.org/kb/trails/cnd.html
- Configuring the NetBeans IDE for C/C++. http://netbeans.org/community/releases/68/cpp-setup-instructions.html
- Eclipse for C/C++ Developers. http://www.eclipse.org/callisto/c-dev.php

**UNIT 4: ADVANCED USE OF PROGRAMMING TOOLS**

Basic References:

- B. W. KERNIGHAN, D. RITCHIE, *The C Programming Language*. 2nd ed. (Prentice Hall, 1988).

Recommended References:

- B. W. KERNIGHAN. *La Práctica de la Programación*. (Pearson Educación, 2000).
- H. SCHILDT. *C: Guía de Autoenseñanza*. (Osborne/McGraw Hill, 2001).
- A. SIERRA URRECHO. *Programación en C/C++*. (Anaya Multimedia, 2005).

**Please Note:** This course does not follow any particular book. It is not recommended to buy any book until you have compared its contents with the program and previously reviewed it in the library.

**Online documentation:** online documents (published in Moodle in STI course page (http://uam-virtual.es)

## 2. Teaching methods

The Practical Software Workshop course is an eminently practical course. All classes will have a strong practical content, where the teacher will briefly describe the main theoretical content necessary for the development of course modules. To achieve the planned objectives and capabilities (see section 1.11), the course is organized into practical sessions in computer labs, where most of the work must be done in small groups.

**LABORATORY SESSIONS**

**Teacher's activity:**

At the beginning of the session, the teacher will assign to each team the work to be done in the laboratory. Also, the teacher will supervise the work of the teams during the session. During the laboratory sessions the following activities will be done:

- Explain and provide the complete framework of the work to be done outside the laboratory.
- Reading and comprehension of course material.
- Analysis and development of case studies.
- Help for learning tools, including tutorials.
- Supervise the work of the working groups in the laboratory.

The material to be used is the software installed in the computers labs and the computers available for the implementation and simulation of programs and learning materials published online.

**Student's Activity:**
*Classroom Activity*: Depending on the developmental stage of the session in question, students must: i) understand the explanations of the techniques employed, ii) work in groups to apply these techniques to the development of the project, iii) write reports on the results obtained. Some sessions will require the presence of the teacher during the execution of the task, who will make appropriate questions to each group member to qualify individually the lab practice.
*Activity outside the classroom*: Meetings with your work group mates in order to finish the work in question. Writing the report of the practice when it is asked for.

**TUTORSHIPS IN THE CLASSROOM**

**Teacher's activity:**
The office hours are organized to (a) doing a personalized monitoring of work groups and resolve conflicts among members; either to (b) resolve common questions raised by the students or the groups, related to the project to be done.

**Student's Activity:**
*Classroom activity:* Approach to individual or group tasks questions and possible solutions to the working tasks.

**TEAM MEETINGS**
**Student's Activity:**
*Activity outside classroom*: face-to-face and online meetings between team members to integrate the partial results achieved by each member.

**REQUIRED READINGS AND INDIVIDUAL STUDY**
**Student's Activity:**
*Activity outside the classroom:* Autonomous learning academically directed by the teacher through the tasks posted on the course website.

## 3. Student work time

| | | Number of hours | Percentage |
|---|---|---|---|
| Classrom | Theoretical Classes | 0 h (0%) | 45% = 34 hours |
| | Practical classes | 28 h (37%) | |
| | Mentoring scheduled throughout the semester | 3 h (4%) | |
| | Examinations | 3 h (4%) | |
| Outside the Classroom | Practical activities | 18 h (24%) | 55% = 41 hours |
| | Weekly study **(0,5 h x 14 weeks)** | 7 h (10%) | |
| | Exam preparation | 16 h (21%) | |
| **Total work hours: 25 hours x 3 ECTS** | | **75 h** | |

## 4. Methods of evaluation and the final grade percentage

**EVALUATION CRITERIA**

The project will be evaluated in 4 assessments, in which the student must submit the required work. The Final Score (FS) of the course is given by the following formula weighted by the four assessments for the case of the students who opted for continuous evaluation method:

**FS = 20% x A1 + 25% x A2 + 25% x A3 + 25% x A4 + 30% x A4**

or by 40% of the weighted grade of the four practices and 60% of the Final Exam (FE), which involves all the subjects developed during the course, and to be held in January for students that don't choose continuous evaluation:

**FS = 60% FE + 40% (20% x A1 + 25% x A2 + 25% x A3 + 25% x A4 + 30% x A4)**

If the student will not submit at least two assessments, it will receive "not evaluated". If a student changes from the continuous evaluation method to the non-continuous evaluation method, he/she can get a maximum of 5 in their final grade.

If the student does not pass the evaluation, for the following one, he/she must submit all the practices correctly and the final exam will be held in June of 2013.

**Please Note**: The Practical Information Workshop Course (PIWC) can be approved with a minimum grade of **5**. The final grade of Practical Computer Workshop Course will be calculated by the average of final grades of Practical Software Workshop Course (STI-S) and Practical Hardware Workshop Course (STI-H). The student must obtain a minimum grade of 5 in STI as in STI-S-H, to be able to calculate the final grade as explained before. Otherwise, the final grade of PIWC will be: Minimum 50% x (5, STI-S) + 50% x minimum (5, STI-H).

**GUIDELINES**

Student's work shall be governed by the following rules:

1. There are four assessments over the course that students must submit and/or mandatory evaluations to pass the course.
2. You must have at least **4.5** in all assessments for the previous weighted formula to be applied. The practices will be evaluated on a scale of 0 to 10.
3. To approve all practice it is required to do:
   3.1. The evaluation of individual performance of Assessment 1.
   3.2. The evaluation of individual performance of the Assessment 2.
   3.3. Deliver the source code of Assessment 3 tasks. It also will be evaluated with respect to:
      - Functional correction: correct response to input data, whether are normal, exceptional or incorrect.
      - Code Organization and quality.
      - Deliver a report of Assessment 3 of this course, as specified later in this document.
   3.4. Deliver the source code of Assessment 4 tasks. It also will be evaluated with respect to:
      - Functional correction: correct response to input data, whether are normal, exceptional or incorrect.
      - Code Organization and quality.
      - Deliver a report of Assessment 4 of this course, as specified later in this document.
4. In the documents that are found on the course Web Page it will be published the due dates for the all the assessments.
5. Any assessment that is submitted and the source code does not compile or there are runtime errors will be penalized with 40% of the final assessment grade.
6. An assessment that is not submitted means that the student failed the course.
7. For each day of delay in submitting, the student will be penalized with 10% of the final practice grade.
8. The composition of teams will be established on the first day and will not be modified except in exceptional circumstances.
9. The grades given to members of a team for the same assessment may be different when the teacher finds justification for doing so.

**PLEASE NOTE**: Any copy that was discovered, both within any of the theoretical or practical activities developed, will be penalized with rigor. The penalty for copying

involves the application of the internal regulations of the EPS, which is to suspend the current call and cannot present in the next one.

**ASSESSMENT REPORT**

For some assessments, students will be asked to deliver a report. This report must contain:

1. Students' name, team identification, group, date and assessment number.
2. Assessment description.
3. Code description, or how students have developed the delivered code.
4. Interesting points to be taken into account by the teacher.
5. Results discussions

**DELIVERIES**

Each deliver will consist on a compressed file (.zip or .rar) that includes the following:

- Source code of all programs develop for the assessment (.c files)
- Headers files (.h files)
- Makefile.
- Readme file that contains authors' information and a short description for launching the program.
- Report describing the development assessment process.

Executable files (*.o) and project files don't have to be delivered. Each file must have a descriptive name that indicates its function. File extensions will be the common ones: .c for source code file and .h for headers files.

The name of the compressed file must be *PxTyEz.zip* where *x* is the number of the assessment, *y* is the group and *z* is the team identification. For example, if team 6 (that belongs to group 164) delivers the code of Assessment 3, the team would upload the zip file with the name *P3T164E6.zip*.

The evaluation of the different assessment will be published on Moodle platform (http://uam-virtual.es) of Escuela Politécnica Superior.

## 5. Schedule

| Week | Content | Classroom hours | Hours outside the classroom |
|---|---|---|---|
| 1 | - Subject presentation and student team formation. | 2 | - |
| 2 | - **Unit 1 Introduction to Computer Science** and Assessment 1 presentation. | 2 | 2<br>- Read Unit 1 proposed documents |
| 3 | - Workshop on Assessment 1. | 2 | 2<br>- Develop Assessment 1<br>- Assessment 1 delivery |
| 4 | - **Unit 2 Basic Concepts about Operating Systems** and<br>Assessment 2 presentation. | 2 | 2<br>- Read Unit 2 proposed documents<br>- Develop Assessment 2 |
| 5 | - Workshop on Assessment 2. | 2 | 2<br>- Develop Assessment 2 |
| 6 | - Workshop on Assessment 2. | 2 | 2<br>- Develop Assessment 2<br>- Assessment 2 delivery |
| 7 | - **Unit 3 Basic use of programming tools with IDE** and Assessment 3 presentation. | 2 | 2<br>- Read Unit 3 proposed documents<br>- Develop Assessment 3 |
| 8 | - Workshop on Assessment 3. | 2 | 2<br>- Develop Assessment 3 |
| 9 | - Workshop on Assessment 3. | 2 | 2<br>- Develop Assessment 3 |
| 10 | - Workshop on Assessment 3. | 2 | 2<br>- Develop Assessment 3<br>- Assessment 3 delivery |
| 11 | - **Unit 4 Advance use of programming tools** and Assessment 4 presentation. | 2 | 2<br>- Read Unit 4 proposed documents<br>- Develop Assessment 4 |

| Week | Content | Classroom hours | Hours outside the classroom |
|---|---|---|---|
| 12 | - Workshop on Assessment 4. | 2 | 2<br>- Develop Assessment 4 |
| 13 | - Workshop on Assessment 4. | 2 | 2<br>- Develop Assessment 4 |
| 14 | - Workshop on Assessment 4. | 2 | 2<br>- Develop Assessment 4<br>- Assessment 4 delivery |