

**towards Open Source Software adoption and dissemination
tOSSad**

Contract No 015981

F/OSS Curriculum Report

D22

Version 1.5

13 March 2007



Copyright

This document is Copyright © 2007 by its contributors as listed in the section titled “version control”. You can distribute it and/or modify it under the terms of either the GNU General Public License, version 2 or later (<http://www.gnu.org/licenses/gpl.html>), or the Creative Commons Attribution License, version 2.0 or later (<http://creativecommons.org>).

All trademarks within this guide belong to their legitimate owners.

Version control

Version	Date	Author	Organization	Description
0.1	18/11/06	Michele Marchesi	UCA	First version, table of content
0.2	16/12/06	Grassi Simone Selahattin KURU	TCD IOTA	Editing & Section 4.5 Sections 1, 2 (2.1, 2.2)
0.3	12/01/07	Grassi Simone Enn Öunapuu	TCD TUT	Editing Section 6
1.0	16/01/07	Grassi Simone Michele Marchesi Görkem Çetin Selahattin KURU	TCD UCA UEKAE IOTA, UILBIN	Editing Section 3 and 10 Section 5 Section 7,8
1.1	25/01/07	Stephen Barrett	TCD	Section 4 (4.1 to 4.4), 9
1.2	04/02/07	Görkem Çetin	UEKAE	Editing
1.3	07/02/07	Grassi Simone	TCD	Editing
1.4	21/02/07	Grassi Simone	TCD	Editing
1.5	13/03/07	Kaan Erkan	UEKAE	Quality

Release approval

Name	Role	Date
Kaan Erkan	Coordinator	13/03/07

Abbreviations

EU: European Union

F/OSS: Free/Open Source Software

GPL: GNU-General Public License

IT: Information Technology

LAMP: Linux, Apache, Mysql, PHP

LAN: Local Area Network

SME: Small and Medium Enterprise

tOSSad: towards Open Source Software adoption and dissemination

VoIP: Voice over IP

Table of Contents

1. Introduction.....	6
2. Needs analysis for F/OSS.....	7
2.1. How to perform training needs analysis.....	8
Assessment tools and techniques.....	10
2.2. The tOSSad Questionnaire.....	10
3. F/OSS in Vocational Curricula.....	11
3.1. Needs that are satisfied and goals of the curricula.....	11
3.2. Prerequisites to enroll.....	12
3.3. Description of the professional figure.....	12
3.4. Characteristics of F/OSS vocational curricula.....	13
3.5. Specific curricula.....	14
4. F/OSS in the 3rd and 4th Level setting.....	15
4.1. The feasibility of F/OSS course provision.....	17
4.2. Quantifying Fourth Level F/OSS delivery.....	18
4.3. A F/OSS MSc Curriculum Approach.....	19
4.4. The MSc Subject Set.....	20
Software Engineering.....	20
Networking and Distributed Systems.....	21
Information Management and Security.....	22
Programming and Design.....	23
Ubiquitous and Pervasive Computing.....	24
Artificial Intelligence.....	25
High Performance Computing.....	26
Vision and Graphics.....	27
4.5. Core F/OSS Options.....	28
4.5.1. Licensing and Legal Issues.....	28
4.5.2. Development Methodologies.....	31
4.5.3. Business Models.....	32
5. Professional learning / professional certification.....	33
6. Training methods	35
6.1. Open Source learning.....	35
6.1.1. Open CourseWare.....	35

6.1.2. Open Knowledge Initiative.....	36
6.2. New possibilities of learning introduced by information technologies.....	36
6.2.1. Distance learning and e-learning.....	37
6.2.2. Cooperative learning through the Internet.....	39
6.3. The obstacles of e-learning	39
6.4. Adaptive e-learning environment.....	41
7. Appendix 1: Needs analysis questionnaire.....	43
8. Appendix 2: Needs analysis questionnaire results.....	51
9. Appendix 3: TCD Academic Questionnaire and Results.....	58
10. Appendix 4: Example of Vocational curriculum: F/OSS system specialist.....	61
10.1. Course list and programs.....	62
10.1.1. Vocational Curriculum Core Block.....	63
10.1.2. Vocational Curriculum Specialized Block.....	66
10.1.3. Vocational Curriculum Practical Work	70
11. References.....	71

1. Introduction

This document reports on the tOSSad project's efforts towards the development of F/OSS Curricula suitable for delivery in a range of educational settings. The need for F/OSS curricula arise for two principal reasons:

- Shortage of workforce skilled in F/OSS
- Lack of components for basic knowledge on F/OSS in current curricula

Shortage of workforce skilled in F/OSS:

The need for professionals competent in differing aspects of F/OSS has increased significantly in the recent years. In other words there is a shortage in skilled workforce in the F/OSS area. This is a result of the success of F/OSS especially in the commercial market in the recent years.

Need for professionals competent in F/OSS arises not only in technical aspects but also in non-technical aspects (legal issues, business model, organisation, etc.) of F/OSS. In other words such curricula should be developed both as technical programs and as non-technical programs. Educational institutions should develop and implement curricula addressing this specific need. This includes curricula to be developed by institutions in the formal education system, curricula to be developed by professional training organizations, and curricula delivered in settings for life-long learning.

Lack of components for basic knowledge on F/OSS in current curricula:

Despite the widespread use of F/OSS in educational settings, it is very common for the delivery of curricula to make use almost exclusively of proprietary software, and F/OSS is rarely mentioned as a topic of study. This is a weakness of the traditional curricula. Lack of basic knowledge on F/OSS is a concern for the graduates of these programs, both from a professional point of view and from a cultural point of view. That means that educational institutions should adopt their curricula to include components to give basic knowledge on F/OSS. Again, this need arises not only in technical programs but in fact in all programs. In other words existing curricula should be adopted to include F/OSS components in all programs. Of course different programs should enhance their programs to include those F/OSS topics that are relevant to their disciplines.

This report examines ways to redress this imbalance, and provides educational models for the study of F/OSS.

The F/OSS movement is some 30 years old. In these years, a large body of knowledge has accumulated, covering various aspects of F/OSS. There is a sufficient subject of study for many educational institutions to provide specialised F/OSS degree programs. In other cases it is clear that greater emphasis on F/OSS material in the curricula is needed.

Retraining of current workforce on F/OSS is an approach to address both of the above needs, that is to produce skilled workforce in F/OSS and to provide basic knowledge on F/OSS to the current workforce.

New methods in learning are promising in training in F/OSS, including e-learning,

work-based learning, and social (community based) learning.

In the following sections we report on the potential for F/OSS study in professional training settings and in 3rd and 4th level settings. Implicit in our analysis is the assumption that F/OSS study can be divided into two main categories:

- Core F/OSS knowledge areas
- Specialized knowledge areas

Core F/OSS knowledge areas include subjects which are considered as basic for the understanding of the F/OSS phenomenon. Every F/OSS curriculum should cover the core knowledge areas, even if the level of detail and formalism may differ from curriculum to curriculum.

Specilized knowledge areas include subjects for which F/OSS is not the central topic of study, but for which it is possible to provide an emphasis for F/OSS solutions and technologies over proprietary solutions. There is of course a practically unlimited number of possible fields of study, particularly in 4th level. However, in practise most programs are for a set of specialised knowledge areas and it is possible to catalogue these and address F/OSS issues for each.

Thus, we anticipate that there is no single F/OSS curriculum, but rather take the view that F/OSS study requires a multifaceted approach which seeks to introduce the subject into mainstream educational settings. It is the setting that determines whether the resulting curriculum are to a greater or lesser extent F/OSS in character, but it is our analysis that there is little in our educational settings that cannot be enhanced substantially by a consideration of F/OSS.

We devide our report as follows. In section 2 we report on a needs analysis process conducted in the project to establish a requirement for F/OSS skills. In section 3 we address the professional training setting. In section 4 we explore 3rd and 4th level settings. In section 5 we review the issue of professional certification before discussing training methods potentially applicable to all settings in section 6.

2. Needs analysis for F/OSS

It is widely recognized that a systematic assessment of training needs must always be conducted in order to ensure that training programs and interventions have relevance to the people being trained in particular business settings [5]: A working definition asserts, *“Needs assessment can be described as a process for identifying the knowledge and skills necessary for achieving organizational goals”*.

The tOSSad project aims to develop curricula for F/OSS training. In order for this goal to be accomplished, systematic work needs to be conducted towards identification and prioritisation of training needs of European SMEs in F/OSS and identification of the content of the training courses that correspond to selected needs.

A systematic assessment of training needs must be conducted especially for the following reasons:

- SMEs are operating in different industries, markets, countries/regions with different missions and goals and different needs.
- A training needs assessment methodology ensures that training programs and interventions have relevance to the people being trained in F/OSS.
- After developing and using certain training needs assessment methodology, generic and specific curricula can be developed for training in F/OSS.

2.1. How to perform training needs analysis

Needs assessment usually involves assessing the need for training at three different levels [6]: organization, task, and person. Organizational analysis involves examining organizational factors and constraints to determine whether the training provided to employees will actually transfer back to the job [7]. Task analysis involves an examination of the tasks that comprise a job and determining the knowledge, skills, and abilities needed to perform those tasks [8]. Person analysis involves identifying which employees are in need of training and in which areas they need to be trained [9].

As a result of training needs assessment at each of these three levels, the training needs of employees may be determined by uncovering any discrepancies between the knowledge and skills needed for a job and the knowledge and skills possessed by the employees assigned to that job. These training needs are then used to develop training objectives during the design phase of the training process. These objectives are used during the development phase to devise a training plan that includes the specific content of the training, as well as instructional methods, materials, and equipment. The training plan developed during the development phase is used during the implementation phase to provide the training to the trainees.

Training needs assessment models

In the literature two substantive models have been offered to direct needs assessment: one by McGehee and Thayer in 1961 [6] and the other by Ostroff and Ford in 1989 [10]. For over 30 years, both managers and researchers applied McGehee and Thayer's approach to needs assessment. This approach identifies three "content" areas: organizational, task, and person. However, as stated in Nelson et al. [11], most applications and investigations of needs assessment have emphasized only the person component, ignoring the critical roles of the organizational and task components. Trying to avoid such a monolithic focus, a lot of emphasis was given on organizational and task components during the needs assessment phase of the project.

In an attempt to broaden the narrow focus of practice and research, Ostroff and Ford expanded on McGehee and Thayer's approach, developing a "content levels" framework, based on general systems theory, that incorporates organizational, subunit, and individual levels into identification of training content in the needs-assessment process. Although this framework has the potential to contribute significantly to practice and research on needs assessment, it has not been adequately tested or applied [12]. Nelson et al. applied the framework and presented a case study that extends traditional research on needs assessment from the individual level to the task and organizational levels.

The Content-Levels Framework

The theoretical foundation of Ostroff and Ford's framework combines the dual perspectives of content (person, task, and organizational) and levels (individual, subunit, and organizational) to form a nine-cell matrix (see Table 1). The intersection of each content factor with each level suggests questions or issues that should be addressed in a more comprehensive approach to needs assessment.

Level	Content		
	Individual	Task	Organizational
Individual	What knowledge and specific skills individuals need to learn for effective performance?	What are the knowledge and skill requirements for the accomplishment of specific tasks by an individual?	How do the goals of an individual effect or constrain performance?
Unit	What skill mix is needed for successful job performance within a given unit?	What activities and technologies should be trained for effective task performance in a unit?	How do work group goals and culture effect or constrain performance?
Organizational	How does the training tie with strategic planning?	What are the technologies of the organization?	How does the organization prefer the training?

Table 1: The Content-Levels Framework

McGehee and Thayer's original set of questions on content emerges in the cells along the shaded diagonal; the off-diagonal cells reflect issues that a more systematic and systemic approach to needs assessment should also consider. This approach essentially expands each of the content areas vertically across the various levels of an organizational system. For example, within the person column, a complete needs assessment would focus on the knowledge, skills, abilities, motivations, and attitudes of individuals, of each formal work group operating in the organization (without consideration of individual workers), as well as of the entire organizational system (without reference to specific individuals or groups). In addition, within the task column, assessment of

individual needs would entail an analysis of the job to be performed by a trainee upon completion of a training program (e.g., a job description), while assessment at the more “macro” levels would focus on the activities, technologies, and behaviours necessary for effective performance within a given subunit (e.g., division, department, or project) as well as organization-wide.

The questions in each cell illustrated in Table 1 are indicative and provide the basic direction and guidance for those who conduct the needs assessment. These questions/directions provided great help in order to formulate the first set of questions that are used during the data collection techniques preparation.

Assessment tools and techniques

Assessment techniques mainly refer to data gathering methods for conducting needs assessment. Data gathering is the cornerstone of any needs assessment project. The fundamental premise of needs assessment is that in order to make effective decisions about training needs data must first be gathered.

There are many ways to collect data. The most commonly used methods are interviews, focus groups, surveys and questionnaires, and observations. Due to resource and time limitations and the fact that the user organizations are geographically dispersed, focus groups and observations were excluded as assessment techniques in the training needs assessment that we conducted.

As a main assessment technique, questionnaires were used. This assessment technique is appropriate when conducting job and task analysis and training needs assessment and when respondents are geographically dispersed [13].

2.2. The tOSSad Questionnaire

The tOSSad questionnaire [14] strictly follows the content-levels approach described in previous sections. In other words, it tries to identify training needs at individual, team and organizational levels and questions are designed to address training needs related to persons, tasks and organization. The individual aspect gathers the needs of the individual trainee. The organizational aspect gathers the training needs for a whole organization. The task oriented aspect, analyzes the trainee’s and the organization’s training needs on specific tasks.

The questionnaire has been designed to be simple and to avoid bias, and also considering the respondent’s frame of reference. Three different types of questions have been used; contingency questions (to facilitate branching through out the questionnaire), matrix questions (identical responses are assigned to different questions), and scaled questions (from a scale of 1 to 4). All questions are close ended to foster an efficient quantitative analysis of the questionnaire.

The target groups of the questionnaire are; software developers, administrators or support personnel, IT consultants, and end-users.

The questionnaire is divided into four distinct sections each serving a different evaluation. The first section gathers introductory information from the respondents as well as the current and future consideration of F/OSS in their IT strategy. The second section gathers how well the concepts and features of F/OSS are perceived by the respondent. The third section gathers the level of skill of the organization in F/OSS and software development in general. The

fourth section is the part where actual training needs are gathered. The questions in this section are either task oriented (where tasks are specifically named) or organization oriented (where job titles are specifically named). In task oriented questions we try to gather the following information for each specific task:

- Frequency of this task to occur among trainees
- The performance gap that occurs between the expected and actual results
- The level of skill needed for this task

The organizational questions gathers which job title (project manager, developer, consultant, etc) should be trained on what topic (development, requirements analysis, licensing, etc.)

The target groups for the tOSSad questionnaire was European SMEs with some interest in F/OSS. Over 50 SMEs filled the questionnaire from all the countries of tOSSad partners. The analysis was done at European level, not on a country basis. The questionnaire was designed and implemented as an online questionnaire.

The tOSSad questionnaire itself and the results obtained are given in the appendices of this report.

3. F/OSS in Vocational Curricula

This section of the report deals with curricula on F/OSS targeted to students who took their secondary high school degree, and need a one-year specialization curriculum. The curriculum, however, is not similar to a BSc university curriculum, because it is shorter (one year), and is aimed to give competences immediately useful for a technical career.

The curricula here presented are not developed in vacuum, but resembles two “IFTS” courses that have been actually designed, accepted, and are presently taught (as of end-2006) in two schools of Sardinia.

This presentation is of course a generalization of those curricula, but it is based on a real experience, and takes advantage of the strengths and weaknesses of that.

3.1. Needs that are satisfied and goals of the curricula

From surveys and interviews made to industries, public bodies and other organizations, it comes out that, together with the need of highly professional skills, there is a strong need of technicians able to make systems work, and configure and write simple software. These technicians are not requested to have design skills, but to know basic technology and to be able to follow instructions and to configure or to build software systems with simple architecture. They are usually young people, fairly motivated and with wages in the low-end range. They typically have a high school degree, taken from a technical high school in computer science or electronics, or had a good training in computer usage.

The need of this kind of technicians is particularly felt in the F/OSS field, because often technical high schools use proprietary operating systems and development environments to train their students.

The main goal of the proposed vocational curricula is to train a technician who:

- knows what F/OSS software is, how it is developed, and the basics of F/OSS software business models;
- knows the basics of open source licenses, and is able to ascertain if a given action is compatible with the license of a given OS application;
- is able to work in teams, interacting with other members and with the team manager;
- is able to solve non complex design problems, and to perform her job starting from design documents and guidelines;
- is able to install, start, configure and use the F/OSS software working environment subject of the curriculum (operating system, IDE, database, etc.);
- is able to follow technology advancement.

3.2. Prerequisites to enroll

The vocational curricula are designed for people with a high school degree with strong emphasis on computer science or electronics, or with proven professional experience in computer applications.

In deeper details:

Basic skills needed: fair general culture level; basic knowledge of English language (for non-English speaker countries); ability to use basic logical and mathematics reasoning.

Complementary skills: ability to communicate with others; ability to adapt to a changing environment; ability of self-evaluation.

Technical skills: basic PC operating system usage; basic knowledge of an office automation suite; basic knowledge of a programming language or database query language; basic Internet access knowledge.

3.3. Description of the professional figure

In the followings we will not describe only a single curriculum, but a set of curricula related to F/OSS programming, operating systems and/or applications. First, we will describe the characteristics common to all vocational curricula, and then we will introduce a set of possible curricula, detailing a couple of them.

All these curricula are related to F/OSS, and require to build knowledge about what is F/OSS, how it is developed by F/OSS communities, F/OSS business models, patent and rights management, basic GNU-Linux and other mainstream F/OSS software knowledge.

Each single curriculum, then, specializes in some specific fields, such as GNU-Linux administration, LAMP administration and programming, F/OSS databases, Java programming and F/OSS IDEs, etc. We call “F/OSS technician” the generic professional figure which results from following the proposed curricula.

In general, the F/OSS technician is a professional able to tackle the various issues related to simple design, development and administration of information systems based on technologies accessible with open source licenses. It is a professional figure able to autonomously manage the various operating aspects of firms – especially SMEs – and public bodies, which decide to use, or to migrate to F/OSS software.

This figure works as an employee or consultant of a firm or of a public body, and is able to effectively exchange information with teammates and management in a wide and diversified working context. This assumes the ability to relate with other persons, and with institutions, obtained through teamwork during training, and through learning problem solving techniques in a set of different fields.

In deeper detail, the F/OSS technician will obtain the following skills:

- ability to effectively work in teams;
- ability to describe in detail, and to solve problems;
- knowledge of what is an enterprise, and of basics of enterprise management;
- knowledge of duties and legal responsibilities of software professionals;
- knowledge of what copyright and patents are, and of usual open source licenses and licensing models, and capability of reasoning about their consequences;
- knowledge of F/OSS software communities and how they work, both developers’ and users’, and how to contact them looking for solutions to problems;
- knowledge of the various possible F/OSS business models, and of their economic consequences for firms and organizations which use or produce F/OSS software;
- capability of reasoning about the main features of F/OSS software, compared to proprietary software;
- basic knowledge of a GNU/Linux mainstream distribution, such as Red Hat, Debian, SUSE, Ubuntu: installation of the system, basic configuration, basic system commands and utilities;
- basic knowledge of mainstream F/OSS office automation applications, namely Mozilla Firefox and Thunderbird, and Open Office; these tools are used throughout the course to interact with the Web, to exchange mails, and to produce and deliver documents;
- basic knowledge of SQL, and of a mainstream F/OSS DBMS (MySQL or Postgresql): how to install it, how to make simple queries, how to configure the database.

As reported above, these skills are complemented by other, more specific skills, depending on the specific F/OSS curriculum.

3.4. Characteristics of F/OSS vocational curricula

To reach the formative goals described above, and to provide students with further specific capabilities, the F/OSS technician program is structured in three blocks. The total expected effort is of 60 ECTS credits (one full year of course). A part of these credits can be re-used in the case the student wishes to enroll in a standard BSc university course, after total or partial completion of the curriculum. The three blocks refer to the basic skills outlined above, to curriculum-specific technical skills, and to a practical stage to be followed in companies, public administrations, F/OSS software projects.

The three blocks can be summarized as follows:

- Core block:** it will cover the subjects needed to cover the knowledge and skills common to all curricula, and described above. The block will include subjects on basic computer architecture, statistics, legal issues, economic aspects and business models, sociological and ethical aspects, project management, English language (for non-English speaking countries), problem solving, which should provide the non-technical and mathematical background needed. In parallel, subjects on GNU-Linux management, F/OSS office automation tools and development environments and tools will provide exposure to more technical issues. After following these subjects, students are expected to have acquired knowledge and expertise about F/OSS software from the most important points of view, and be able of reasoning and taking decisions about it.
- Specialization block:** This block, which varies according to the specific curriculum followed, will include subjects aimed at providing the students with specialized knowledge and capabilities. The exact subjects will depend on the specific curriculum, and will be detailed in a following section.
- Practical work:** It will provide students with practical experience and hands-on involvement with real problems and issues related to F/OSS software. Depending on student's needs, and firms or public administrations availability, this block may consist in a stage in a firm or other kind of organization, performing work on a real project, or in supervised participation to a F/OSS software project.

3.5. Specific curricula

Besides general skills and knowledge, which all students must learn, the proposed F/OSS technician curriculum must specialize in specific fields, devoting the second block of the curriculum to this task.

Possible curricula cover the fields of expertise most relevant to and most needed by industry and public administration. A non-exhaustive list might be:

1. **F/OSS system specialist:** the F/OSS system specialist is a professional able to manage a small IT infrastructure, based on F/OSS software. She knows how to install, configure and administrate F/OSS software systems, and how to upgrade to new versions, to solve problems and ask for help. The target of this expertise is the organization's LAN, its servers and workstations, common Internet services such as the mail, a simple Web server, a firewall.
2. **LAMP specialist:** the LAMP specialist has specific skills on how to write and manage typical Web-based applications or portals, with the server being based on GNU-Linux and Apache systems, with application data residing on a F/OSS DBMS (usually Mysql and/or Postgresql), and with Perl (or Python, or PHP) as a scripting and programming language.
3. **F/OSS systems and Java specialist:** the F/OSS systems and Java specialist knows how to elicit and formalize system requirements in terms of system architecture and technologies, using F/OSS and open standards. She cooperates in planning and designing distributed and multi-platform system architectures, and knows how to develop and integrate F/OSS solutions with existing system and network management software.
4. **F/OSS database specialist:** the F/OSS database specialist knows in detail database programming and GNU-Linux DB server configuration. She knows how to correctly design, implement, tune and administer a relational database, with specific knowledge of Mysql and/or Postgresql F/OSS systems.
5. **Office automation specialist:** the office automation specialist has extensive knowledge of the main F/OSS productivity tools, such as Mozilla Firefox and Thunderbird, OpenOffice, the GIMP, CLAMAV anti-virus software, etc. She is able to effectively plan and support a migration from traditional, proprietary productivity applications to F/OSS ones.

As an example, we describe in detail in the Appendix 1, curriculum 1. The other curricula can be planned and assembled following the same principles.

4. F/OSS in the 3rd and 4th Level setting

In the area of third and fourth level education, the introduction of F/OSS curricula is less than straightforward but offers great potential. In our analysis, an approach that seeks to introduce a distinct F/OSS degree program, whether at 3rd or 4th level, is possible but this approach is not widely employed in mainstream educational settings. With some exceptions, such courses being delivered today that carry the F/OSS badge are in reality principally mainstream engineering and computer science courses with varying emphasis placed on F/OSS content within their syllabus.

There may be many reasons for the lack of distinct F/OSS courses. Inertia in the face of new educational opportunities is of course one, and we note that a number of European Universities form a vanguard willing to provide well focused F/OSS programs. However, a key issue is that we discern no general trend in the university sector to consider F/OSS study as a sensible standalone subject. A preliminary survey of academics conducted at one university (Trinity College Dublin) did not uncover a sentiment in favor of introducing F/OSS courses to complement existing degree programs either at undergraduate or postgraduate level (see the Appendix 3). This despite a generally favorable reported disposition towards F/OSS.

Thus, rather than seek to develop a definitive F/OSS curricula for 3rd and/or 4th level, as other work such as CALIBRE attempts, our approach to the promotion of F/OSS studies in third and fourth level is one based on assimilation of F/OSS material into existing curricula. To this end we develop a general model of postgraduate curriculum and methods by which F/OSS emphasis can be applied. We consider this to be the most pragmatic approach given the reality of the educational landscape. We also consider the balancing effect of introducing F/OSS study into mainstream degree programs to be a very worthwhile goal.

For clarity, we distinguish between three uses of F/OSS technology and methodologies in use in the graduate and postgraduate educational context. Firstly, we note the focus of number of courses being taught today at various institutions on F/OSS as the core subject matter of study. Such courses are given at both undergraduate and Masters level, and it is this usage of F/OSS that is relevant to our work. Secondly, there is the use of open source technology in the context of general computer engineering and computer science courses, and in specialist Masters courses whose focus is that other than F/OSS. Thirdly, there is an emerging trend amongst educationalists in the use of F/OSS methodologies to underpin a process for the development of curriculum materials, but where the curriculum subject matter is not F/OSS. Rather, the tools and techniques of F/OSS development are co-opted for educational purposes in much the same way as legal and other professions have begun to do.

An educational context where F/OSS is used for infrastructure and where F/OSS methodologies are used in the development of course material is of course likely to be conducive to the study of F/OSS. However, that F/OSS has substantial use in universities both in the provision of infrastructure and teaching environments is self evident, but not relevant to our discussion, save where this directly results in a focus on F/OSS in curriculum. Our analysis suggests that this proximity effect is small. Indeed it is perhaps a general characteristic of computer science degree programs that despite the prevalence of F/OSS software in infrastructure, very little of that technology appears in the subject matter delivered to students. Moreover, outside the context of constructionist learning models, the process by which educational material is developed is irrelevant to its delivery.

Thus, we hold that these uses of F/OSS are largely irrelevant to our present goals. In the following analysis, we seek to separate the study of F/OSS and methodologies from their use in order to arrive at a concise model for the development of F/OSS curriculum. That analysis suggests two complementary points:

- Courses bearing a F/OSS identity are, in curriculum terms, substantially

mainstream and quite typical engineering and science courses with relatively little distinguishable F/OSS content.

- The majority of courses not carrying an F/OSS identity could, if the providing institutions so chose, be enhanced to provide equivalent focus on F/OSS.

The development of a curriculum for the teaching of F/OSS should, in our view, also be informed by the educational contexts at which it is hoped it will be adopted. Thus, it is necessary not only to review existing courses given at certain institutions, but also to assess the attitudes and opportunities at other institutions where the study of F/OSS is to be encouraged.

4.1. The feasibility of F/OSS course provision

In seeking to develop F/OSS curricula, one must be aware of the goals and requirements of potential students, those who pay for their education, including governments and industry, and the educational institutions that provide courses. We focus here on the students and the universities and we note that in the context of F/OSS, the goals of these groups (students and academics) do not necessarily align. Despite the presence of F/OSS for some 30 years, and the substantial impact this software has had, we note very little F/OSS focus in the educational sector. In this section we explore the possible causes of this in order to identify remedial actions we might propose.

Our analysis suggests that though an education focused on F/OSS may be in the interests of sectors of the student population and industry, the teaching of F/OSS subjects is of marginal value to universities and their goals, except where such courses can be expected to boost student numbers in existing programs or provide self funding models for new courses. We believe this to be most likely at postgraduate level.

However, as the economic incentives in the third level sector shift decisively towards research excellence (and to the provision of postgraduate programs over undergraduate education), academic educational goals are increasingly defined in terms of research expertise. There is a commensurate drive towards a differentiation of course offerings, particularly at MSc level, that build on core research competencies of the institution's academic staff. For a subject to be adopted therefore, there must be substantial research interest in it that is capable of sustaining prolonged academic inquiry. If F/OSS subjects lack research interest, we may expect that the delivery of F/OSS focused material will be resisted, or at best considered of little importance.

Nevertheless, the generally positive view of F/OSS within the academic community, and the view commonly expressed therein that an understanding of F/OSS forms part of a balanced education, suggests opportunity for an increased role for F/OSS within existing university curricula.

If backing for formal F/OSS programs is not to come from academia then it may perhaps come from student demand. We now consider the likely motivations of undergraduate and postgraduate students in turn.

It is our anecdotal experience that our students are increasingly focused on the acquisition of skills with immediate application in the industrial sector. Indeed, in the recent past, degree courses have emerged that are narrowly focused on

aspects of a traditional computer science or engineering degree, such as programming, games development, networking and telecommunications, etc. that are proving attractive to these student goals. This offers the possibility that F/OSS focused programs at undergraduate level could provide competitive differentiation of a university's offerings. However, it is not evident on any available metric that the demand is substantial among the undergraduate population at present for formal F/OSS study.

There does appear to be potential at MSc level for student demand to encourage the provision of specialized courses identifiable as F/OSS focused. This is the natural home of overt specialization as the examination of any postgraduate degree program will attest. Many postgraduate students return to the university from the workplace, often for seeking specific training for their work or career. This retraining can be employer sponsored, or a personal decision. It is thus our view that F/OSS software is most likely to be considered worth dedicated further specialized study by graduates. In practical terms, there seems to be less impediments within universities for the provision of new postgraduate courses. Issues of certification carry less weight as courses tend to be promoted on the strengths of the host university and it's own research credentials. Moreover, unlike undergraduate education which is quite typically funded by state intervention at often below economic cost, postgraduate courses tend to adopt a self funding model. For these reasons, we anticipate that the provision of postgraduate courses with strong F/OSS focus is the most viable today, and thus focus on this sector in this report.

4.2. Quantifying Fourth Level F/OSS delivery

In seeking to develop a F/OSS curriculum for fourth level, it is useful to explore the range of relevant F/OSS courses provided today. The range of such courses span from the short duration technical usage courses up to MSc level. We place short duration technical training outside the scope of our present analysis and focus on higher level undergraduate and postgraduate courses purporting to have particular F/OSS credentials. In our analysis, the study of F/OSS at this level does not in practice constitute a sufficiently broad or deep subject matter to warrant dedicated, standalone programs. Please note that our analysis does not seek to assess the quality of these programs, which we assume to be high, but rather the proportion of the curriculum of such courses that contains readily identifiable F/OSS subject matter. Some universities do offer full Masters programs in F/OSS, such as the UK's Sheffield Hallam University [16] and Lincoln University [15] which offer a MSc in Open Source Systems and a MSc in Open Source Computing respectively. Others, such as the University of Victoria in Canada, offer modules which count as towards more general postgraduate studies in software engineering and computer science. When we analyze the proportion of these curricula that comprise of F/OSS focused material (as opposed to material that could be delivered with the aid of F/OSS technologies and methodologies as we have previously delineated) we find this to be a very small proportion. Of this, if one omits non-technical F/OSS studies that are perhaps properly considered as part of a broad curriculum programs, we find that perhaps one or two tenths of a typical F/OSS curriculum has primary focus on distinguishable F/OSS topics. Taking the example of the course provided by Lincoln University, a number of modules with specific F/OSS emphasis are offered, including "Open Source Business Models" and "HCI, Multimedia,

Graphics, and Human Factors in OS.” Whilst we readily acknowledge the study of business models relating to F/OSS to constitute clear F/OSS subject matter, we discern little justification for the particular study of F/OSS as it impacts fields such as human factors, multimedia and graphics. Nevertheless, allowing these and general subjects such as “Project Preparation and Professional Issues,” and introductory courses such as “Introduction to Open Source Software and OS Case Studies” as having strong F/OSS focus, we find that traditional computer science based modules, with a focus on distributed software engineering, make up more than half of the course. Furthermore, whilst it is normal and appropriate for subjects of varying technical complexity and substance to be found in any comprehensive course, the imbalance we find in this example does suggest that the substance of the course as a whole is less focused in practice on F/OSS than the curriculum might suggest. Similarly, at Sheffield Hallam University, though the proposed course is aimed at students with a background in computing or related disciplines who seek a “career as independent consultants in open source solutions”, the vast majority of its modules are again generic and technical, with less than half of those having a F/OSS emphasis, judging from the published curriculum.

That the badge of F/OSS is applied to these courses would appear to some degree to be a matter of seeking differentiation, perhaps to emphasize ethos, rather than indicating substantial difference in subject material. We submit that this can usefully and legitimately characterize the methodology and tools with which computing subjects are to be studied, but the study of courses that use this nomenclature does not greatly aid the identification of a viable curriculum for F/OSS.

In clear contrast with these declared “open source” courses, institutions such as the City University of London provide software engineering MSc courses with F/OSS studies delivered as modules. The university’s Information Systems and Information Systems & Technology course contains F/OSS electives. The University of Victoria, offers a single module rather than an entire Masters curriculum, which includes the study of F/OSS history, its characters and revolutionaries, methods, philosophies and other abstract aspects.

We seek to provide a F/OSS curriculum model that is broadly applicable across the educational sector. It would appear from these examples that the study of F/OSS does not in practice constitute a sufficiently broad or deep subject matter to warrant dedicated, standalone programs at postgraduate level at any but the few institutions that include within their faculty substantial and ongoing F/OSS activity and research focus. We judge this to be the minority. Moreover, it appears that F/OSS studies can either be included within the broad form of mainstream engineering and computer science curricula, or provided as minor modular additions to such courses. Nor do we discern an incremental approach from these examples, whereby over time a greater emphasis might be placed on F/OSS subjects at the expense of general engineering and computer science subjects: that the examples on which we focus are in some sense under development.

4.3. A F/OSS MSc Curriculum Approach

We advocate a F/OSS curriculum approach in which the study and use of F/OSS is considered in the context of mainstream Masters programs, as an adjunct to the

core discipline and to which the association of the term F/OSS with the evolved curriculum is meaningful. To this end, in the following sections we identify two distinct subject sets. First, we have developed an MSc subject set, being a taxonomy of disciplines included in a wide range of Masters degree programs. These subjects are not F/OSS subjects themselves, but opportunities exist for the introduction of F/OSS emphasis in both the material considered, and in the technology used in practical work associated with a course. Second, we identify a set of core F/OSS options that are F/OSS focused and that would form the core of any course supposing to be F/OSS oriented.

In this way we suggest a curriculum model in which an existing or new degree program might be enhanced by the inclusion of varying F/OSS study, as the needs of the program design dictate.

4.4. The MSc Subject Set

In this section, we outline a course subject set distilled from a general review of taught MSc courses. The subjects we describe in the remainder of this section are representative of the kinds of subjects presented to MSc students in a wide range of courses. We would not expect all of the subjects we discuss to be included in any particular course, and we would expect other subjects which we do not discuss to be present in particular MSc curricula. Our purpose is to recommend methods for introducing F/OSS technology and study into these subjects, thereby providing the basis, in conjunction with core F/OSS subjects, for the creation of well grounded MSc curricula with legitimate claim to F/OSS emphasis. We list our core MSc subject set below, and describe each in turn in subsequent subsections.

- Software Engineering
- Networking and Distributed Systems
- Administration and Information Security
- Programming and Design
- Ubiquitous and Pervasive Computing
- Artificial Intelligence
- High Performance Computing
- Human-Computer Interaction
- Vision and Graphics

Each institute surveyed treated each group of subjects differently; some of them offered specific MSc degrees in one of the areas while others ignored that aspect of Computer Science.

We discuss this treatment in the following sections.

Software Engineering

Review and Objectives

The purpose of a Software Engineering Course is to provide students with a thorough knowledge of software development as a technological and engineering

discipline. Such courses aim to develop skills in:

- Analyzing user requirements and designing appropriate software solutions.
- Designing and creating software to solve realistic problems.
- Evaluating and using modern software engineering environments, design methods and programming languages.
- Evaluating and solving interoperability issues.
- Working as an individual and as part of a team to develop and deliver quality software.

Themes of such courses include Software evolution and life-cycles; Domain analysis, requirements capture and refinement to specification; The design of small-, medium-, large- and enterprise-scale systems; Software Modeling; Large scale programming, its principles, practices and pitfalls; Advanced implementation techniques, including component and framework technologies; Software processes and Project management; Group working; Software Quality, verification and validation;

It is also typical for students to complete individual project work and to contribute to collaborative project work.

F/OSS Enhancement

In a software engineering course focused on F/OSS it is natural to assume a focus on development methodologies. However, we have suggested elsewhere that a core subject for a F/OSS should be its associated development methodologies. Thus while we would assume delivery of teaching on the topic of development paradigms will focus on F/OSS methods, we would expect that this would not be exclusively the focus in a general course of software engineering. However, even general teaching of design and architecture for example, can make use of the wide range of F/OSS software now available that support modeling and other activities.

www.tigris.org hosts a very large number of relevant projects delivering tools such as Argo UML and Subversion. www.sourceforge.net provides a repository unfocused in respect to software engineering, but nevertheless hosting projects such as Umbrello UML modeler, Nant, Draco, CruiseControl.

Relevant Project support environments include Eclipse (www.eclipse.org) and Gforge (www.gforge.org). Requirements and process modeling tools include DRES (www.sourceforge.net/projects/dres). Unit testing tools include Nunit (www.nunit.org) and Junit (junit.sourceforge.net) and the GNU compiler collection (gcc.gnu.org).

Networking and Distributed Systems

Review and Objectives

Building distributed applications is a difficult task due to the concurrency, communication latency, and possibility of partial failure that is inherent in distributed systems. As in other areas of computer science, the trend in providing support for building distributed applications has been towards presenting the

application developer with ever higher levels of abstraction and, in the particular case of distributed programming, of location transparency. Courses in Networking and Distributed Systems aim to develop skills in:

- Managing the complexities and tradeoffs involved in designing and building distributed applications.
- Evaluating and deploying key paradigms used in the area.
- Evaluating the potential application of research in the area.
- Integrated use of core technologies in solving real world problems.

Course Themes include Paradigms for distributed computing including message passing; remote procedure call; client-server computing; shared file systems; distributed objects; distributed transactions; component models; Enterprise Component Technology Stacks; process groups; Web technology including XML Web Services.

Students are typically expected to complete small individual concurrent and distributed programming projects, which often have a collaborative element centered on adherence to shared interactions protocols.

F/OSS Enhancement

All Network and Distributed Systems courses include a substantial practical element, for it is in the use of these technologies that understanding is built. There are F/OSS implementations of most relevant technology, obviously including Network protocol stacks in F/OSS operating systems such as Linux. A key technology focus is the distributed object technology, such as Java's RMI, CORBA, .Net Remoting, XML Web Services.

Open Source implementations exist for each of these:

Ubik RMI (<http://www.sapia-oss.org/projects/ubik/>);
Sun's RMI (<http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp>);
CORBA orb's such as TAO (<http://www.cs.wustl.edu/~schmidt/TAO.html>),
MICO (<http://www.math.uni-goettingen.de/micoe/>),
Jonathan (<http://www.objectweb.org/jonathan/jonathanHomePage.htm>),
JacORB (<http://www.inf.fu-berlin.de/~brose/jacorb/>),
and ORBacus (<http://www.ooc.com/ob/>);
Mono's .Net Implementation (www.mono.org);
Apache's Tomcat (www.apache.org);

Other useful technology includes: GNU Zebra (www.zebra.org, Free Routing Software for TCP/IP based routing protocols); BitTorrent (www.bittorrent.com, is a peer-assisted, digital content delivery platform); ISPConfig (<http://www.ispconfig.org/>, ISP control panel, manages httpd, ftp, bind, pop3, mysql, webalizer, mailquota, traffic limits, SSL, SSI etc...)

Information Management and Security

Review and Objectives

Information management studies inculcate skills in information mediation between users, resources and information systems in specific organizational contexts. Security is a key element of this. Information security requires a clear understanding of relevant technological issues and of the social/organizational issues, as well as the relationships between them. Courses in Information Management and Security aim to develop skills in:

- Supervision, design, development and management of an information system in all its dimensions: strategic, technological, human, financial and organizational.
- Integration of the geographic component into various professional applications of an information system.
- Practical use of information management technology.
- Technology and implementation approaches related to information security.

Course themes include Computer Networks; Architecture of information systems; Information Technology; Data Mining; Geomatics; Information Systems Management; Basic Techniques for Information Security; Information Security Standards; Security Incident Management and Response; Advanced Information Security Technologies;

F/OSS Enhancement

Information Management and Security studies will benefit from a case study approach, that focuses on the deployment of technology.

A number of open source database technologies are available, including MySQL (www.mysql.com), PostgreSQL (<http://www.postgresql.org>), MaxDB (<http://www.mysql.com/products/maxdb>), Firebird (<http://firebird.sourceforge.net>) and Ingres (<http://opensource.ca.com/projects/ingres>). In addition SAP (www.sap.com) have released software under GPL licensing.

Open Source Security Information Management (OSSIM – <http://www.ossim.net/>) provides a comprehensive compilation of tools which grant a network/security administrator with a detailed view of a system. OpenSSL (<http://www.openssl.org/>) provides a F/OSS implementation of the secure socket layer and transport layer security. Other tools such as OpenSSH (<http://www.openssh.com/>), a secure shell, provide the basis for case studies. Network monitoring applications include Nagios (<http://www.nagios.org/>). Network management case studies can also focus on open source tools such as SpamAssassin (<http://spamassassin.apache.org>) for addressing the issue of spam, and antivirus technology such as ClamAV (<http://www.clamav.net>). The apache foundation also deserves special mention for its web facing technology such as Apache, tomcat and so forth (<http://www.apache.net>).

Programming and Design

Review and Objectives

Most MSc courses include a module in which research, design and programming skills are focused on, by way of a substantial individual or occasionally group

project. Such work is often augmented with a lecture series that covers algorithmic concepts and practical techniques for solving advanced computational problems in the context of the MSc domain. The production of a thesis detailing the research and development is typically required. The project focused modules aim to develop skills in:

- Solving advanced computational problems
- Algorithm implementation.
- Software programming techniques.
- Research and design, and technical exposition.

Themes include Object Oriented Programming; Systems Architecture; Algorithms and problem solving; Data Structures and Algorithms; Event Driven and Internet Programming; Software Development Project; Database Systems; Modeling;

F/OSS Enhancement

While one can make no specific recommendations regarding F/OSS technology to be used in particular project work, there exists a wide range of open source development environments and programming infrastructure that make it possible for a strongly focused F/OSS development strategy to be adopted. Examples of tools include subversion for source control, the UML modeling tool ArgoUML, these and many other tools hosted at <http://www.tigris.org>. Eclipse is perhaps the most well known open source development environment (<http://www.eclipse.org>). Emacs, and its variant Xemacs, remain as some of the most powerful editing environments

(<http://www.gnu.org/software/emacs/emacs.html>)

and preferred by many experienced programmers. Many frameworks exist also, such as the Symfony Project (<http://www.symfony-project.com>) which provides advanced development support for commonly used programming languages.

Kal Toth provides a useful model for the use of F/OSS software in a group project context that is probably also applicable in individual programming projects, in his IEEE Software article “ Experiences with Open Source Software Engineering Tools”,

(http://www.computer.org/portal/cms_docs_software/software/homepage/2006/s6044.pdf)

Ubiquitous and Pervasive Computing

Review and Objectives

This subject provides a grounding in the integration of computing and electronic technologies, concepts and practices for the creation of pervasive (also known as ubiquitous) computer systems. Courses of this sort seek to build skills and understanding of the main requirements, techniques and solutions associated with Ubiquitous and Mobile Computing, including mobile devices and mobile application/middleware development, mobile services and associated protocols as well as research and development issues in the field.

Themes include Embedded Real-Time Operating Systems; Software development methods and tools; Mobile and Ubiquitous Computing Devices; HCI; Autonomous Systems; Data Communications and Wireless networking; Middleware; Computer Vision;

F/OSS Enhancement

A wide range of software systems can be used in the context of practical work in a ubiquitous and pervasive computing course.

Cantag is a F/OSS toolkit for designing and deploying marker-based vision systems (<http://ieeexplore.ieee.org/iel5/10682/33716/01604788.pdf>). Boingo wireless have released their embedded architecture toolkit as an open source product (<http://www.boingo.com>), and this kit is widely used commercially. IBM Pervasive Computing have contributed Speech Software to the Apache Software and Eclipse Foundations. Eclipse provides support for device and embedded development (<http://www.eclipse.org>). The Gcc compiler has been modified to work as a cross compiler for many micro controllers (<http://gcc.gnu.org>). A number of relevant projects include

- <http://www.opengroup.org/tech/management/arm/>
- <http://www.skyeye.org/index.shtml>
- <http://winavr.sourceforge.net/>
- <http://www.bsdhome.com/avrduke/>
- <http://www.avrfreaks.net/>
- <http://gridkit.sf.net>

Networking for pervasive computing is provided for by a number of projects:

- <http://www.isi.edu/nsnam/ns/>
- <http://moment.cs.ucsb.edu/AODV/aodv.html>
- <http://www.olsr.org/>
- <http://dast.nlanr.net/Projects/lperf/>

Artificial Intelligence

Review and Objectives

Research in AI spans knowledge representation and reasoning, the study of brain processes and artificial learning systems, computer vision, mobile and assembly robotics, music perception and visualization. AI courses are interdisciplinary in nature, and the focus of a number of quite specialized courses. Students of these courses, and at courses more general in nature, develop skills in:

- understanding the range of intelligent systems techniques currently available.
- selecting an appropriate intelligent systems technique to suit a given task.
- assessing the relative merits of competing approaches in a given scenario.
- developing new intelligent systems techniques.

Themes include subjects such as Supervised Learning; Information Theory; Evolutionary Systems; Programming Matlab; Unsupervised Learning; Pattern Recognition and Machine Vision; Machine Learning; Bioinformatics; Intelligent

Systems in Business; Intelligent Robotics; Natural Language and Language Engineering; Neural Computation and NeuroInformatics; Theoretical Computer Science.

F/OSS Enhancement

AI is well served by the F/OSS community.

The OpenAI (<http://openai.sourceforge.net>) is developing a range of classic development tools for system classes including including Neural Networks, Genetic Algorithms, Finite State Machines, Mobile Agent Systems. A large collection of AI software packages is available from the CMU Artificial Intelligence Repository (<http://www.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/areas/0.html>). IBM provides a software package for creating Internet agents using Java applets, called Aglets (<http://www.trl.ibm.com/aglets/>).

A wide range of computing languages of interest to AI teaching includes: XSB prolog (<http://xsb.sourceforge.net/>); the Java Embedded Object Production System (<http://www.di.ufpe.br/~csff/jeops/>);

POPLOG is a free, open source, multi-language software development environment providing incremental compilers for a number of interactive programming (<http://www.cs.bham.ac.uk/research/projects/poplog/poplog.info.html>);

A comprehensive survey of AI software, including Knowledge representation systems, natural language processing, speech, neural networks, robotics and so forth, can be found at <http://www.faqs.org/faqs/ai-faq/general/part6/preamble.html>.

High Performance Computing

Review and Objectives

High-Performance Computing is the use of the most powerful computer systems to solve the substantial technical and numerical problems that arise in simulations of complex physical, biological and financial systems. Courses in this area seek to equip students with the intellectual tools, knowledge and practical skills they will need to become developers and exploiters of the parallel and distributed systems. Courses in High Performance Computing aim to develop skills in:

- the use of the major techniques for programming parallel supercomputers and their application to real world HPC applications.
- the use of variety of HPC programming languages.
- mathematical techniques for complex systems.

Themes include HPC Software Programming Paradigms; Computer Architecture; Simulation; Computational Grid; Scientific Visualization; Hardware, Compilers and Performance Optimization.

F/OSS Enhancement

The high performance computing world is well provided for in terms of F/OSS and we provide a brief introduction here. The GNU tools (<http://www.gnu.org>) are used extensively as default compilation tools. Others tools such as OCTAVE provide matlab-like numerical analysis (<http://www.octave.org>). Cluster computing infrastructure is provided by software such as: OpenMOSIX – a F/OSS Linux Cluster system - <http://openmosix.sourceforge.net/>; OSCAR – a F/OSS cluster application resources – <http://oscar.openclustergroup.org>; Rocks – <http://www.rocksclusters.org>; DCC – <http://dcc.irb.hr>. OpenMPI provide a message passing library for high performance computing (<http://www.open-mpi.org>).

Cactus is a F/OSS problem solving environment designed for scientists and engineers (<http://www.cactuscode.org>). The Globus project supports the construction of grids for high performance computation (<http://www.globus.org>). Condor is a software system developed at the University of Wisconsin that provides computing capabilities through efficient capture of cycles on idle machines (High Throughput Computing) (<http://www.cs.wisc.edu/condor/>).

Besides technology, there are very active F/OSS focused research communities that make it relatively easy to focus a high performance computing course on open source content. The site <http://www.openclustergroup.org> provides contacts to a F/OSS cluster computing community.

Vision and Graphics

Review and Objectives

Computer graphics generates realistic images of scene from models of the scene stored in the computer. Machine vision is the opposite process to computer graphics: it constructs a model of the scene from camera images. Courses in Vision and Graphics aim to develop skills in:

- mathematical and algorithmic principles underlying computer graphics and vision
- theoretical and practical understanding of techniques and mechanisms
- the use of key software libraries and systems
- evaluation of new techniques

Themes include mathematical methods, algorithms and implementations; image processing; graphical modeling and visualization; mathematical programming; Pattern recognition; virtual environments; rendering and animation; medical scientific computing.

F/OSS Enhancement

A range of F/OSS systems exist that are suitable for use in Vision and Graphics focused courses.

An example is The *Open Source Computer Vision Library*, which can be downloaded from <http://www.intel.com/technology/computing/opencv/>. A community of users of this software, numbering over 3,000 can be found at OpenCV-Subscribe@yahoogroups.com. This library is aimed mainly at real time computer vision applications, including Object Identification, Segmentation and Recognition, Face Recognition, Gesture Recognition, Motion Tracking and Mobile Robotics.

An extensive collection of open source code is to be found at <http://www.cs.cmu.edu/~cil/v-source.html>. This page includes research code, Image Processing Toolkits, and Display tools, Synthetic Data Generators and Mathematical toolkits.

Vision Egg (<http://www.visionegg.org>) is a powerful, flexible tool for producing stimuli for vision research experiments, using inexpensive PCs and standard consumer graphics cards. TINA (<http://www.niac.man.ac.uk/Tina>) provides C open source libraries for 3D robotics and medical image analysis. VXL (<http://vxl.sourceforge.net/>) is a portable F/OSS C++ library for computer vision. Insight Segmentation and Registration Toolkit (<http://www.itk.org/>) provides open source C++ libraries for medical oriented imaging.

4.5. Core F/OSS Options

The following core options cover distinctly F/OSS subjects and we would expect these to be included to a greater or lesser degree in educational programs, depending of course on the audience, number of years of the course, background of the participants, and last but not less important the main objectives of the course.

4.5.1. Licensing and Legal Issues

Licensing is a crucial issue in F/OSS if it is to gain traction in mainstream commercial contexts. The range of Open Source licenses is varied and with many difference in how the original concept of “freedom” is approached and regulated. The GPL [31] approach is called protective or copyleft. One of the aims is to maintain the code it protects under the same open license terms. Some relevant licenses with the same style are LGPL [32], Mozilla Public License [34], Sun Public License [33] and Open Software License [20]. A relevant, and different approach, called non-copyleft, is used by the BSD License [30], that allows open source code to be included in a closed source release, typical of commercial licensing. Some relevant licenses with this approach are FreeBSD [29], NetBSD [27] , OpenBSD [26], Academic Free License [25], Apache Software License [24] , W3C Software Notice and License [17]. Is important to explore the difference between code freedom and developers freedom, to let developers and companies be aware of the proper license to use before to release any software.

Another issue is the compatibility between licenses, and a proper evaluation of license issue is not a simple task. Some tools exists [21] to help in this process, but even if they can be used as a first analysis, they cannot cover the full procedure to decide what is the proper licensing schema to use. It's obvious that F/OSS does not exclude business, but using F/OSS to create a Commercial solution involves the proper consideration of some important aspects. Care is needed in the choice of the license to use and the knowledge of the license that covers F/OSS eventually utilized. BSD style licensing allows change in any software covered and subsequent publication on closed commercial terms. On the other hand, the use of GPL prevents this kind of change, forcing the propagation of licensing terms to derivative work. LGPL relaxes this strict propagation property of the GPL license.

Is also important to outline how the legal aspects of F/OSS licensing is not well defined and some restriction, due to a license, may become an issue depending on the intention of the subject that owns the copyright on the software that has

been used or modified. GPL derivative work are required to be GPL'd, but the definition is not clear. A noticeable example is the addition of modules to the Linux Kernel. They may be considered derivative work, and some companies tried to solve the problem loading them after the Kernel is loaded, as external modules. The FSF would consider them derivative, and in that case they must be covered by GPL, but the owner of the Linux Kernel Copyright, Linus Torvald leaves a bit more freedom. He considers derivative any module, despite when is loaded, but not if it is just a mere rewrite of a module, originally written for another operating system. This makes the situation not only under the variability of interpretation of the license itself, but also on how the owner of the copyright intends to use the license to protect the software.

Legislation and licensing

An important role in the issue of evaluating a license is the interaction between the license itself with different legislations and the interaction between legislation and legislation. As a representative example is the use of the word "derivative" in GPL license. This word comes from the US legislation definition, but can be interpreted in different ways, allowing a wide range of uncertainty in how in a legal act this term would apply. The discussion is really open and is enough to consider the difference in executed software on one side and interpreted software on the other, that introduces other levels of difficulties in applying license rules.

Consideration has to be made of commercial solutions based on a F/OSS, both in case of a service or a further development of an original Open Source solution. Care must be put in covering responsibilities about the software not developed by the company itself, but coming from the F/OSS solution. Open Source licenses usually don't cover accept any responsibility for malfunctioning software including losing data, the responsibility relies on who use the F/OSS. To protect the company from legal act a detailed contract has to be prepared, protecting from unknown bugs from the original F/OSS. Examples include commercial solutions based on a Linux distribution or the Linux Kernel itself, that don't directly cover the malfunctioning due to unknown kernel bugs, but include the configuration and maintenance to obtain the patch or updates as soon as they are available.

Working on this kind of contract it's a complex task and usually must be prepared by legal experts in consultation with more technical persons. Another factor introducing complexity is the interaction between different legislation in different countries and the level of understanding each legislative body has of Open Source licensing. A proper planning of a license scheme has to deal with the possibility of change in the publication requirements of the release: there are examples of closed solution that moved to an Open Source license and successful F/OSS solution that reemerged partially closed to create commercial services and software solution based on them. Many scenarios are possible. An example is the change of licensing strategy because of the size of the software project or commercial strategy. A flexible and modular project may allow a licensing scheme able to be adapted to a growing project, to enable business solution to be added as plug-in in a common open core. On the other side, an inflexible software project based on a big, atomic core, may find it difficult to be used as a base for a commercial solution, specially if it is covered with more protective F/OSS licenses.

As outlined above, different legislations cover and understand licenses in different ways, so the evolution of EU, national legislations and international agreement can have a crucial role in the evolution of licenses and how business solution create, adopt or support Open Source. The situation is evolving and is important to have the capability to evaluate both the pure legal and licensing side, but in relation to the business model and the development methodologies in use.

Information privacy and security

Information privacy and security is another issue to evaluate, in principle is an issue for any software with any license, but the current legislations care about protecting sensible and personal data, in the aim to obtain privacy and the possibility to notify the subject in case of any thief of information. The development life cycle and licensing schema of an F/OSS or a commercial, not open solution based on an original F/OSS solution, may encounter limitation or misinterpretation about who has the responsibility in case of weak protection of sensible data, thief and related issues. A good understanding of the legal consequences has to be evaluated when selecting the media and the contract when an Open Source is provided, or on how to react in case of unknown bugs in case of a commercial solution that is in some way based on an original F/OSS.

Intellectual Property, Copyrights and Patents

As with privacy and security issues, the issue of intellectual property must be considered independently of open or closed status, and the licensing scheme. Again F/OSS and the controversy of some Copyright Acts, and agreements, makes the F/OSS prone to misinterpretation and legal issues. An example is the Digital Millenium Copyright Act (DMCA), that restricts the creation, distribution and use of technologies that can circumvent systems intended to protect copyrighted material. Unfortunately, from the legal point of view, a F/OSS not originally developed for those purposes can be used or easily changed to do so. The controversy of interpretation of the rule has to be considered and evaluated when creating contracts and licensing scheme [22]. A sector to explore to have a real scenario example is the music distribution and the use of peer-to-peer (p2p) architectures.

Speaking about intellectual property of the software itself, it is worth mentioning the controversy surrounding the case of SCO against Linux. SCO claimed that some code with proprietary legal status was included in the Linux Kernel, and was asking to have a reimbursement from the Linux developers and users [22]. Independent of whether this claim is successful or not, the uncertainty such claims can raise a strong disincentive to the use of F/OSS. A dispassionate understanding of the issues are no longer an option for F/OSS practitioners.

There is an important further issue regarding the use of software patenting, a divisive issue at present. This is a crucial issue for the F/OSS practitioner and user and the debate is complex – more complex than either camp would generally admit. Issues such as protection of invention, the protection of investment, the nature of innovation and the correct characterization of invention in software, the best legal methods to encourage software innovation, and other quite philosophical questions about the nature of software are at the center of this debate. There is much muddled thinking on both sides.

As a final statement it is important for developers and professional developers of F/OSS solution to be aware of the outlined issues, being them involved in producing themselves intellectual property, tools to manage copyrighted content and eventually under the coverage of software patents. On the other side is the professional developer, that has now a new role, to provide consultancy and the knowledge to properly evaluate the interactions between the software, licensing, copyrights and the involved legislations.

4.5.2. Development Methodologies

In purist terms, F/OSS refers merely to a form of licensing and access to source code, but the licensing form and code distribution allows software development to take on a very different decentralized aspect, and it is this bazaar style of development that has become the bedrock of F/OSS development. The typical F/OSS developer's day to day working life is quite different compared to those operating in a classic commercial solution. F/OSS projects involve people distributed geographically, with a loosely coordinated activity, while a commercial environment is more likely to have a centralized control, with more pressure to obtain prefixed results due to company commitments. Some F/OSS projects started as a small community, if not a single author, with no initial interaction with other developers, and only when the software reaches a reasonable level of maturity and visibility does areal community of developers involved at different levels emerge. Some of the developers maybe full-time, paid by companies with interest in developing any part of the project or extensions, but most are usually part-time with unpredictable response time, because they are working for their personal interest in the project, for self-learning and motivated by the philosophical approach to the Open Source.

Both the cathedral style or the bazaar style are used in Open Source projects. Those shaped by the first usually begin with a single individual or a small community, that creates a main core of developers with a small interaction with other developers via suggestions and contributions. The second style is based on a open environment, where a dynamic interaction allow to continuously collect comments and contributions. Usually it is based on tools, like mailing lists, to manage open discussions. This approach allows more contributions, but introduces the potential drawback of having less control.

In any case, an issue of F/OSS is the creation of new projects as branches of the original project, known as “forking” presumably after the Unix fork command which duplicates a process. An open community may help in hosting all the issues and covering more requests, so going in the direction to avoid the need to create another project. On the other side a project that becomes too wide may be more difficult to control and further develop. The creation of branch project affords the possibility to go in a direction not needed by the original group controlling the main development, but sometimes is a waste of resources, and simply splits a community in groups instead of maintaining a common effort. The management of the project, and the architecture and organization of the code is important and F/OSS shows the capability to have a modular structure, while commercial solutions usually show a relation with the structure of the company itself that can introduce limitations. As an example modularization is not only a concept of F/OSS, but a general software engineering concept, but within F/OSS is a great solution to overtake license limitation, project’s branches pollution, and to enable hybrid commercial-open solutions.

A different strategy in F/OSS development methodologies is also on how to manage new releases and backward compatibility. An interest of the community itself is to maintain users, that increase the group of potential contributors. Not having a cost of licensing and distribution for new releases, the project need to deal with the issue of the maintenance effort in upgrading or having new releases working with old data and/or environments. In different important projects different teams manage different main releases, an example is the Linux kernel, with development versions, collecting proposal and main improvements (and giving the possibility to test the functionalities under development by users), and stable releases, representing the last step for production versions.

The Apache project

As a valuable example we give some details about the Apache project, born in 1995 to coordinate the effort in improving the NCSA httpd program. A planning was done on how to coordinate distributed people contributing to the project on a volunteer part-time basis. A set of tools to help in the coordination of people distributed on the territory has been used to maintain the code and to make decisions. The developers is made by a core group, made by not more than 15-25 persons. A survey [19] shows that about 80% of the changes in the code is done by core developers, but at the same time problem reports and defect repairs are more distributed in the community. More than 400 are the people that contributed to the code while more than 3000 contributed sending a problem report. Currently the Apache Server is used for 60% of the Web Sites with about double of the quote than the first commercial solution.

4.5.3. Business Models

Business with Open Source is currently done using different approaches, communities that became a company, companies that moved proprietary solution to F/OSS. A few examples shows that is likely to happen when a company is lagging behind the leader in the sector or when the company is nearly to disappear (Netscape is an example, releasing the Mozilla engine, that has been boosted opening the source code). However, certain big firms adopted F/OSS solutions pro-actively, or create services relative to F/OSS. An example is IBM, working with Linux and Apache. A business created around a F/OSS consists of providing commercial complementary services and products not provided with F/OSS. Two example of companies doing that are Red Hat and VA Linux. F/OSS shows the capability to challenge proprietary solutions and to have an important role in the Business environment. The original approach itself, of having a free circulation of knowledge in the form of F/OSS is not against the business itself, but was indeed confusing making the adoption of F/OSS much slower in business environment. Is now understood by companies that F/OSS can be, on the opposite, a great resource. Some interesting studies explain why F/OSS is not anti-capitalist but indeed is an evolutionary step in the so called late capitalism [18].

Services or Software

Before 2001 the software sector could legitimately claim to be industrial, manufacturers of products. However, since 2001 it is becoming more accurate to see an increasing proportion of the software sector as a services industry.

Customers may be unwilling to invest in software or upgrades during recession phases stage, while services oriented companies are more protected having long term contract with customers. This may be interpreted as a positive news for F/OSS that enables the creation of commercial services (e-services) whose profits derive from the use of even free software. The interest of big software companies such as IBM naturally derives from this.

We consider Linux an important example in the F/OSS panorama, to speak about independent software vendors and value added resellers. The first approach, sees companies releasing solutions based on Linux or F/OSS libraries to build commercial solutions. On the opposite, value added resellers add a value providing ready solutions based on F/OSS, like some companies do with Linux distributions. An interesting example on how the flexibility of F/OSS effects markets, is again Linux for the embedded systems market, where Linux enabled a bigger group of companies to create advanced solutions.

The use of F/OSS promoted and pushed the business connected with the need of qualified consultants and consultancy services. For example in system administration including F/OSS set-up and maintenance. Connected to those aspects there is the need of training, due to the request of qualified professional and also training for final users. The F/OSS philosophical approach touches this issue, allowing to have Open documentation, sometimes very complete and detailed. Examples come from the know main F/OSS projects and to related material, covering some aspects of the IT, not describing a single F/OSS solution, but collecting F/OSS oriented knowledge, [23] is a valuable example for the Networking field.

Has also to be mentioned the importance given by the opportunity of introducing and using F/OSS in developing countries. This is not only important in trying to reduce the digital divide, but also a way to apply and create business models in completely new markets, with the possibility to fully enjoy the benefits of F/OSS.

5. Professional learning / professional certification

The adoption of F/OSS depends on some external factors which are not clearly seen the first time. One of these factors is professional education, or training, which has a close relationship with Linux training at schools. Currently Linux holds a considerable market share on general purpose server systems, mission critical servers and enterprise software. IT managers look for opportunity costs while deploying Linux servers, and discounted costs coming from using F/OSS based systems will be a definite plus for all kinds of organizations. In order for Linux to benefit from a rich market, such deployments should be supported with knowledgeable people.

However, not every time the skill to deploy, maintain and support Linux and F/OSS based systems is gained at school. While many universities give courses about F/OSS development in BS and MS level, most do not offer a competitive curriculum which will guide the new graduates survive in an open source world. This situation eventually poses a great importance on professional Linux training in which Linux training, service and distribution companies have been working on for years in order to provide their own curriculum.

Professional training can be in forms like courseware, Internet based, instructor-

led, video, books and training centers. Some of these methods should be supported by certificates for proper measurement. Moreover, customers can choose among three types professional training companies with some minor changes in offerings.

1. **Distribution specific professional training:** This kind of program is mainly driven by distribution companies such as Novell/SUSE, Red Hat and Mandriva. Clients generally take a 3 to 5-day training and a hands-on and theoretical certification examination. Certificate is granted to those who has a successful score. However, the success rate is roughly more than 50% due to harsh passing limits, sometimes well over 80%.
2. **Community driven professional training:** There are not much examples for this option, however Linux Professional Institute (LPI) is by far the most known training organization. In this model, the curriculum is not tied to one choice of Linux distribution, and hence, is known to be independent from Linux distribution companies. In particular, LPI is supported by IBM, NEC, SGI and also Linux distribution companies like Turbolinux and Novell.
3. **Professional training via Linux support companies:** Such trainings are given by companies with a strong Linux background and usually given in the company building. They are usually not bound to one Linux distribution and/or Linux distribution company. Usually the certifications are titled as "certificate of success" or "participation certificate". Such training experiences are usually bound to one country, i.e the same training is not offered in another country.

The table below shows various aspects of the trainings.

	(1)	(2)	(3)
Cost	Usually higher than (2) and (3). Separate pricing for training and certification is possible	Fixed cost rate	Depends on the company, sometimes given for free for market awareness.
Duration	Training usually lasts for 3 days to 5 days	Same as (1)	Generally tailored to suit customer's needs
Coverage	Covers only the distribution company products	Can cover any F/OSS based software	Same as (1), also additional trainings of proprietary software for migration is possible
Method	Via training partners	Same as (1)	Self allocation in company and/or customer training/learning rooms

Legend:

- 1) Distribution specific professional training
- 2) Community driven professional training

3) Professional training via Linux support companies

IT professionals who want to test their skills can choose to have a distribution-neutral or generic training and certification. However, this approach can be problematic when it comes to proving the skills in a professional environment and may cause unwanted results, since distributions are mostly getting apart from each other by means of administration, package management, methodologies for system handling.

6. Training methods

In this section the state of art of training methods are presented. We begin from open source learning. New possibilities of learning introduced by information technology such as adaptive e-learning environment based on F/OSS and e-learning standards and learning design is introduced in second part of section.

The need for joint efforts by the EU in the E-learning area is undisputed. This is recognized in EU eEurope Action Plans of 2002 and 2005, the E-learning initiative and programme as well as the i2010 initiative. More broadly, it is implicit in the objective set in Lisbon in 2000 by the EU Heads of State and Government of the European Union becoming the most competitive and dynamic knowledge economy in the world by 2010.

6.1. Open Source learning

At universities visions of learning communities, open development and exchange of ideas and useful services is one direction of institutional culture growth. This is comparable with open source software movement. Since higher education and the open source software movement share common values, it is possible that higher education might use an open source model for creation their curriculum. The main idea is that in higher education and elsewhere the advantages to be gained through the open development and exchange of ideas. The open source development can be divided to the two categories:

- open source courseware development,
- open source knowledgware development – the tools.

MIT's partnership with Stanford on the Open Knowledge Initiative (<http://web.mit.edu/oki>) is an example of a project designed to develop a learning management system – web based tools for storing, retrieving, and disseminating educational resources and activities.

Projects such as MIT's OpenCourseWare effort (<http://web.mit.edu/ocw/>), which aims to make instructional materials available free on the web, and MERLOT project (<http://www.merlot.org/Home.po>), which endeavors to place on the web knowledge objects that have been evaluated for quality, represents variations on an open source courseware development process.

6.1.1. Open CourseWare

MIT OpenCourseWare (MIT OCW) makes the course materials that are used in the teaching of almost all MIT's undergraduate and graduate subjects available on

the Web, free of charge, to any user anywhere in the world. MIT OCW is a large-scale, Web-based publication of MIT course materials, and is not a degree-granting or credit-bearing initiative. MIT OCW would not be possible without the support and generosity of the MIT faculty who choose to share their research, pedagogy, and knowledge for the benefit of others.

The MIT OCW project aligns closely with MIT's institutional mission (*to advance knowledge and education and serve the world*) and is true to MIT's values of excellence, innovation, and leadership.

MIT faculty share a passion for teaching and contributing to their field of study. The MIT OCW staff strives to support the MIT faculty by:

- Helping faculty put their course materials online for teaching
- Providing a vehicle for faculty to share their ideas and contribute to their discipline

With 1100 courses now available, we expect MIT OCW to reach a steady — though never static — state by 2007. Between now and then, we will publish the materials from virtually all of MIT's undergraduate and graduate courses.

6.1.2. Open Knowledge Initiative

Phase 1

The Open Knowledge Initiative started at MIT in 2001 with funding from the Andrew W. Mellon Foundation. O.K.I. completed its first phase in July 2003 with delivery of the following:

- A set of specifications that define interactions between educational software modules, programs, and systems within and across institutions.
- Open source code for a reference implementation of each specification, with documentation of the architectural assumptions that underlie the specifications.
- Open source code demonstrating the application of OSIDs to achieve service level interoperability.

Phase 2

Having provided the architectural framework for a new educational infrastructure, O.K.I. seeks to create and sustain a community where both open source and commercially licensed products can evolve. To achieve this, O.K.I. has the following goals:

- Maintain and refine the core specifications
- Research and develop new educational interoperability specifications
- Build a global developer community by providing developer training and access to information

Advance customer adoption of O.K.I. educational software by helping educational institutions and vendors learn and implement the architecture

6.2. New possibilities of learning introduced by information technologies

Distance education, computer based learning environments and e-learning are today actual topics. At the same time there's been some confusion on usage and

understanding of this terms. In this part of report we make some clarification on this issue. E-learning is not only Internet learning materials, but Internet can be also a cooperative learning media. These possibilities are discussed also in this section.

6.2.1. Distance learning and e-learning

Concept clarifications

Distance education, computer based learning environments and E-learning

There are many terms for new computer and Internet based education. Some of them are: virtual education, online education, Internet-based education, web-based education, and education via computer mediated communication.

Distance education is a form of education characterized by:

- the quasi-permanent separation of teacher and learner throughout the length of the learning process (this distinguishes it from conventional face-to-face education);
- the influence of an educational organization both in the planning and preparation of learning materials and in the provision of student support services (this distinguishes it from private study and teach yourself programmes);
- the use of technical media – print, audio, video or computer – to unite teacher and learner and carry the content of the course; the provision of two-way communication so that the student may benefit from or even initiate dialogue (this distinguishes it from other uses of technology in education); and
- the quasi-permanent absence of the learning group throughout the length of the learning process so that people are usually taught as individuals rather than in groups, with the possibility of occasional meetings, either face-to-face or by electronic means, for both didactic and socialization purposes.

If we accept that online education represents a subset of distance education we may define online education by accepting distance education definition and changing the third and fourth points to the following:

- the use of *computers and computer networks* to unite teacher and learners and carry the content of the course;
- the provision of two-way communication *via computer networks* so that the student may benefit from or even initiate dialogue (this distinguishes it from other uses of technology in education).

The term '*e-learning*' (and also '*m-learning*') have entered the scene more recently.

Some writers in the distance education field have been skeptical to the value of the e-learning concept. Most examples of e-learning programs seem to be extremely costly to develop and most often cover low-level knowledge and facts based on a simplistic view of what learning is. This view seems to suppose that

e-learning is defined to include interactive learning in which the learning content is available online and provides automatic feedback to the student's learning activities only. However, it seems that most definitions of e-learning now include the availability of online communication with real people (co-learners and/or tutor(s), and thus focus both on the learning content and on communication and interactions with people.

The *E-learning project example* of the European Vocational Training Association (EVTA 2005) presents the following typologies of e-learning or different ways of exploiting the Internet for learning purposes:

"A. free use of the network for accessing unstructured aids following a specific training pathway, just as if one had access to a huge library to leaf through books on a specific subject;

B. use of teaching aids following a specific training, worked out to be exploited in distance and individual learning;

C. use of teaching aids to be exploited in distance, mainly individualized learning with help from the coaching source;

D. use of teaching aids, not necessarily structured within a real individual training course, with the help from a tutor or trainers made available by the coaching source;

E. use of blended approach (in-class or distance) based on the complementary character between in-class and distance learning;

F. use of real network training approaches based on the very interaction of all the learning parties (participants, tutors, experts);

G. use of the practice communities aimed at forming co-operative groups, made up of former trainees or professionals, for instance, sharing experiences, knowledge and practices of excellence with a view to the collective growth of the whole group."

E-learning is often used as a more generic term and as a synonym for online education and more and more as a synonym for distance education.

At *eEurope – Europe's Information Society Thematic Portal* e-learning is defined as:

"The use of new multimedia technologies and the Internet to improve the quality of learning by facilitating access to resources and services as well as remote exchanges and collaboration." This definition was originally launched in *The eLearning Action Plan* (European Commission 2001).

The term e-learning is, as one can see, not very precise, and it should be pointed out that learning is just one element of education. One may also claim that e-learning companies often focus on course content, while online education institutions and other institutions offering distance education/online education cover the whole range of educational services of which student support most often is given major emphasis. One should note that the *Open and distance Quality Council* in UK defines e-learning as:

"E-Learning is the effective learning process created by combining digitally delivered content with (learning) support and services."

This definition claims e-learning to include student support and student services and thus excludes self-instructional programs, pure computer-based learning from the e-learning definition.

6.2.2. Cooperative learning through the Internet

The traditional "lecture" model of science, mathematics, and technology curricula emphasized the transmission of factual data and skills from a teacher to a student. It is based on the concept of scientific knowledge as a stockable and transferable entity. Today most distance-learning courses are no more than sophisticated correspondence courses, in which lone students in front of computers take in a certain amount of information transmitted via the Internet.

In contrast, a number of studies about how professionals acquire knowledge indicate that learning is inextricably intertwined with multi-directional activities such as work and play, and that learning is essentially a social activity. This position on learning argues that the process of acquiring knowledge cannot be separated from the process of applying it, because knowledge is temporary, developmental, and socially and culturally mediated.

In contrast to these models, the Collaborative Learning and Teaching (COLT) model developed jointly by the College of William and Mary in Virginia and Keio University in Japan moves one step further, to engage students because of the following factors:

- The COLT model requires that students conduct their own research for knowledge creation at local sites;
- It connects students with different cultural backgrounds for direct cross-cultural interface;
- The collaboration for co-knowing is international;
- The students employ a much wider range of communication tools (i.e. International video conferences, chat, e-mail, trans-Pacific file exchange, Web-page creation, bulletin boards, PowerPoint presentations, and film-archive retrieval);
- The final project is not only edited collaboratively, but also presented collaboratively. The students, as well as the course instructors, participate in cross-national collaboration in order to successfully implement the course; and
- It intentionally creates a learning environment where participants need to manage uncertainty and uncertain knowledge.

The COLT model allows collaborative groups to execute tasks that are too complex for one individual to undertake. It provides opportunities for students to participate in cross-cultural group dynamics, to articulate, explicate, and defend their ideas and hidden motives, and to manage their work flow amid a high degree of uncertainty about how the project should be done. At the end, they must create an intellectual product collaboratively.

6.3. The obstacles of e-learning

Together with in a number of advantages e-teaching methods have the number

of **weaknesses**.

Common problems of a e-learning process:

(http://www.certmag.com/articles/templates/cmag_department_ge.asp?articleid=317&zoneid=11):

1. Statistics for online learning place the student drop rate at about 80 percent
2. For students who had finished, retention is a disappointing 30 percent lower than for students using printed material.
3. A static nature of content and the lack of interactivity are a major barrier to success learning

These problems will be solved by application of adaptive environment for the e-learning technologies and application of the combined methods which consist in combination of the e-learning and traditional technologies of studies (http://www.eurodl.org/materials/contrib/2005/Christian_Gutl.htm)

Time and money to create the whole E-learning environment are two key obstacles highlighted by most of the experts. Inadequate and not fully accessible technology is another issue. Finally, many experts underlined the importance of social/cultural barriers (especially: lack of motivation in students; resistance to change in teachers) and obstacles related to training for ICT skills/eLearning and teaching methods.

Time

For some regions a big obstacle is the lack of time to start developing eLearning activities. In particular, lack of time to produce resources and mediate learning, to create the whole eLearning environment and the framework for digital literacy, to prepare special infrastructures for people with special needs, to develop skills, materials/content and to evaluate. Finally, too much time is spent in preparing administrative documentation.

Cost

Costs for individuals

Individuals who are not employed find it hard to afford expensive technology, to build up multimedia contents and to develop E-learning materials. Costs for hardware and software are still too high.

Funding accessibility

For some regions funding for Continuous Professional Development is not easily accessible.

Funding opportunities and budget

There is a lack of external funding opportunities for eLearning and the budget of the European projects is often too small. Gaining initial support for the concept and convincing stakeholders to fund development is another difficulty.

Technology

Access and inadequateness

Full access to technology is often hard to get, especially availability of networked

technology at home. In many cases technology is unreliable and inadequate. Investing in technology sometimes fails to meet its required objectives.

Disabled people

Fundamental design flaws in existing eLearning systems prevent people with sight problems and disabled people in general from accessing them.

Software

Websites are not sufficiently accessible and there are problems with software incompatibility.

ICT skills

Lack of training of both learners and tutors is a huge barrier in general. Where accessible software and technology is available, the training on how to use it often is not.

Social, cultural Barriers

Some regions are concerned about the need to promote learner motivation more effectively, developing a sense of community with and between students. eLearning activities should also motivate those who already use ICT as part of their work and seek a learning experience which is different from their everyday work. There is often a lack of rigor by students in the follow up of their learning pedagogical system. The motivation from students in distance learning has to be very strong, as often the student feels isolated by the learning process which leads to a high drop out rate.

6.4. Adaptive e-learning environment

The Internet has radically changed the way in which we learn and teach. E-learning systems are often not addressing fundamental learning objectives and are not being rigorously evaluated. E-learning is being approached as a technical solution rather than an educational solution.

We are at a change point in the history of human learning. The distinguishing features of this change are:

- With a shift of emphasis from teaching to learning a need for systematic approach about the art of learning is arising.
- Open source movement also fundamentally changes how we learn and teach.
- Information technology and Internet enables new adaptive e-learning environments and systems. In these systems learning standards are extremely important.

In this part of report a adaptive learning environment based on open source movement and learning standards is analyzed. Main idea is that in adaptive e-learning environments all three pedagogical approaches will be integrated using information technology to holistic adaptive environment.

In this approach e-learning standards play a very important role.

The model of adaptive e-learning system are proposed in (<http://www.elearn.cdac.in/eletechIndia05/PDF/02Adaptive%20eLearning%20environment%20for%20students%20with%20divergent-Anandi-02.pdf>)

Conclusion

We are living in an interesting era. The Internet, open source movement, advancements in education theory and information technology changes the way how we learn and teach. As the result of the analysis of main education theories we can conclude that they are not in conflict with each other but can be used in conjunction. Especially influential for learning is open source movement. In this movement in the context of this article open source courseware is most important. Up to this time every educational institution creates educational materials for their own use. Exchange of materials was technically, legally and willfully hard. Now the situation is changing. Educational standards make the exchange of educational resources technically possible. The advancements in open course development like MIT's OpenCourseWare create new understanding how we can cooperate with each other. F/OSS makes all this available for all of us. For teaching and learning F/OSS we can use all spectrum of learning methods:

- Instructor managed classroom learning
- Individual Internet and computer based learning using forums and etc.
- Adaptive learning environment

In real situation we have to use them in combination.

7. Appendix 1: Needs analysis questionnaire

The tOSSad questionnaire for determining training needs for F/OSS is divided into four sections.

Section 1. Organizational Profile and IT Strategy: The first section gathers basic information from the respondents such as the organization type, size and its affiliation within the software industry as well as the current and future consideration of F/OSS in their IT strategy.

Section 2. F/OSS Perception: This section gathers the perceptipn level of F/OSS concepts by the organization. The F/OSS concepts include licensing, cost, and technical concepts such as reliability, security, documentation, etc.

Section 3. F/OSS Experience: This section gathers the level of skill of the organization in F/OSS and software development in general. Various types of software categories (operating systems, networking, data management, etc.) are asked for rating on a scale of 1 to 4.

Section 4. Training Needs in F/OSS: This section is the part where actual training needs are gathered. The questions in this section are either task oriented (where tasks are specifically named) or organization oriented (where job titles are specifically named). In task oriented questions we try to gather the following information for each specific task:

- Frequency of this task to occur among trainees
- The performance gap that occurs between the expected and actual results
- The level needed skill force for this task

The full questionnaire is given below.

Section 1 - Organization Profile and IT Strategy

Q1.1. Please write down your name and surname (optional).

Q1.2. Please write down your country (mandatory) and your organization's name (optional).

Q1.3. Please write down your contact details below (optional).

Q1.4. Please select the option that best describes your answers in this survey.

- a. Speaking for the whole organization.
- b. Speaking for a section of the organization.
- c. Speaking for myself.

Q1.5. If you are responding for a section of the organization, please identify your department (optional).

Q1.6. Please tick all the boxes which describe your job responsibilities.

- a. Software procurement
- b. Training
- c. Software development
- d. Legal issues related to software
- e. Management
- f. Other (Please specify):

Q1.7. Which best describes your organization's type.

- a. Large Corporation
- b. SME
- c. Public Administration
- d. Research
- e. University
- f. Education
- g. Other (Please specify):

Q1.8. Please specify your organization's size.

- a. < 10 employees
- b. 10 - 50 employees
- c. 50 - 200 employees
- d. 200 employees

Q1.9. Which best describes your organization's affiliation with software industry?
Note that there can be more than one.

- a. Developer
- b. Deployer / Administrator / User Support
- c. Consultant

- d. End-User
- e. Education
- f. No involvement with software industry
- g. Other (Please specify): N/D

Q1.10. What is your organization's average level of awareness in F/OSS?

- a. Not aware or slightly aware of open source concepts
- b. We know open source software concepts, but in general further information is needed
- c. Open source has been investigated and decisions have been made

Q1.11. Does your organization have an IT policy or strategy that addresses to consider F/OSS as an option?

- a. No explicit IT policy / Don't know
- b. There is an IT policy, but F/OSS is not mentioned
- c. Our IT policy is of not using F/OSS
- d. Our IT policy explicitly considers F/OSS as an option
- e. Our IT policy considers F/OSS as a preferred option
- f. Our IT policy considers F/OSS as the only option

Section 2 - Perception of F/OSS by Organization

Q2.1. What your relationship with a software vendor should be like?

Straight purchase for a defined set of features

Continual collaboration over time about support and product development

Other (Please specify): N/D

Q2.2. How would you rate the following sentences about open source software?
Rate from 1 to 4 (1= Strongly disagree, 4= Strongly agree)

- a. Open source provides independence from vendors
- b. Open source provides licensing advantages
- c. Open source is overall cheaper than proprietary software

- d. Open source is superior in terms of functionality
- e. Open source software is reliable
- f. Open source software is secure
- g. Open source software issue bug fixes more rapidly
- h. Open source software is more interoperable
- i. Open source software is easy to install and configure
- j. Open source software comes with good documentation
- k. Open source software easily fits with hardware
- l. Open source software is intuitive and easy to use
- m. Open source software meets my overall requirements
- n. Open source software is supported within my organization

Q2.3. What is your level of awareness of licensing issues regarding software and F/OSS?

- a. Absent or very low
- b. I know the basic concepts, but not the details
- c. I am quite knowledgeable
- d. I am an expert

Q2.4. Please rate from 1 to 4 the importance (1= Not important at all, 4 very important) of the following features in evaluating a software product

- a. Cost of the product
- b. Ease of use of the product
- c. A vendor which is still going to be around in five years' time
- d. A company which takes responsibility of the product, and stands to profit if it succeeds
- e. Availability of support contracts
- f. Books and published documentation
- g. Lively newsgroups and mailing lists for support
- h. Training courses
- i. Availability of skilled consultants or contractors able to work with the product
- j. Support from more than one potential vendor

Q2.5. Would you donate money, join a consortium or a user group in order to support a product you value?

- a. Yes
- b. Maybe
- c. No

Q2.6. Would you pay for consulting from a vendor, to help get you up and running quickly and minimizing problems?

- a. Yes
- b. Maybe
- c. No

Q2.7. Would you be comfortable with a free core product, paying for accessories which make it easier to use?

- a. Yes
- b. Maybe
- c. No

Q2.8. Where do you find support for Open Source software in your business or organization?

- a. We don't use F/OSS
- b. 3rd Party
- c. Internet
- d. Books
- e. Newsgroups/Forums
- f. Training
- g. Other (Please specify):

Section 3 - Experience in F/OSS

Q3.1. Please rate from 1 to 4 the skill (1= No skill at all, 4 Significant skill) of your organization in the following kinds of F/OSS usage.

- a. Office automation and other end-user activities (mail client, Web browser...)
- b. System software (operating systems, Web servers, network management...)
- c. Developing software applications using F/OSS components (like DBMS, F/OSS development environments...)
- d. Development of F/OSS applications

Q3.2. Is your organization involved in using F/OSS for developing the following types of software? Mark only the relevant kinds of software.

- a. Operating Systems
- b. Networking Systems (DNS Servers, web servers, email services, VoIP, etc)
- c. Information Systems (Portal frameworks, content management systems, learning management systems, etc)
- d. End user tools (office automation, email or web browser clients, multimedia tools, etc)
- e. Web applications or web services
- f. Software development environments or tools (compilers, debuggers, IDEs, etc)
- g. Network security systems (Firewalls, anti-virus, privacy, etc)
- h. Data management systems (Database management systems, knowledge management systems, etc)
- i. Specialized applications for subject based communities (GIS, engineering systems, statistics, etc.)
- j. Other (Please specify):

Section 4 - Training needs in F/OSS

Q4.1. Please state your job title.

- a. Project Manager
- b. Marketing Manager
- c. Developer
- d. Technical Support
- e. Technical Promoter
- f. Consultant
- g. Other (Please specify): N/D

Q4.2. The following table is to be filled in one of the two following ways:

1. The table will be filled by all of the trainee population individually
2. The table will be filled by the department heads of the manager on behalf of the respective trainee population

For the following tasks, please rate the frequency of occurrence in your department, the difference between actual and desired performance (performance gap), the need of skilled workforce:

Frequency (1 = Not frequent at all, 4 = Very frequent)

Performance Gap (1 = No gap, 4 = Huge gap)

Needed Skilled Workforce (1 = Not needed at all, 4 = Essentially needed)

- a. Eliciting and specifying requirements
- b. Software project management and risk management
- c. Analysis and design of software systems
- d. Developing the code
- e. Version control and source code maintenance
- f. Testing and bug tracking
- g. Producing documentation
- h. User support
- i. Participating to F/OSS projects
- j. Understanding and conforming to F/OSS licenses
- k. Administration (configuring, patching, updating, etc) of used F/OSS applications
- l. Using F/OSS components inside developed applications
- m. Integration and interoperability of F/OSS with other (legacy or 3rd party) systems
- n. Migration to F/OSS
- o. Marketing F/OSS products

Q4.3. Please rate from 1 to 4 the importance (1= Not important at all, 4 very important) of the type of training for F/OSS which your organization prefers.

- a. Hiring graduates with specific training in F/OSS
- b. 1-year master in an Academic setup

- c. Traditional courses and seminars (in-house or at educational institutions sites)
- d. Hands-on the job training
- e. Mentoring and code/document reviews
- f. Courseware for self-training
- g. E-Learning
- h. Face-to-face
- i. Other

8. Appendix 2: Needs analysis questionnaire results

In this section we give the results obtained from the training needs analysis questionnaire. The results are given in parallel with the questions in the questionnaire. The questionnaire itself is also given at the end of the section.

The sample

The sample size is 56, that is a total of 56 responses were collected.

Majority of the respondents (60%) have filled the questionnaire for the whole of their organizations while the remaining has filled it either for a section of their organizations or for themselves.

Job responsibilities of respondents included management (68%), software development (36%), training (30%), software procurement (27%), legal issues related to software (18%) and other responsibilities (18%).

Majority of respondents' organizations were SMEs (60%) followed by large corporation (9%), research (9%), education including universities (9%).

Most organizations had less than 10 employees (48%), followed by 10-50 employees (21%), 50-200 employees (7%) and over 200 employees (7%).

Most organizations described their affiliation with software industry as developer (48%), consultant (45%) and deployer/administrator/user support (39%) followed by education (23%) and end-user (18%). 9% had no involvement with software industry.

Most organizations were aware of open source to the extent that they had investigated open source and made decisions about it (46%) or that they knew open source concepts although they needed further information (29%). Only 11% had no or little awareness on open source.

25% of organizations did not have an IT policy. 27% consider F/OSS as a preferred option in their IT policy and 25% consider F/OSS as an explicit option in their IT policy. Very few do not consider F/OSS in their IT policy.

Perception of F/OSS by Organization

Majority of organizations prefer their relationships with software vendors to be a continual collaboration over time about support and product development (54%) while the rest (33%) prefer straight purchase of software for a defined set of features.

	Strongly disagree	Disagree	Agree	Strongly agree
F/OSS provides independence from vendors			√	√
F/OSS provides licensing advantages			√	√
F/OSS is overall cheaper than proprietary software				√
F/OSS is superior in terms of functionality		√	√	
F/OSS software is reliable			√	
F/OSS is secure			√	
F/OSS issue bug fixes more rapidly			√	
F/OSS is more interoperable			√	
F/OSS is easy to install and configure		√		
F/OSS comes with good documentation		√		
F/OSS software easily fits with hardware		√		
F/OSS is intuitive and easy to use		√		
F/OSS meets my overall requirements			√	
F/OSS is supported within my organization		√	√	√

Table 2: F/OSS aspects

Table 2 summarizes perception of F/OSS by organizations, where responses with a high percentage are marked with √. We see from Table 2 that organizations see independence from vendors, licensing advantages, cost, reliability, security, bug fix speed and interoperability as positives for F/OSS while ease of installation and configuration, documentation, fitting with hardware, ease of use somewhat negative for F/OSS. Overall they are satisfied with F/OSS.

Most respondents (42%) are knowledgeable about licensing issues regarding software and F/OSS while a significant portion (35%) know basic concepts but not the details.

Table 3 summarizes importance of different features in evaluating a F/OSS product, as perceived by respondents.

Only a small percentage (8%) of respondents are not willing to donate money, join a consortium or a user group to support a product they like. A significant percentage of respondents are either willing to donate money, join a consortium or a user group to support a product they like (48%) or they consider it as a

possibility (42%).

Feature	Not important at all	Not so important	Somewhat important	Very important
Cost of the product			√	√
Ease of use of the product			√	√
A vendor which is still going to be around in five years time			√	√
A company which takes responsibility of the product, and stands to profit if it succeeds		√	√	
Availability of support contracts			√	√
Books and published documentation		√	√	√
Lively newsgroups and mailing lists for support		√	√	√
Training courses		√	√	√
Availability of skilled consultants or contractors able to work with the product			√	√
Support from more than one potential vendor		√	√	√

Table 3: evaluating a F/OSS product

Similarly, only a small percentage (4%) of respondents would not pay for consulting from a vendor, to help get you up and running quickly and minimizing problems. A significant percentage of respondents would pay for consulting from a vendor, to help get you up and running quickly and minimizing problems (58%) or they consider it as a possibility (34%).

Only a small percentage (11%) of respondents would not be comfortable with a free core product, paying for accessories which make it easier to use. A significant percentage of respondents would be comfortable with a free core product, paying for accessories which make it easier to use (50%) or they consider it as a possibility (35%).

Most respondents (77%) find support for Open Source software in their business or organization in Internet, followed by newsgroups and forums (52%), books (38%), training (33%) and third party (19%).

Experience in F/OSS

This section has been filled in **51** times.

Table 4 shows skill level of respondent's organizations in relation to F/OSS usage. As we see, skill in office automation and other end-user activities is at the highest level, followed by system software. In software development using F/OSS software and in development of F/OSS applications the skill level is at the extremes at the same time, that is either at significant level or at no skill at all level.

Skill	No skill at all	Not so much but some skill	Quite some	Significant
Office automation and other end-user activities (mail, client, Web browser...)		√	√	√
System software (operating systems, web servers, network management...)		√	√	√
Developing software applications using F/OSS components (like DBMS, F/OSS development environments...)	√	√	√	√
Development of F/OSS applications	√	√	√	√

Table 4: skill levels

Highest level of involvement of organizations in using F/OSS for the development types of software is in web applications or web services (51%), followed by .information systems (portal frameworks, content management systems, learning management systems, etc) (45%), networking systems (DNS, web servers, email services, VoIP, etc) (37%), operating systems (33%), network security systems (firewalls, anti-virus, privacy, etc) (33%), data management systems (database management systems, knowledge (management systems, etc) (33%), end user tools (office automation, email or web browser clients, multimedia tools, etc) (29%), software development environments or tools (compilers, debuggers, IDEs, etc) (20%), and specialized applications for subject based communities (GIS, engineering systems, statistics, etc.) (16%)

Training needs in F/OSS

Job titles of most respondents were project managers (37%), followed by marketing manager (15%), developer (13%), consultant (9%) and technical support (6%).

Table 5 shows the frequency for different tasks performed in the organization.

Task	Not frequent at all	Somewhat frequent	Quiet frequent	Very frequent
Eliciting and specifying requirements		√	√	
Software project management and risk management		√	√	
Analysis and design of software systems	√			√
Developing the code	√			√
Version control and source code maintenance	√			√
Testing and bug tracking	√	√	√	√
Producing documentation	√	√	√	√
User support	√		√	√
Participating to F/OSS projects	√			
Understanding and conforming to F/OSS licenses	√	√		
Administration (configuring, patching, updating, etc...) of used F/OSS applications	√	√	√	
Using F/OSS components inside developed applications	√	√		
Integration and interoperability of F/OSS with other (legacy or 3 rd party) systems	√		√	
Migration to F/OSS	√	√	√	
Marketing F/OSS products	√			

Table 5: task frequency

Table 6 shows the performance gap for the same tasks.

Task	No gap	Small gap	Big gap	Huge gap
Eliciting and specifying requirements		√	√	
Software project management and risk management		√		
Analysis and design of software systems	√	√		
Developing the code	√	√		
Version control and source code maintenance	√	√		

Testing and bug tracking	√	√		
Producing documentation	√	√		
User support	√	√		
Participating to F/OSS projects	√	√	√	
Understanding and conforming to F/OSS licenses	√	√		
Administration (configuring, patching, updating, etc...) of used F/OSS applications	√	√		
Using F/OSS components inside developed applications	√	√		
Integration and interoperability of F/OSS with other (legacy or 3 rd party) systems	√	√		
Migration to F/OSS	√	√		
Marketing F/OSS products	√			

Table 6: performance gap

Table 7 shows the needed skilled workforce for the same tasks.

Task	Not needed	Needed somewhat	Quit	Essentially needed
Eliciting and specifying requirements			√	√
Software project management and risk management		√	√	
Analysis and design of software systems		√	√	
Developing the code		√		√
Version control and source code maintenance	√	√		
Testing and bug tracking	√	√		
Producing documentation		√	√	
User support	√	√		
Participating to F/OSS projects	√	√	√	
Understanding and conforming to F/OSS licenses	√	√	√	
Administration (configuring, patching, updating, etc...) of used F/OSS applications	√	√	√	
Using F/OSS components inside developed applications	√	√	√	

Appendix 2: Needs analysis questionnaire results

Integration and interoperability of F/OSS with other (legacy or 3 rd party) systems	√	√	√	
Migration to F/OSS		√	√	
Marketing F/OSS products	√			

Table 7: needed skilled workforce

Table 8 shows importance of the type of training for F/OSS in respondent's organization.

Type of training	Not important at all	Not so important	Somewhat important	Very important
Hiring graduates with specific training in F/OSS	√	√	√	√
1-year master in an academic setup	√	√		
Traditional courses and seminars (in-house or at educational institution sites)		√	√	
Hands-on the job training			√	√
Mentoring and code/document reviews		√	√	
Courseware for self-training		√	√	
E-Learning		√		
Face-to-face				√

Table 8: Importance of the type of training

9. Appendix 3: TCD Academic Questionnaire and Results

The following questionnaire was sent to all academic staff in the Department of Computer Science at Trinity College Dublin. Results follow.

Teaching Relevance:

1. Open Source Software and development methodologies are academically relevant to my courses	
2. Open Source tools form the platform on which my student work	
3. Open Source methodologies, as opposed to tools, are currently taught in my course	
4. Open Source study is (or could be) a module within the courses at my institution	
5. Open Source is the focus of: a standalone postgraduate or undergraduate degree	
A stream within such a course at my institution	
6. Open Source could be the focus of: a standalone postgraduate or undergraduate degree	
A stream within such a course at my institution	
7. The curricula of my subjects would be enhanced by a focus on Open Source	
8. I use Open Source technology with my student regularly	
9. I make use of Open Source code in Open Source Systems in my teaching	
10. I would like to use Open Source as part of my teaching but do not know enough about it	

Comments:

Research Relevance:

1. I am aware of Open Source projects that are relevant to my research interest	
2. My research is focused on Open Source Software Engineering methodologies	
3. I use Open Source software in my research work	
4. I deliver new, or contribute to existing Open Source software as part of my research	
5. Research projects I am engaged in include the delivery of Open Source software as a deliverable	
6. My teaching is informed by my research work and Open Source helps me in this	

Comments:

Are there any other issues which you believe need to be addressed or future consideration which you think are relevant?

General Comments:

Results from the academic questionnaire follow:

What is your interest in introducing Open Source methodologies into your course?	Average of who responded	Average of those questionnaire
No interest	1	1
Software development and delivery are not relevant to my courses	2	2
Open Source methodologies could be incorporated into one or more of my courses	5	5
Open Source methodologies are not useful to me	1	1
1. I am interested in Open Source Software and associated development methodologies	4.33	4.33
2. I am knowledgeable regarding Open Source methodologies and technologies	3.22	3.22
3. I have been an active contributor to Open Source projects	2.5	2.22
4. I have released software under Open Source licensing	2	1.78
5. I intend to become involved in Open Source Software projects in the future	3.56	3.56
6. I use commercially produced software in preference to Open Source solutions	1.56	1.56
7. Open Source software is generally of high quality	3.89	3.89
8. The Open Source Community is rarely a source of technology innovation	1.89	1.89
9. I approve of the Open Source Community and its approach to technology development and distribution	4.67	4.67
Technology Relevance		

1. Open Source Software and development methodologies are academically relevant to my courses	3.25	2.89
2. Open Source tools form the platform on which my student work	3.2	2.29
3. Open Source methodologies, as opposed to tools, are currently taught in my course	2.17	1.86
4. Open Source study is (or could be) a module within the courses at my institution	2.83	1.89
5. Open Source is the focus of: a standalone postgraduate or undergraduate degree	1	0.22
A stream within such a course at my institution	1	0.22
6. Open Source could be the focus of: a standalone postgraduate or undergraduate degree	1.6	0.89
a stream within such a course at my institution	2.4	1.33
7. The curricula of my subjects would be enhanced by a focus on Open Source	2.17	1.44
8. I use Open Source technology with my student regularly	4.63	4.11
9. I make use of Open Source code in Open Source Systems in my teaching	3	2.33
10. I would like to use Open Source as part of my teaching but do not know enough about it	2	1.56
Research Relevance		
1. I am aware of Open Source projects that are relevant to my research interest	4.44	4.44
2. My research is focused on Open Source Software Engineering methodologies	1.38	1.22
3. I use Open Source software in my research work	4.33	4.33
4. I deliver new, or contribute to existing Open Source software as part of my research	2.29	1.78
5. Research projects I am engaged in include the delivery of Open Source software as a deliverable	1.86	1.44
6. My teaching is informed by my research work and Open Source helps me in this	3.25	2.89

10. Appendix 4: Example of Vocational curriculum: F/OSS system specialist

This curriculum is the subject of an “IFTS” course which is being presently taught in Carbonia, Sardinia, Italy. The F/OSS system specialist is a professional able to manage a small IT infrastructure, based on F/OSS software. Such kind of infrastructure is usually composed by a LAN with one or more servers, and five or more clients, often running Microsoft Windows operating system. The services to manage are LAN management (including wireless networks), Internet connection, Internet services management (mail, Web server, firewall, anti-virus, anti-spam, etc.) file and database server management (directory allocation and organization, simple database administration, LAN data exchange, back-up, recovery, etc.), operating system and main application management.

All these services can be run by F/OSS software, and the F/OSS system specialist has good competences about mainstream F/OSS applications able to give these services.

In deeper detail, the F/OSS system specialist will obtain the following skills:

- Operational knowledge of structured programming methodologies:
 - ability to distinguish among different languages and IDEs;
 - ability to use structured programming techniques;
 - ability to use flow diagrams and knowledge of main standard algorithms;
 - ability to distinguish and use the various elements of a programming language;
 - ability to use control statements, and to define and use functions;
 - ability to adequately document the code;
- Operational knowledge of a server side scripting language:
 - knowledge of language data types and syntax;
 - ability to debug the code;
 - ability to access a database, and to make use of external libraries;
 - ability to manage errors and exceptions;
- Operational knowledge of a client side scripting language:
 - knowledge of language data types and syntax;
 - ability to debug the code;
 - ability to define classes and instances of objects;
 - ability to use scripts for creating interactive Web pages;
 - ability to manage security in Javascript;
 - ability to use cookies;
- Operational knowledge of databases and Web servers:
 - ability to install and configure a DBMS;
 - ability to create and administer a database;
 - ability to configure the remote access to a database;

- ability to install and configure the Web server Apache;
- ability to install and configure the DBMS Mysql;
- ability to install and configure Perl language;
- Understanding and managing network security:
 - knowledge of TCP/IP protocols;
 - operational knowledge of secure protocols;
 - ability to correctly give IP addresses and sub-network masks;
 - ability to evaluate and enforce system security;
 - ability to install and upgrade network anti-virus products;
 - understanding how a firewall works;
 - ability to manage fault tolerance and disaster recovery procedures;
- Knowledge of basic telecommunication principles:
 - knowledge of what a signal is, and what are its power and energy;
 - knowledge of sampling theory, numerical transmission, Nyquist theorem;
 - knowledge of analog and digital modulations;
 - knowledge of information and coding theory;
- Knowledge of wireless networks principles:
 - knowledge of radiomobile apparatuses;
 - knowledge of satellite networks;
 - knowledge of electromagnetic compatibility principles;
- Knowledge of network architecture principles:
 - knowledge of the main architectures and typologies of telecommunication networks;
 - knowledge of main functions of network nodes;
 - knowledge of LANs and metropolitan networks main features;
 - knowledge of digital transmission networks;
 - knowledge of ISDN networks, and of xDSL accessing techniques;
- Ability to manage GNU-Linux systems:
 - ability to install and use a GNU-Linux system, even in presence of other operating systems already installed;
 - ability to use main commands and to write simple scripts;
 - ability to correctly configure and administer a GNU-Linux system;
 - ability to share resources and users;

10.1. Course list and programs

In this section we show the courses of the vocational curriculum, divided in core block and specialist block.

For each block, we show a table summarizing the courses and their length, and then the syllabus for each course.

10.1.1. Vocational Curriculum Core Block

Table 9 reports the names of the courses, the number of hours devoted to theory, to practice (including verification) and the total number of hours.

Course Name	Theory	Practice	Total
Introduction to the curriculum, acquaintance and communication	10	5	15
English language	50	30	80
Basic computer science	15	35	50
Basic Statistics	10	10	20
Business Organization	20	15	35
Basic Business Law and Work Law	15	5	20
The Libre Software phenomenon	10	5	15
Open Source licenses	15	5	20
GNU-Linux operating system	35	35	70
Total	180	145	325

Table 9. The courses of the core block.

Introduction to the curriculum, acquaintance and communication

Effort: 15 hours

Objectives: To get acquainted with the curriculum, the teachers and other students. To develop basic communication and relational skills.

Contents:

1. Getting basic communication skills.
2. Self-promotion and self-evaluation.
3. Analysis of personal skills and resources, matching of these with the external context.
4. Roles within a team.
5. Teams and team-work. Team dynamics.
6. Detecting strengths and weaknesses.
7. Emergence of creative and design skills.
8. Interpersonal communication.
9. Firm communication.
10. Cooperation and conflict.
11. Team productivity.
12. Team decision processes; team problem solving.

English language

Effort: 80 hours (4 credits)

- Objectives: To get basic English knowledge, and medium English technical reading skills.
- Contents: 1. Traditional English course, depending upon the context, with the aim to prepare students to get a TOEFL score of at least 320.
2. Basic words and typical understanding of Computer Science, Communications and Electronics technical jargon.

Basic computer science

- Effort: 50 hours
- Objectives: To get basic knowledge of how a computer works, and how to operate a PC.
- Contents: 1. Architecture and working of a computer.
2. Typical peripherals.
3. Operating system and application software
4. Computer networks.
5. Typical window-based PC environments (Windows and KDE): desktop, windows, menus, icons, etc.
6. Setting and control of printers and peripherals.
7. Structure and use of file systems.
8. Basic ASCII text creation and editing.
9. the Open Office automation suite (word processor, spreadsheet, presentation editor, personal database) and its basic usage.
10. Bitmap graphics and graphic editors. Basic usage of the Gimp editor.
11. Managing and playing multimedia.
12. Internet and Web fundamentals.
13. Web browsers and their use.
14. Web search engines. Examples of search.
15 e-Mail clients and their use.
16. Messaging, chat and VoIP services.

Basic Statistics

- Effort: 20 hours
- Objectives: To get basic knowledge of statistics, its usefulness and its usage.
- Contents: 1. The concept of probability.
2. Variables and distributions.
3. Mean, variance, standard deviation and their best estimates.
4. The main distributions.
5. Relationships and correlations.
6. Random variables.
7. Economic statistics and historical series.

8. Simple and multiple linear regression.
9. Interpreting the results of an interpolation.

Business Organization

Effort:	35 hours
Objectives:	To get basic knowledge of economics and of how firm are organized and operate
Contents:	<ol style="list-style-type: none">1. Economy: microeconomics and macroeconomics2. Market economy and economic circuit.3. The firm as the main tool of economic activity.4. The various kinds of firms.5. The general economic activity fluxes.6. The typical organization of a medium enterprise.7. The business model.

Basic Business Law and Work Law

Effort:	20 hours
Objectives:	To get basic knowledge of what is the Law, and of the main laws ruling business and labor.
Contents:	<ol style="list-style-type: none">1. The juridic system and the concept of law.2. Civil and penal laws.3. Law interpretation and the role of courts.4. Private laws and public laws.5. Hierarchy of rulers: European Union, national governments, local governments.6. The main laws ruling the contract between employers and employees.7. Rights and duties of employers and employees.

The Libre Software phenomenon

Effort:	15 hours
Objectives:	To provide an introduction to the main subject of the curriculum.
Contents:	<ol style="list-style-type: none">1. Traditional software business models.2. A short history of software production.3. What is libre software and its legal implications.4. Development models of F/OSS, developer and user communities.5. Business models of F/OSS software.6. Overview of the F/OSS main actors and applications.

Open Source licenses

- Effort: 20 hours
- Objectives: To address legal and licensing issues related to F/OSS. Students will be prepared to understand the implications of OS software licensing, of choosing a license, of using a software under a given license, etc.
- Contents:
 1. Patents and copyright.
 2. Free Software Foundation
 3. Open Source Licenses.
 4. Choosing a license.

GNU-Linux operating system

- Effort: 70 hours
- Objectives: To get operative knowledge of installation, usage and maintenance of GNU-Linux operating system.
- Contents:
 1. Introduction to GNU-Linux.
 2. Debian distribution as a starting point.
 3. Installation and new releases management of GNU-Linux.
 4. Use of Linux shell (bash script programming).
 5. Managing files and directories
 6. Managing access to system resources
 7. System administration.
 8. Samba, sharing resources in heterogeneous networks.
 9. Tools for managing networks.

10.1.2. Vocational Curriculum Specialized Block

Table 10 reports the names of the courses of this block, the number of hours devoted to theory, to practice (including verification) and the total number of hours.

Course Name	Theory	Practice	Total
Telecommunications	40	10	50
Wireless networks	20	20	40
Network architecture	25	10	35
Communication protocols and data exchange standards	30	20	50
Network and data security	40	25	65
GNU-Linux advanced	25	25	50
Apache Web server	25	25	50
Open source databases	40	30	70

Web programming and Javascript	25	30	55
Perl language (or Python, or Ruby)	25	25	60
PHP language	25	25	50
Total	320	245	565

Table 10: The courses of the specialized block.

Telecommunications

- Effort: 50 hours
- Objectives: Obtaining basic knowledge of communication principles
- Contents:
 1. Elements of mathematics.
 2. Theory of signals.
 3. Discrete signals and numerical transmission in base band.
 4. Modulations
 5. Information theory and codes.

Wireless networks

- Effort: 40 hours
- Objectives: To get knowledge about how wireless transmission networks work.
- Contents:
 1. Radiomobile networks.
 2. Satellite networks.
 3. Electromagnetic pollution and related laws.

Network architecture

- Effort: 35 hours
- Objectives: Introducing the student to the various solutions and standard architectures for making computers communicate each other.
- Contents:
 1. Architecture of a telecommunication network.
 2. Communication networks, circuit switching and packet switching.
 3. LAN and MAN networks.
 4. Digital transmission systems.
 5. Broadband access.

Communication protocols and data exchange standards

- Effort: 50 hours
- Objectives: To know the basic communication protocols used in the Internet. To get acquainted with the importance of standards

and open standards for data exchange among applications.

Contents: Secure and insecure communication protocols.
UDP
TCP/IP basics.
TCP/IP protocol suite: telnet, ftp, mail protocols, http, https, rtp, rtsp.
XML

Network and data security

Effort: 65 hours

Objectives: To introduce students into fundamentals of computer and network security. To show common threats and security-related tools.

Contents: 1. What is computer security.
2. Access control and privileges.
3. Malicious code.
4. Security politics of UNIX environment.
5. VPN
6. Privacy laws.
7. Firewalls and network security.
8. Fault Tolerance and Disaster Recovery.

GNU-Linux advanced

Effort: 50 hours

Objectives: The students should get deeper insight into GNU-Linux system administration, including advanced issues like kernel compilation, package management, system tailoring to user's needs.

Contents: 1. Structuring and tuning the system: files, sw packages, peripherals
2. Managing the boot process.
3. Managing user access and rights
4. Process control and logging
5. Back up, recovery and troubleshooting
6. Controlling and configuring network, peripherals and services
7. The Linux kernel: structure, functions, drivers, recompilation.

Apache Web server

Effort: 50 hours

Objectives: To provide students with the skills to configure and manage a web server, install web pages, setup virtual domains and

manage security using access control.

- Contents:
1. Introduction and basic Apache configuration
 2. Installation and administration of Apache.
 3. Apache Virtual Hosts
 4. Dynamic pages with CGI and PHP
 5. Security and Authentication
 6. Apache customization through modules

Open source databases

- Effort: 70 hours
- Objectives: To provide the students knowledge about the fundamentals of database systems, as well as of management and use of one of the mainstream open source DBMS.
- Contents:
1. Information systems and databases.
 2. The relational model. The SQL language.
 3. Designing a database.
 4. Interfacing a database with GUI applications.
 5. Installation, administration and programming of one of the mainstream F/OSS DBMS: Mysql or Postgresql.

Web programming and Javascript

- Effort: 55 hours
- Objectives: To give students basic knowledge of how to design and implement a Web application from the client perspective. To teach the basics of HTML and Javascript.
- Contents:
1. Introduction to HTML.
 2. HTML content as a tree.
 3. HTML syntax and its usage.
 4. Javascript as a Web programming language.
 5. Variables, strings, numbers, conversions in Javascript.
 6. Control of flow, functions, objects.
 7. Event handling in Javascript.
 8. Advanced techniques, debugging.

Perl language

- Effort: 50 hours
- Objectives: To provide the students sufficiently detailed and hands-on knowledge about Perl language and the related tools. As an alternative, Perl could be substituted by Python or Ruby.
- Contents:
1. Introduction to Perl.
 2. Perl syntax, I/O and flow control, operators.
 3. Strings and data structures.

4. Regular expressions, subroutines.
5. Using and writing Perl modules.
6. Object-oriented Perl.
7. Perl Net Tools

PHP language

Effort:	50 hours
Objectives:	To provide the students sufficiently detailed and hands-on knowledge about PHP language and its usage.
Contents:	<ol style="list-style-type: none">1. Introduction to PHP.2. PHP language syntax and constructs.3. Manipulating strings and other data.4. PHP functions.5. Writing Web pages with PHP.6. Manipulating user input.7. Accessing databases from PHP scripts.8. Error handling and debugging.

10.1.3. Vocational Curriculum Practical Work

Besides following courses, most of which having practical content and including laboratory practice, we deem that is fundamental that students are exposed to real, first hand working experiences.

These experiences should preferably consist in a stage at a firm, or at a public body, performing work with F/OSS software. This work should be done pairing with an expert, and could regard GNU-Linux administration, Web programming (server side or client side) using F/OSS languages and tools, OS DBMS programming or administration, and so on.

Alternatively, the students will take part in a real F/OSS project, under the supervision of an educator.

In both cases, the students will be asked to write a report as a result of their experience, detailing its most important aspects.

The length of the practical work should be of about 300 hours, spanning for two months including the writing of the report.

11. References

- [1] Devedzic, V. (2000), Inside Intelligent Tutoring Systems - Using Design Patterns", International Journal of Knowledge-Based Intelligent Engineering Systems, Vol.4, No.1, January 2000, pp. 25-32.
- [2] Duval Erik. (2003), Learning Objects & ARIADNE – *advanced learning*. ICALT2003 - 3rd IEEE International Conference on Advanced Learning Technologies, 9-11 July 2003, Athens, Greece, keynote presentation. On [www march 2003](http://www.march2003.com).
<http://www.cs.kuleuven.ac.be/~erikd/PRES/Recent.html>
- [3] Hamada, T., Scott K.(2000), Collaborative Learning model. The Journal of Electronic Publishing September, 2000 Volume 6, Issue 1. <http://www.press.umich.edu/jep/06-01/hamada.html> Last checked 29.june 2006.
- [4] Fred de Vries (2006) The Regional Dimension of the EU eLearning Agenda. <http://www.elearningeuropa.info/files/media/media10154.pdf> Checked 29.06.2006
- [5] Moore, M. L., Dutton, P., 'Training Needs Analysis: Review and Critique,' The Academy of Management Review, Vol. 3, No. 3, pp. 532-545, 1978
- [6] McGehee, W. and Thayer, P. W. *Training in Business and Industry*. Wiley, New York, 1961.
- [7] Goldstein, I.L. 'Training in work organizations,' *Handbook of Industrial and Organizational Psychology, Vol. 2* (2nd ed.), M.D. Dunnette, L.M. Hough, Eds. Consulting Psychologists Press, Inc., Palo Alto CA, 1991, pp. 507-619.
- [8] Tannenbaum, S.I. and Yukl, G. 'Training and development in work organizations,' *Annual Review of Psychology, Vol. 43*, M.R. Rosenzweig and L.W. Porter, Eds. Annual Reviews, Inc., Palo Alto CA, 1992, pp. 399-441.
- [9] Goldstein, I.L. and Gessner, M.J. 'Training and development in work organizations,' *International Review of Industrial and Organizational Psychology*, in C.L. Cooper and I. Robertson, Eds. John Wiley & Sons, New York, 1988, pp. 43-72.
- [10] Ostroff, C. and Ford, J.K. (1989). 'Assessing training needs: Critical levels of analysis,' In *Training and Development in Organizations*. Jossey-Bass, San Francisco, 1989, pp. 25-62
- [11] Nelson, R.R., Whitener, E.M., Philcox, H.H., (1995). 'The Assessment of End-User Training Needs,' *Communications of the ACM*, July 1995 Vol 38, No 7, pp. 27-39.
- [12] Wexley, K. N. 'Contributions to the practice of training.,' in *Training and Development in Organization*, Jossey-Bass, San Francisco, 1989, pp. 487-500.
- [13] Gupta, K. "A practical guide to needs assessment", Jossey-Bass/Pfeiffer, ASTD, San Francisco, 1999.
- [14] Arsun, O. I., Kuru, S., "A Questionnaire for F/OSS Training Needs Analysis", Open Source Software 2006 Conference, Como, 2006.
- (from section 8)
- [15]http://www.lincoln.ac.uk/home/courses/dci/postgraduate/open_source/index.asp
- [16]http://www2.shu.ac.uk/prospectus/op_pglookup1.cfm?id_num=CMS030&status=TN
- [17] W3C Software and Notice License, <http://www.opensource.org/licenses/W3C.php>
- [18] Samir Chopra, Scott Dexter, "The Political Economy of Open Source Software"
- [19] Audris Mockus, Roy T. Fielding, James D. Herbsleb, "Two Case Studies of Open Source Software Development: Apache and Mozilla"
- [20] OSI Certified Open Source Licenses, <http://www.opensource.org/licenses/index.php>
- [21] Nordquist, P., Petersen, A., and Todorova, A. 2003. License tracing in free, open, and proprietary software. *J. Comput. Small Coll.* 19, 2 (Dec. 2003), 101-112.
- [22] Jeffrey H. Matsuura, "An Overview of Leading Current Legal Issues Affecting Information Technology Professionals"

References

- [23] "How To Accelerate Your Internet", Editor: Flickenger R. Associate Editors: Belcher M., Canessa E., Zennaro M. Publishers: INASP/ICTP, © 2006, BMO Book Sprint TeamFirst edition: October 2006, ISBN: 0-9778093-1-5 (PDF Available to <http://bwmo.net/>)
- [24] Apache Software License, <http://www.opensource.org/licenses/apachepl.php>
- [25] Academic Free License, <http://www.opensource.org/licenses/afl-3.0.php>
- [26] OpenBSD License, <http://www.openbsd.org/policy.html>
- [27] NetBSD License, <http://www.netbsd.org/Goals/redistribution.html>
- [28] <http://www.sourceforge.net>
- [29] Free BSD License, <http://www.freebsd.org/copyright/freebsd-license.html>
- [30] BSD License, <http://www.opensource.org/licenses/bsd-license.php>
- [31] GNU General Public License, <http://www.gnu.org/copyleft/gpl.html>
- [32] GNU Lesser General Public License, <http://www.gnu.org/copyleft/lesser.html>
- [33] Sun Public Licence, <http://java.sun.com/spl.html>
- [34] Mozilla Public Licence, <http://www.mozilla.org/MPL/MPL-1.1.html>