A Formal Selection Approach for Digital Learning Environment Implementation

Morad AKTAM¹, Abderrahim TAHIRI² and Kamal Eddine El KADIRI³
morad.aktam@gmail.com  abderahim.tahiri@gmail.com  elkadiri@uae.ma

Correspondence concerning this paper should be addressed to Abderrahim TAHIRI, Ecole Nationale des Sciences Appliquées de Tétouan - Avenue de la Palestine - BP 2222 Mhanech II Tétouan Morocco. E-mail: abderahim.tahiri@gmail.com

¹ Software Engineer Collaborate with Abdelmalek Essaïdi University-Tetuan-Morocco
² Professors-Researchers of Abdelmalek Essaïdi University-Tetuan-Morocco
Abstract

Many empirical studies on software selection methods agree that most organizations still select their products using ad hoc selection methods. However the use of such methods may lead to adverse effects, like overhead in costs and even the rejection of the software by the end users. One of the main reasons is the lack of needed support and guidance during the all selection process phases. Besides that, many of these methods do not include important intangible factors that may affect the success of any implementation of an information system.
This paper proposes a context-based formal software selection method which deals with the abovementioned challenges. In addition, the proposed method emphasizes the importance of the functional and the usability evaluation phase and. It also suggests the appropriate software quality aspects, metrics and simple mathematical formulas for assessing the suitability of evaluated software.

The proposed method has been applied during a PSI (Package Selection and Implementation) process. This PSI has been conducted in the National School of Applied Sciences of Tetouan (ENSATE) in order to select a suitable open source learning management system. The most phases of this PSI project are described in this paper to offer potential method users a real example on how they can apply it.

Introduction

Everybody agrees that our lives are dramatically changed the last or two decade(s) ago thanks to the huge impact of the Information technology on almost every side of our daily routines and everybody’s behaviour. Yet, as much as these possibilities offer many opportunities, we should do more efforts to stay updated and skilful with new technologies and applications. In addition, it is commonly known that the more choices we have, the more difficult it becomes to make the right choice and decision. In order to make an appropriate decision we first need to know the problem, the need and the purpose of the decision, the decision criteria, their sub-criteria and the affected groups (Saaty, 2008). Furthermore, the selection of appropriate techniques and methods is very important to succeed any selection process. Therefore evaluating and selecting software packages that meet the organization’s requirements is considered to be difficult software engineering process (Anil Jadhav, Rajendra Sonar, 2009).

Many selection methods have been developed in order to help organizations to win this specific challenge. However these methods vary in their scope and approaches which they are based on. This variety presents another challenge for the organization in terms of choosing the appropriate selection methods to their context. Many empirical studies have been conducted in order to compare formal selection methods. (Anil Jadhav, Rajendra Sonar, 2009) suggested based on their study that 1) analytic hierarchy process has been widely used for evaluation of the software packages (2) there is a lack of a common list of generic software evaluation criteria and its meaning, and (3) there is a need to develop a framework comprising software selection methodology evaluation techniques, evaluation criteria, and a system to assist the decision makers in their software selection.
Another empirical study that analysed eleven prominent COTS selection methods, suggested that most developers still select COTS products using ad hoc methods due to the lack of support and guidance for applying the suggested methods (Tom Wanyama, Behrouz H. Far, 2008). While another empirical study shows that the OSS (Open Source Software) developers almost always use own developed selection approaches. In these approaches the context constraints are more important than the criteria proposed in general evaluation schemas. Furthermore, OSS developers prefer a “first fit” strategy rather than the “the best fit” approach based on formal selection methods (Øyvind Hauge, Thomas Østerlie, Carl-Fredrik Sørensen and Marinela Gerea, 2009).

Based on abovementioned facts we concluded that a successful formal selection approach should consider the next aspects:

1. Selecting the most important selection criteria which are common between all software, which may affect directly the final decision. While allowing the user to define the rest of the criteria which are more specific to his context. The method should also support the evaluators by giving important criteria definitions and suggesting techniques and metrics in order to quantify these defined criteria;

2. Offering an overall generic formula for the calculation of the final score which may also include any potential user defined criteria;

3. Providing sufficient explanation about the selected approaches in order to convince the user about their importance.

The benefits of this approach reside in: 1) avoiding too many details that may make the method unpractical due to the needed implementation effort in term of cost, time and resources. 2) Making the method adaptable for different contexts by suggesting a general framework and allowing evaluators to customize it based on their evaluation constraints.

In the following sections we will explain our proposal for a generic formal selection method that would be used by organizations to select either COTS or OSS products.

In order to give an operational explanation for our method, we chose to explain it along with its implementation on a Package selection and implementation project conducted at the ENSATE. The goal of this project was the selection of an open source DLE (Digital Learning Environment) which meets the ENSATE requirements.

**Business Case Description**

ENSATE realized that an effective communication between different participants of the educational environment is obviously one of the major factors influencing the quality of education.
This fact has been proven in many academic studies (Morreale, Sherwyn P. Pearson and Judy C, 2008). It is also commonly agreed that the education in the 21th century should combine practical, intellectual and social skills as never before. This is as a response to the student’s expectations which are deeply influenced by the use of information technologies. Digital learning environment (DLE) is one of the trends which are widely implemented by organizations in order to reach earlier mentioned purposes. However the selection of the best DLE that meets the organization requirements is not an easy process. DLE’s vary in their scope, possibilities, applied learning methodologies and technical aspects. For this reason ENSATE decided to start a PSI project in order to find a DLE that fits adequately in its context. The main functionality of the upcoming DLE is to facilitate remote communication in general and in particular the share of documents between teachers and students in a structural, effective and secured way. In addition, the implemented DLE should be extensible and adaptable in order to fit both the current requirements as well as the future strategic and organizational changes.

**Description of the Selection Method**

**A Formal Selection Approach**

In order to have a successful products evaluation, a formal process is needed in order to fulfil an adequate evaluation of the products and their vendors.” The term formal means having an established and documented process to perform selection and evaluation of activities in a consistent, quantifiable and repeatable manner.” (Bandor, 2006). By using a formal method, it is possible to mix different tangible and intangible criteria into a cohesive decision. Moreover, it is important that this method must cover all aspects and evaluation phases starting from the initial pre-selection up to the final decision. According to (Anil Jadhav, Rajendra Sonar, 2009) The main activities involved in evaluating software packages are: identifying evaluation criteria, assigning weights to each criterion, setting up a rating scale for each criterion, calculating the score and finally to rank the alternatives and select the best one. In addition to these activities, we consider forming the evaluation team and the pre-selection phase being very important activities which may have big impact on the end evaluation result.

**Forming the Evaluation Team**

The primary consideration during the establishment of the evaluation team is that it must include representatives from different viewpoints like stakeholders and end system users. In ENSATE case the team was consisted of:

Two teachers. One of them is also the chief of the software engineering department. He represented the management of the school. The other teacher represents the teacher’s viewpoint.
and the platform administrator as this role will be assigned to one of the teachers in the future;

Five senior students from the software engineering program. They represent the student’s viewpoint which forms the largest users segment. The choice of senior students is made because such students possess sufficient knowledge about the software quality aspects and the COTS or OS development.

**Identifying Evaluation Criteria**

The most crucial phase during the selection process is the identification of selection criteria out of the many existing software quality aspects in order to obtain reliable selection results. These results must lead evidently to the acceptance of selected system within the organization in question. Many theories and studies have been conducted in order to reveal why users accept or reject a certain information system. Certainly is the Technology Acceptance Model or TAM (Davis, 1993) is the most adopted and the most cited in researches that deal with user acceptance of information systems. Davis hypothesized in his model that the user attitude toward the use of a system determines whether he will accept or reject it. While the attitude toward the system is determined by two other variables namely, how user perceived usefulness and the ease of use of the system. Based upon this fact, we suggest that high priority should be assigned to evaluation of the functional aspects (usefulness) and the usability aspects (ease of use). However a big mistake committed by a lot of organizations is the restriction of the evaluation to only the functional aspects of the product. Conversely, other non-functional (intangible) aspects may lead to big problems in term of usage, efficiency, maintenance and continuity of the software. (Christian Litke, Michael Pelletier, 2002) State that many costs that organizations need to worry about for the long term—are hidden. Some examples of these intangible factors indicated are shown in the next Table (Christian Litke, Michael Pelletier, 2002):

<table>
<thead>
<tr>
<th>Area</th>
<th>Intangible factor</th>
<th>Consideration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business and Functional</td>
<td>Are you going to change?</td>
<td>Are your organization's business processes/requirements subject to a large</td>
</tr>
<tr>
<td>Functional Requirements</td>
<td></td>
<td>amount of change?</td>
</tr>
<tr>
<td></td>
<td>What is the scope?</td>
<td>Is the system being reused across many areas?</td>
</tr>
<tr>
<td></td>
<td>Is it overkill?</td>
<td>Are you buying features that can't be used or that could have an effect on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the architecture?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In addition to the earlier mentioned criteria other categories need to be considered and evaluated are the costs and the security.

### The Requirements Document

In practice, this document can be derived from the functional design document if this is already composed because the organization intended to build the needed functionalities. Otherwise composing the requirements document should be the first building block in the evaluation process. Moreover this document presents the basis of the most work coming afterwards.

The organization may follow many strategies to be sure of including the necessary criteria from the different viewpoints. Interviews with all stakeholders and future system users are one of the essential options. Besides that, analysing an existing system or the software demos may help in gathering more specific software criteria’s (P. van Staaden, and S. Lubbe, 2006).

### Prioritizing Requirements Categories and Criteria

In the proposed method we will reuse principles of two techniques used widely in the evaluation studies namely:

<table>
<thead>
<tr>
<th>Usability</th>
<th>Can other people work on it?</th>
<th>Does the software require specialized language training or techniques to use it or integrate it into the system?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Remember the end user!</td>
<td>Will integrating this software require additional training or changes to the process?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maintenance</th>
<th>What is Time / cost needed to modify the software?</th>
<th>Interface development may still be needed to integrate the software or fully take advantage of its features.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Integration</th>
<th>How well does it integrate with the other applications within the architecture?</th>
<th>If the software doesn't integrate well, it may be necessary to make a significant change to the architecture. Remember that time is not on your side!</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Support</th>
<th>What kind of documentation and support are available?</th>
<th>If there is a lack of documentation and support which meet your company’s needs.</th>
</tr>
</thead>
</table>

Table 1: Examples of Intangible Factors
The AHP (Analytical Hierarchical Process). This technique will be used for assigning weights or priorities scales for main criteria categories. And also for several sub-criteria.

The AWS technique (Average Weight Sum) for calculating the average score of each criterion. And finally for calculating the overall score of each evaluated system.

The Analytic Hierarchy Process (AHP) has been developed by (Saaty, 2008). It allows users to assess the relative weight of multiple criteria or multiple options against given criteria in an intuitive manner even when quantitative are not available (N. Kasperczyk, K. Knickel).

Supposing we have two system criteria A en B and Criterion A is two times more important than criterion B. In this case Rating the relative “priority” of the criteria is done by assigning a weight between 1 (equal importance) and 9 (extreme importance) to the more important criterion, whereas the reciprocal of this value (1 till 1/9) is assigned to the other criterion in the pair. The next table shows an example of two criteria:

<table>
<thead>
<tr>
<th></th>
<th>Criteria A</th>
<th>Criteria B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criterion A</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Criterion B</td>
<td>1/2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2: Examples of System Criteria

After assigning the importance scale of each criterion, we can than calculate the weight of each criterion by the next formula:

\[
\text{Weight(Criterion A)} = \frac{\sum \text{all priority scales in Row Criteria A}}{\sum \text{All priority scales}}
\]

The Requirements Main Categories

In the proposed selection method we recommend evaluating the systems on next aspects categories:

- Functional aspects;
- Usability aspects;
- Maintainability of the software by evaluating its extendibility and its complexity;
- Support;
- Security;
- Costs.
Applying the AHP approach on the ENSATE Context we got the importance scales figured below in table 3.

While assigning the importance scales we have considered the next aspects:

- The functionality and usability criteria must have the highest priority. This because Functionality (usefulness) and usability (ease of use) are the most important factors which determine whether a system will be accepted or rejected by the end users (Davis, 1993);
- According to the same Study, Davis found that perceived usefulness is 50% more important than perceived ease of use in determining user acceptance for a given system;
- The importance of other criteria depends on the evaluation context. Security may have the highest priority in safety critical systems for example.

<table>
<thead>
<tr>
<th></th>
<th>Functionality</th>
<th>Usability</th>
<th>Maintainability</th>
<th>Extendibility</th>
<th>Support</th>
<th>Security</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>1</td>
<td>3/2</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Usability</td>
<td>2/3</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>3/2</td>
</tr>
<tr>
<td>Maintainability</td>
<td>0.2</td>
<td>0.25</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1/5</td>
<td>1/6</td>
</tr>
<tr>
<td>Extendibility</td>
<td>1/7</td>
<td>1/6</td>
<td>0.5</td>
<td>1</td>
<td>2</td>
<td>0.2</td>
<td>1/3</td>
</tr>
<tr>
<td>Support</td>
<td>0.2</td>
<td>0.25</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
<td>1/6</td>
<td>0.2</td>
</tr>
<tr>
<td>Security</td>
<td>1/3</td>
<td>1/3</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>1/3</td>
</tr>
<tr>
<td>Cost</td>
<td>0.5</td>
<td>2/3</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3: Scales by Applying the AHP Approach
Based on the earlier importance matrix, we obtained the following criteria weights:

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>0.258413</td>
</tr>
<tr>
<td>usability</td>
<td>0.212707</td>
</tr>
<tr>
<td>maintainability</td>
<td>0.061351</td>
</tr>
<tr>
<td>extendibility</td>
<td>0.045806</td>
</tr>
<tr>
<td>support</td>
<td>0.029709</td>
</tr>
<tr>
<td>security</td>
<td>0.189854</td>
</tr>
<tr>
<td>Cost</td>
<td>0.20216</td>
</tr>
</tbody>
</table>

Table 4: Criteria Weights

**The Final System Grade**

The final system grade will be calculated based on the weighted average:

$$ \text{Grade(System A)} = \frac{\sum_j \text{Weight} \times \text{Score } (\text{Criterion}(j))}{\sum_j \text{Weight}(\text{criteria}(j))} $$

According to AHP approach,

$$ \sum_j \text{Weight}(\text{criteria}(j)) = 1 $$

This simplifies the final Grade into following:

$$ \text{Grade(System A)} = \sum_j \text{Weight} \times \text{Score } (\text{Criterion}(j)) $$

Applying this on ENSATE case, we got the following final grade for evaluating the preselected DLE’s:
It is very import to normalize each obtained criteria score between 0 and 10. This in order to avoid any fault compensation due to different score intervals.

**Functionality Evaluation**

The functional evaluation is based on a decision spread sheet including all required functions and system features. These features are subdivided into sets based on the existing system user roles. This classification will facilitate the assignment of evaluation to the right evaluators. Furthermore, Each Feature or function will be evaluated using the following metrics:

<table>
<thead>
<tr>
<th>Feature Id</th>
<th>Name</th>
<th>Description</th>
<th>Priority</th>
<th>Available</th>
<th>Steps</th>
<th>Path</th>
<th>Errors</th>
<th>Easiness</th>
<th>Learnability</th>
<th>User interface</th>
</tr>
</thead>
</table>

Each of the defined metric should be scaled based on an operational definition. This definition provides the evaluator a clear understanding of what kind of work is acceptable. (Deming, 1982) stated that Operational definition increase the consistency of the obtained results. The next suggested scales and their definition can be adapted to the evaluation context.

1- Priority: This scale indicates how important a given functionality is for the client or for an adequate functioning of the DLE. The values of the priority metric will range between 2, 4 or 6.

- Value 6: means that the given functionality or feature is critical for the functioning of the DLE or it is considered as very important for the client.

- Value 4: means that the given function is important either for the client or for the functioning the DLE but not critical.

- Value 2: means that the given function is optional and its absence doesn't affect an adequate functioning of the DLE.

2- Availability: this metric may take two values. 1 when the function exists in the software and 0 if not.
3- Steps: this represents the sum of the clicks which the user needs to perform from the home page till the execution of the whole function. Clicks for Moving between fields are not included. This metric indicates the amount of effort a user performs to achieve a task.

4- Path: it stands for the path that the user needs to follow starting from the homepage in order to find the desired activity or the function. This feature may be used for any evaluation review.

5- Errors: it represents the sum of errors encountered during the execution of the function multiplied by their severity scale. Error severity scale may take 3 values: Infinity $\infty$ when an error causes a system crash or complete execution failure of the given function. 3 when an error causes a partial execution failure of the given function. 1 when an error causes a partial execution failure of the given function.

6- Easiness: This scale can take the next values which indicate how easy is to carry out the giving function or activity: 1 if is too difficult or complicated, 2 if is difficult or complicated, 3 if is normal, 4 if is easy and 5 when is very easy.

7- Learnability: (B. Shneiderman, and C. Plaisant, 2005) defined it as “the time it takes members of the user community to learn how to use the commands relevant to a set of tasks”. The assigned scale for the learnability indicates how easy and quickly was for the user to find and execute the function the first times he used the DLE. 1 if is too difficult, 2 if is difficult, 3 if is normal, 4 if is easy and 5 if is very easy.

8- User interface: This aspect indicates how pleasant is the interface which is used to perform the giving function. This factor can have the next values: 1 if is very bad, 2 if is bad, 3 if is normal, 4 if is pleasant and 5 if is very pleasant.

The average score of each function is calculated using the next formula:

$$Score(Function) = \frac{priority \times availability \times (easiness + learnability + user\ interface)}{\sum_{i=1}^{j}Severity\ of\ Error(i) + steps}$$

where $j$ is the total of errors encountered during the execution of the function.

The final Functionality score for the system will be calculated by the next formula:

$$Functionality\ score(DLE) = \frac{\sum_{j=1}^{j}Score(Function(i))}{j}$$

where $j$ is the total of evaluated functions.

**Non-Functional Evaluation**

Intangible criteria can be related either to the system self or to the system provider as well.
Intangible criteria applied on the system are commonly known as the non-functional requirement. Non-functional requirements can be defined as constraints and the quality attributes related to a system. While Intangible criteria on the system provider are related to its capability to guarantee the continuity of the supplying adequate support to the system users. Many of these criteria are difficult to quantify since the evaluation of such intangible notions is more or less relative and subjective. However any organization needs to take them in consideration as they affect either the acceptance of the system or its continuity in the future.

**Extendibility**

By the extendibility of the system we mean the ability of the system to be extended by extra functionalities in the form of plugins or modules which are developed either by the system provider or the system community.

In this evaluation method the extendibility score will be calculated by the next formula:

\[
\text{Extendibility(DLE)} = \frac{\sum_{i=1}^{n} \text{Relevance scale item}(i)}{n}
\]

Where,

- Item : a found plug-in or module;
- Relevance: Scale between (1 and 10) indicates the relevance of the item to the organization context;
- \(n\): total of items found

**Maintainability**

According to the IEEE Standard Glossary of Software Engineering Terminology (IEEE, 1990), the maintainability is defined as “The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment”.

There are many factors that may influence the maintainability of the system. In our proposal we limited the study to the assessment of the software complexity and the technical support provided by the software provider or community. Hereby we emphasize that this evaluation must be conducted by users who may be involved in the future maintainability projects. In the ENSATE case, the students and teachers are the best candidates for this purpose.

The Maintainability grade will take the following values: 3 if is too complex, 5 if is complex, 7 if normal, 8 if is easy and 10 if is too easy.

In order to find the answer on the previous question we need to assess the complexity of the source code.
The complexity of the source code: According to (IEEE, 1990) the system complexity can be defined as “The degree to which a system or component has a design or implementation that is difficult to understand and verify”.

To most practical approach to assess the complexity of system is using of static metrics. These metrics could be subdivided into formatting metrics and logical metrics:

Formatting metrics: deal more with aspects like indentation conventions, comment forms, whitespace usage, and naming conventions and so on. In the ENSATE context, we let students to assess this aspect. Because it has no sense if we have thousand lines of comments if none understand them. Hereby we talk more about a qualitative and not quantitative of comments.

Logical metrics: deal with such things as the number of paths through a program, the depth of conditional statements and blocks, the number of parameters and arguments used etc. In this context we limited our measurement to the Cyclomatic metric of the systems. “It is probably the most useful logical metric.” (Charney, 2005) stated “It is probably the most useful logical metric”

McCabe Cyclomatic Metric: The McCabe Cyclomatic Metric was introduced by Thomas McCabe in the year 1976.

\[ M = E - N + X \]

Where \( M \) is the McCabe Cyclomatic Complexity metric, \( E \) is the number of edges in the graph of the program and \( N \) is the number of nodes or decision points in the graph of the program and \( X \) is the number of exits from the program.

The usability of the software

Usability has many definitions. The ISO 9241-11 provides guidance on usability and defines it as: “The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.” Thus Usability is more or less about:

- Effectiveness - can users complete tasks, achieve goals with the product, i.e. able to do what they would like to do?
- Efficiency - how much effort do users require in order to do this? We calculated it in terms of sum of clicks which performed by the use which is proportional to the time needed to execute tasks.
- Satisfaction – what do users think of about the products ease of use?

According to Usability Expert Jakob Nielsen, Leaving is the first line of defence when users encounter a difficulty with software or a web application in particular:

“On the Web, usability is a necessary condition for survival”. In this research work some usability aspects are analysed during the functional evaluation. Namely, the needed effort, the easiness, the
learnability and the user interface for each function or activity. This type of technique is called the feature inspection. However, this technique is not sufficient to get an overall rating for the usability. In addition to this technique, we chose to use the Heuristic evaluation technique. The general idea behind heuristic evaluation is that several evaluators independently evaluate a system, this in order to come up with potential usability problems. It is important that there will be several of these evaluators and that the evaluations should be done independently. Nielsen’s experience indicates that around 5 evaluators usually results in about 75% of the overall usability problems being discovered.

The evaluators need to answer questions as presented in the following table:

<table>
<thead>
<tr>
<th>#</th>
<th>Review Checklist</th>
<th>Yes</th>
<th>No</th>
<th>N/A</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Does every display begin with a title or header that describes screen contents?</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Questions for Answer by the Evaluators

We used the finding from heuristic checklist to calculate final usability score for each solution:

\[
Usability\ \text{score} = 3x\sum yes - \sum no
\]

**Support**

(IEEE, 1990) defines support as “The providing of information, assistance, and training to install and make software operational in its intended environment and to distribute improved capabilities to users”.

We used the same formula as by the evaluation of the extendibility in order to rate the support aspect:

\[
Extendibility(DLE) = \frac{\sum_{i=1}^{n} Relevance\ scaleitem(i)}{n}
\]

Where item may be a technical documentation, user guides, tutorials, security updates etc. The answer on the previous question will be presented also in a grade between 1 till 10.

Initial pre-selection

(Bandor, 2006) state that “An inappropriate pre-selection strategy for COTS products can lead to adverse effects”. It could result in a short list of COTS products that may not be able to fulfil
the required functionality. In addition, it might introduce overhead costs in the system integration and maintenance phases. To avoid this problematic the initial selection must be based on different aspects to guarantee an objective pre-selection:

- Information and reviews on websites which are specialized in system context;
- The vendors or providers documentation found on the official websites;
- Communication with other users using or they have already used some of these systems;
- Technical specifications found on the internet.

**Evaluation Results on the ENSATE Context**

**Initial pre-selection results**

After having analysed all these resources and considering the fact that the product should be open source (school requirement), we ended up with the initial LMS's products of the next figure which are considered as the best known open source DLE's.

![Initial LMS's Open Source Products](image)

Figure 1: Initial LMS's Open Source Products

In this ENSATE evaluation process we decided to limit the evaluation of the proposed approach to only two products: Dokeos (version 2.1.0) and Moodle (Version 2.1.2+). This choice has been based on next criteria's:

1. Technical specification: We checked generally whether the main business and functional requirements are found in the technical documentation;
2. The amount and the clarity of the documentation offered by the provider or the product community;
3. The well-known products based on the amount of organizations and users using these products;
4. An intuitive judgment about the usability of the user interface based on the screenshots and demo's found on the internet;
5. The recommendation of the people with previous experiences with selected software.
A Formal Selection Approach for Digital Learning Environment Implementation

Functional evaluation results

<table>
<thead>
<tr>
<th></th>
<th>Functional score</th>
<th>Normalized score</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLE 1: Dokeos</td>
<td>37.24717554</td>
<td>7.449435107</td>
</tr>
<tr>
<td>DLE 2: Moodle</td>
<td>33.88276513</td>
<td>6.776553026</td>
</tr>
</tbody>
</table>

Table 6: Dokeos and Moodle Functional Results

Non-functional evaluation results

Extendibility Results/Plugins and modules:

<table>
<thead>
<tr>
<th></th>
<th>Extendibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLE 1: Dokeos</td>
<td>7</td>
</tr>
<tr>
<td>DLE 2: Moodle</td>
<td>6.23</td>
</tr>
</tbody>
</table>

Table 7: Non-Functional Results of Dokeos and Moodle

Maintainability/complexity evaluation results:

Logical metrics (Cyclomatic complexity):

Moodle (Understand results)

<table>
<thead>
<tr>
<th></th>
<th>AvgCyclomatic</th>
<th>MaxCyclomatic</th>
<th>MaxNesting</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory Structure</td>
<td>2.9</td>
<td>808</td>
<td>157</td>
<td></td>
</tr>
<tr>
<td>Grade</td>
<td>9</td>
<td>3</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 8: Moodle Complexity Metrics Results
Dokeos

<table>
<thead>
<tr>
<th></th>
<th>Avg Cyclomatic</th>
<th>Max Cyclomatic</th>
<th>Max Nesting</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory Structure</td>
<td>3.27</td>
<td>282</td>
<td>116</td>
<td></td>
</tr>
<tr>
<td>Grade</td>
<td>9</td>
<td>8</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 9: Dokeos Complexity Metrics Results
Formatting metrics results:

**Moodle, Final formative metrics grade**

<table>
<thead>
<tr>
<th>Source code architecture</th>
<th>Source code comments</th>
<th>Readability</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

7.66

Table 10: Moodle Formative Metrics Grade Results

**Dokeos**

<table>
<thead>
<tr>
<th>Source code architecture</th>
<th>Source code comments</th>
<th>Readability of the source code</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

7/10

Table 11: Dokeos Formative Metrics Grade Results

Maintainability and complexity scores comparison:

<table>
<thead>
<tr>
<th></th>
<th>Formative metrics</th>
<th>Logical metrics</th>
<th>Maintenance/complexity score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dokeos</td>
<td>7</td>
<td>8</td>
<td>7.5</td>
</tr>
<tr>
<td>Moodle</td>
<td>7.66</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 12: Moodle and Dokeos Maintainability and Complexity Scores

**The Overall Final Grade**

This next table summaries all previous calculated grades in order to get an overall comparison
between Dokeos and Moodle.

Due to time constraints, we were not able to assess the costs and security aspects in depth. Therefore we chose to give both DLE’s same score 7, in order to keep the final score formula and its weight valid.

<table>
<thead>
<tr>
<th></th>
<th>Functional</th>
<th>Usability</th>
<th>Maintainability</th>
<th>Extensibility</th>
<th>Support</th>
<th>Costs</th>
<th>Security</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dokeos</td>
<td>7.449435107</td>
<td>8.196</td>
<td>7.5</td>
<td>7</td>
<td>8.5</td>
<td>7</td>
<td>7</td>
<td>7.5</td>
</tr>
<tr>
<td>Moodle</td>
<td>6.776553026</td>
<td>7.44</td>
<td>7</td>
<td>6.23</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 13: Moodle and Dokeos Overall Final Grade

Results Discussion

Based on all earlier performed evaluations and obtained results, we recommended the ENSATE School to implement Dokeos 2.1.2 and integrate it in its information system in order to support the learning processes, particularly in improving the online distance communication between students and teachers.

- Dokeos fulfils up to 72% of the functional requirements of the ENSATE School.
- Furthermore it scores better than Moodle on the usability aspects.
- Dokeos can be considered as maintainable software and can be well integrated with the school information system, especially with Active directory.

Conclusion and Perspectives

However we would like conclude this document by emphasizing that the most of these results are context based results and are strongly attached to the ENSATE school context. That means that results may change when applied on other context, specially the functional and the maintainability results which are more or less subjective as the functional requirements and the sense of complexity varies from context to other and from person to other.

A further application of the method on other case study is necessary in order to assess the validity of the approaches in general, and also the validity of the used scales and techniques for obtaining criteria scores.
Acknowledgments

Our acknowledgments to the students team of 3rd and 2nd Computer Sciences Engineering Cycle of ENSATE which has participated in the evaluation and the implementation of the proposed approach of this research work.

Bibliography

Christian Litke and Michael Pelletier, 2002, Build it or Buy it? The Febricator.


Bandor, M. S., 2006, Quantitative Methods for Software Selection and Evaluation.


