**video conferencing” support**

“video conferencing” support. By the use of video conferencing tool, virtual class application can be performed

by using tools such as; online chat, file transferring (.pdf, .swf, .doc, .docx, .xls, .xlsx, .ppt, and .pps), whiteboard

application, two side video and voice transfer on a specified date and time.

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**Seniority**

The age of OSS is very relevant. The company gains more confidence when it knows that the product will continue to exist and also will continue to be supported by its developers. In other words, software that is reviewed on a regular basis has fewer errors and bugs and is, therefore, more stable. So, it is very important to take into account the time that the software spent in the market in comparison with the others since a newer one would always be more utilized.

As explained before, the age of OSS is very important. Software that has existed for years is more likely to provide a stable version and gives confidence to the continuity of its existence.

**Age (Product)**

The longer a product remains under active development, the smaller the chance becomes

that the developers suddenly stop. For all Open Source initiatives the first year is the largest

hurdle. Commonly the initiative is halted due to lack of response (lots of work, no glory) or that

the group is too small to sustain the workload that the product generates. As long as there's no

financial compensation for all this effort the group must attract new developers or seek a

product sponsor. Either of these will allow the group to sustain the development effort

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**Licensing**

OSS is presented with a set of rules. It comes with a license that we must understand and accept first.

There are several licenses that organize the use of open source (e.g. GPL, BSD and Apache ... etc.). When thinking of adopting open source software, decision maker has to be

very careful and check the delivered license before,

especially if he/she wants to modify the source code for a

specific use (e.g. a government that has decided to migrate

to OSS, and wishes to adopt it in the various state services,

changes may contain confidential information which

should not be disclosed), because some licenses prohibit

the reuse of a modified code with a proprietary license and

require reproduction with public access. For a better

understanding, we start with the "GNU LGPL V3" license

that allows the modification of the source code but

prohibits the licensing change for the developed program.

Another example is the BSD license, which allows the

change of the modified source code to make it private.

Open Source products can choose from several

different licenses. The choice made tells us

something about the way in which the intended

users are approached. Some are very

restrictive, so restrictive in fact that they become

the subject of the discussion if the product can

still be considered Open Source under such a

license. Some companies try to find ways in

attracting large numbers of non-paid developers

(something Netscape tried for some time),

allowing the company to lower the development

costs. Others offer several licenses, even

catering for commercial variants (MySQL does

this).

*License*: Not all open licenses are equal. Some licenses are more restrictive than

others. If you need to extend the software, copy left properties are important because

they allow modification of the code base and the redistribution of the modified

version as long as the new product stays open

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**Developer community**

An OSS community has many

people who develop software and make it available under a

license. They usually use experience to provide continuous

monitoring, functionality tests and user support as well as

to improve the product and make it competitive.

Consequently, we suggest considering four groups. The

first group consists of individuals (GI) who develop a

project with no guarantee of support or development

continuity. The second is the organization (OR) offering

effective management of the project life cycle, a vigilant

support for various applications and also a website offering

structure. The third is the foundation (Fdn) of an open

source profit. Commercial organization (Commer) is the

last group; its goal is to develop OSS by adding specific

features with more support and maintenance options. The

latter is a paid service.

The saying 'many hands make light work'

certainly holds true when dealing with an Open

Source product. Changes to people's ambitions

and personal life frequently result in that person

moving on to do other things. The greater the

group of active developers the less chance that

the product development stalls. A large group

also requires that the group must organise to

continue to effectively work together. Group

organisation is one of the driving forces behind

an effective community

**+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**Integration**

The "integration" group represents the technical control of

OSS. It helps decision makers to choose the appropriate OSS

product which would operate with their companies’ software

already deployed. It also assists them to know all the served

features and the levels of security. So, we suggest three

criteria:

**Collaboration with other products**

**(Integration)**

As the product gains acceptance within the

target audience the call for the ability to work

with other products is heard more often. Usually

the request to be able to 'script' certain aspects

of the product's functionality is the first feature

that points to interoperability. The next step is

the incorporation of more structural changes,

like using PAM (Plugable Authentication Module,

a generic interface to allow interaction between

a product and a separate authentication system)

for example. Collaboration with other products is

therefore the result of change requests that have

been accepted by the development team. So it

isn't just the product that is collaborating.

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**Interoperability**

Today, it is becoming more and more

necessary to select computer hardware and software that

validate their ability to function and communicate with

other suppliers products. This is called interoperability.

The different OSS maturity assessment models do not treat

the interoperability thoroughly.

To measure interoperability, it is sufficient to confirm

whether OSS is compatible with the existing technology.

Moreover, information can be found very easily.

Interoperability is the ability of different computer systems, applications or services to communicate, share and exchange data, information and knowledge in a precise, effective and consistent way (Martínez & Navarra, 2007). In the eLearning field this topic is extremely important since there is the need for all systems that typically compose an eLearning environment to communicate and share data consistently. The LMS plays a central role in any eLearning architecture. Choosing an LMS can be a challenging task for an organization. Several studies have been conducted to analyse and evaluate these types of systems from pedagogical and institutional perspectives (Pantel, 2007; Britain & Liber 1998). However, we are not aware of any study to evaluate the interoperability of LMSs with other systems typically found in an educational institution. A major issue in LMS interoperability is the eLearning standardization.

These interoperability issues affect the flexibility of the teaching-learning process and lead to a decrease of end user satisfaction and learning success.

Recently the eLearning community started valuing more the interchange of course content and learners' information, which led to the definition of standards for eLearning content sharing and interoperability. Standards can be viewed as "documented agreements containing technical specifications or other precise criteria to be used consistently as guidelines to ensure that materials and services are fit for their purpose" (Nichani, 2009). In the eLearning context, standards are generally developed for the purposes of ensuring interoperability and reusability in systems and of the content and meta-data they manage. In this context, several organizations (e.g. IMS GLC, IEEE, ISO/IEC, ADL) are developing specifications and standards (e.g. IMS CP, IMS CC, IMS DRI, LOM, SCORM) in the last years (Dagger & O'Connor & Lawless & Walsh & Wade, 2007). These specifications are closely related with the learning object concept as context.

**Modularity (Integration)**

As an Open Source product gains more market

share, others could become more interested in

parts of the product's functionality. This allows

the developers to develop a more flexible

licensing scheme (protecting the core, but

allowing fewer restrictions on other parts), which

could even allow commercial developments to

hook up to parts of the system. By splitting the

product into several modules commercial

interest can be attracted without sacrificing the

Open Source principle. This is what happened

with Xfree (XFree86), the X-server used by most

Linux distributions. Previously video card

manufacturers where required to give out all the

card's inner details to get Xfree to work with their

card. This was changed to allow (closed source)

binary drivers to hook up with the rest of Xfree.

The users gained access too much more recent

and powerful hardware and video card

manufacturers were now more involved in

supporting Xfree. The opposite also happens,

commercial products are offered in a trimmed

down version to users as Open Source, while

the full product remains commercial (closed

software).

*Compliance with standards*: for several application domains compliance with

standards is important. For instance, in a website implementation, valid W3C-HTML

code is a first step toward more compatibility with browsers and better rendering of

pages. Using only strict HTML (that is, the Strict HTML DTD) makes the site easier

to maintain and evolve.

++++++++++++++++++++++++++++++

*Books*: the availability of books about the software is a strong indicator of the

software’s level of maturity and popularity

++++++++++++++++++++++++++++++++++++++++

**Security**

The advantage of using OSS is that you can

access to its source code, analyze and detect any

abnormalities that may present a security risk. As the

source code is analyzed by hundreds of people within the

community, developers provide a rapid response to any

critical demand compared to proprietary software. We

acknowledge that we cannot reach a 100% safety level. So,

there should be a continuous follow-up to improve the

system with upgrades, patches ... etc

**Functionality**

it is vital to determine the different features

that a piece of software has to offer in terms of safety,

license, regulatory, support, and documentation. We take

as an example a commercial company that has deployed

new customer management software CRM with some

features. Later on, it turns out that the system cannot be

used to configure e-mails and send messages to customers;

this requires either special development which would

increase the adoption costs or selection of an appropriate

solution.

It is necessary to

ensure that the OSS that we wish to adopt provides all the

functionality required by users.

**Training**

Any change in management requires a good

organization to succeed. When adopting a new OSS that

will replace another system (e.g. the case of the

implementation of a multi-module ERP system), it is

strongly recommended to train a staff and improve their

skills. For this reason, we insist on "training" as a criterion

in the maturity model. So, we should investigate in

advance about the training available for the targeted OSS

before its selection.

**Documentation**

it is a paperwork used to identify a system

and linked safely to its destinations. As stated before, it is

necessary to be sure of the availability of documentation

that would serve in two different parts: the first is the

“user’s documentation” describing how to use the software

and its various features. The second is the “system

programmer’s documentation” that explains the source

code and how to modify it for any need by adding or

changing its functioning. In addition, it is important to

check the availability of the FAQ tool (Frequently Asked

Questions) which provides some advice about the

application usage. It goes without saying that the OSS

forums remain of great help to users who need assistance.

After a bibliographic research, we have found that the

documentation is a key element in the OSS selection

**Support**

it is the key element for the survival of any system. It is a decisive factor that provides solutions especially to the IT teams who want to ensure the smooth running of business systems and continuity of production. According to the survey we have conducted among 200 Moroccan

SMEs, 56% of the businesses that have responded do not

wish to set up an OSS because for them it is developed by a

team of volunteers and not by a professional organization.

There is also a fear that it might not be as efficient as

possible. To remedy to this situation, the OSS communities

have begun to offer detailed descriptions of the developed

systems so as to understand its functionality and use it

easily. Therefore, some commercial companies offer paid

support service for OSS (e.g. OpenERP) with the

possibility to have access to 24/7 support. As far as support

requests are concerned, we can classify them into three

categories: the first is an emergency "High", for example

the shutdown of the electronic payment system which may

have a negative impact on the turnover of the company.

The second is of a "Medium" emergency requiring rapid

intervention to prevent the loss of confidential data. Finally,

an emergency "Low" when it concerns simple errors of

application that require only a fix.

The open source software support is a relevant criterion.

We identify three types of available support service in the

literature: self-support made by the user himself, support

provided by the community that has developed the software,

and paid support offered by a commercial organization.

**Support (Use)**

With some products the support is obtained by

mailing the single developer. Others maintain a

discussion group (or even groups), but only a

couple of regulars respond to request. A few

maintain very active discussion groups in which

a large number of members will offer support.

Some products will even guarantee support if

you pay them. The manner in which support is

given or offered says a lot about the way the

development group takes its users seriously

+++++++++++++++++++++++++++++++++++++++++++++

**Ease of deployment (Use)**

If a product becomes so popular that even

independent parties start to offer training

courses, it is almost a certainty that it has

become a mature product; the very least it has

become a very popular product. More commonly

seen with Open Source is that active users start

writing task specific papers (HOWTO's). These

HOWTO's allow new users to accomplish the

desired functionality without having to master all

of the product capabilities. HOWTO's cover all

aspects of product usage, not only how to set-up

for a particular application, but also how to

maintain the product. Existence of

documentation detailing day-to-day maintenance

is an indication of maturity.

**User community (Acceptance)**

Some products generate hardly any noise; some

have several busy discussion groups. When an

Open Source product is well received it is

common to witness an outburst of user requests,

suggestions and problem reports. The

discussion group quickly fills up with large

numbers of messages, so the developers must

expand and start to manage this huge flow of

feedback. This could be described as an Open

Source project's puberty. An active community is

not to be underestimated. When Sybase

stopped all development on Watcom C++ (a

developers tool) the community rallied and

negotiated an open source option. Today it lives

on as Open Watcom C++.

**Market penetration (Acceptance)**

The installed base tells us something about the

importance of the product within the intended

users. A product with a large installed base

(Apache for example) provides additional

stimulus to form communities. Users will want to

voice new requirements, discuss problems, and

therefore require a platform to do so. Some

users may have clear ideas on how to advance

the products; others will appreciate the

possibility to communicate directly with

members of the development team. A lager

installed base indicates a more mature product

**+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**Probability of support in the feature**

General it is needed that a software product is supported as long as it is in use. We

can estimate if the OSS being evaluated will be supported in the future through an indepth

analysis of the community of OSS developers, assessing:

􀁸 The “vitality” of the product, indicated by its age and the number and frequency

of releases. The number of companies involved in the development. A large number of

companies is a good index of probability for a continuative support.

􀁸 The number of developers per company is useful to understand how important –

or even “strategic”– each company considers the OSS product.

􀁸 The number of independent developers is also relevant, since a large community

of developers guarantees a continuous development and maintenance effort.