Krishnamurthy, A. and O'Connor, R. V. Using ISO/IEC 12207 to Analyze Open Source Software Development Processes: An E-Learning Case Study Proceedings 13th International Conference on Software Process Improvement and Capability dEtermination (SPICE 2013), CCIS Vol. 349, Springer-Verlag, May 2013.

Using ISO/IEC 12207 to Analyze Open Source Software Development Processes: An E-Learning Case Study

Aarthy Krishnamurthy[1], Rory V. O'Connor[2]

[1] School of Electronic Engineering, Dublin City University, Ireland
[2] School of Computing, Dublin City University, Ireland
aarthy.krishnamurthy2@mail.dcu.ie, roconnor@computing.dcu.ie

Abstract. To date, there is no comprehensive study of open source software development process (OSSDP) carried out for open source (OS) e-learning systems. This paper presents the work which objectively analyzes the open source software development (OSSD) practices carried out by e-learning systems development communities and their results are represented using DEMO models. These results are compared using ISO/IEC 12207:2008. The comparison of DEMO models with ISO/IEC 12207 is a useful contribution; as it provides deeper understanding to-wards the OS e-learning system development.

Keywords: Software Development Process, Open Source Software, DEMO Models, Activity Flow Diagrams, E-Learning Systems, ISO/IEC 12207:2008.

1 Introduction and Research Approach

The e-learning systems developed as a Closed Source Software (CSS) follow either a traditional software development process (SDP) or a tailored version to suite the local needs and demands. These development processes have associated standards/guidelines that are followed, which mostly results in good quality software products. However on the other hand, OSS e-learning systems are developed by a community of like-minded developers, who are geographically distributed, yet work together closely on a specific software product [1].

OSSD has gained significant attention in recent years and is widely accepted as reliable products (e.g. Moodle, Apache, Linux, etc.). However, they lack a defined SDP which hinders the delivery of high quality systems to its users. Hence it is imperative to analyze and understand the existing and successfully running OS e-learning systems before developing a generalized OSS process for e-learning systems.

To the best knowledge of authors, there has been no comprehensive study performed on OS e-learning system development activities nor it has been modeled. Hence, the aim of this paper is to objectively analyze the OSSD of three most popular e-learning systems - Moodle, ILIAS and Dokeos. Most importantly, this paper discusses the result of the analysis (represented using DEMO Models) in conjunction with ISO/IEC 12207:2008 standard. This is a crucial work towards developing a

generalized OSSDP as using ISO/IEC 12207:2008 is the only way to get a deeper insight of the current OSSD practices.

The research approach is basically divided into two distinct parts. The first part deals with collecting the information about the development practices of Moodle, ILIAS and Dokeos and modeling the results using activity flow diagrams and DEMO models. These are briefly explained in this paper to give an initial understanding of how these results are used in conjunction with ISO/IEC 12207:2008 standard. The second part of the research approach focuses on how these results is used with ISO/IEC 12207:2008. This leads to the detailed understanding of various development activities carried out in all the three OS e-learning systems. These are explained in detail under section 3.

The paper is organized into 4 sections. Section 1 introduced the research background and the objective of this research along with the research approach used. Section 2 describes briefly the DEMO models and its results. Section 3 discusses the important aspect of this paper – ISO/IEC 12207:2008 and its mapping with DEMO results. Finally Section 4 presents the conclusion and future work.

2 Activity Flow Representation and DEMO Models

The initial task under the first part of this research work is towards discovering the current development practices on all the three OS e-learning systems [2]. The findings of the background study were represented as activity flow diagram for Moodle, ILIAS and Dokeos [3]. Each of the three OS e-learning system has executed different activities at different stages of development. Notably, the manner in which each stage is carried out depends entirely on the expertise, experience and availability of resources and skills. Further, the initial background study helped in identifying the various implicit and explicit stages of development. There were distinct similarities and differences between Moodle, ILIAS and Dokeos on different aspects. These are summarized in Table 1. Please note that the activity flow diagrams are introduced here because the results of this background work are an input towards the analysis of OSSDP.

Table. 1. Comparison results based on the background study.

	Moodle	**ILIAS**	**Dokeos**
No of development stages	Not explicitly categorized	Six explicit stages	Not explicitly categorized
Who validates proposed idea	Anyone can validate	Only the core team validates	No validation
Development plan	No plan is produced	No plan is produced	No plan is produced
Person(s) responsible for development	Initial volunteer & subsequent team formation	Initial volunteer & subsequent team formation	Any interested volunteers
Testing	Anyone can test at any time.	Anyone can test at anytime	Anyone can test until release
Release	Two stage release	Two stage release	One stage with no

	process is followed	process is followed	beta release

There have been few works carried out for modeling OSSD process. The model proposed by Jensen and Scacchi [4] for discovering the process followed for OSS development doesn't provide complete clarification for investigating the results obtained which inhibits its use for generalizing the OSSD process. Another model Basili and Lonchamp uses a multi-level approach for modeling the OSSD process [5] However, it does not provide precise notations for specifying the relationship between the product and the role. In addition, both the modeling techniques are depended upon the implementation method. Hence, DEMO methodology was considered in this research work as it overcomes the drawbacks of activity flow diagrams and also is independent of the implementation method.

DEMO (Design and Engineering Methodology for Organizations) is a methodology used for developing high-level and abstract models of construction and operation of organizations. DEMO applies enterprise ontology theory and 'Ontology' can be simply defined as an 'explicit specification of a conceptualization' [6]. DEMO models focuses on the communication pattern between human actors and various outputs produced during software development [7]. In this case we can use DEMO models to provide a high level overview of how the OS e-learning software products are developed without taking into consideration the technology or technique used for the development. The DEMO methodology and models has been already applied to OS systems and has been proved to provide a high quality, abstract model [7].

DEMO specifies various axioms, two of which are used in this wok. The first is the production axiom and according to this axiom, social individuals/actors fulfill the goals of an enterprise by performing 'acts'. The result of successfully performing an act is recorded in a 'fact'. On the ontological level, two kinds of acts occur: production acts (P-acts) and coordination acts (C-acts). Performing a P-act correspond to the delivery of products, services and information to the environment of an organization. By performing a P-act, a new production fact (P-fact) is brought into existence. In order to complete the performance of a P-act, social individuals /actors have to communicate, negotiate and commit themselves. These activities are called coordination acts (C-acts), and they result in coordination facts (C-facts).

The second axiom is the transaction axiom and it states that the coordination involved to successfully complete a P-act can be expressed in a universal pattern, which is called a 'Transaction'. A transaction consists of three phases: order phase, execution phase and result phase. In the order phase, the actors negotiate about the P-fact that is the subject of the transaction. Once an agreement is reached, the P-fact is produced in the execution phase. In the result phase, the actors can negotiate and discuss about the result of the transaction. These phases are subdivided into process steps, which consist of four coordination acts and one production act. C-act includes request, promise, state and accept. While the production act includes execute (process step). In DEMO, exactly two actors are associated with a transaction: an initiator and an executor. The authority over the execution of a single transaction is assigned to the executor [6]. This authority can be attributed to individuals or groups of individuals.

There are several ways (i.e., numerous diagrammatic representations) for modeling a development process using DEMO methodology. They include*: State model, Action model, Interstriction model, Process structure diagram (PSD) and Actor transaction*

diagram (ATD). The ATD shows the various actors' involvement in specific communication for executing a task and which actor actually produces the P-fact. This is a major advantage over the activity flow representation. In addition, ATD provides an overview of the actors and transactions within the scope of the enterprise/project and therefore aggregates the information contained within the PSD. In this paper we present the ATD for all three OS e-learning systems along with various outputs produced during the software development.

DEMO Model (ATD) for Moodle: The ATD for Moodle development is shown in Fig. 1, wherein the information of each of the PSD is aggregated. The actors involved in developing Moodle include; the Moodle community, core team/owner, developer, triage, integration reviewer, tester and a maintainer. Notably, Moodle carries out 11 transactions in total, from inception to release. These are denoted by 'T0x', where 'x' ranges from 1 to maximum number of transactions. In addition, Fig. 1 demonstrates two important points: Firstly, it shows which actor starts communicating with the other for executing a particular task. Secondly, it shows which actor actually executes the task to produce corresponding output (P-fact). For instance, 'Community' starts communicating with the 'Core team' for performing a transaction 'T01'. It is the 'Core Team's' responsibility to carry out the task and is denoted by a '■' at the end of the line.

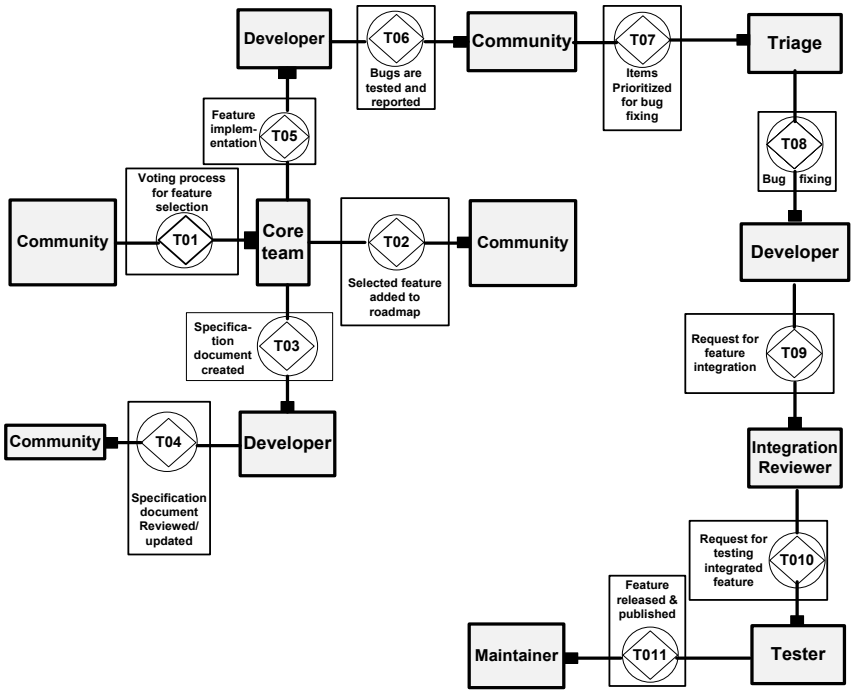

Fig. 1 ATD for Moodle Development

In Moodle, there are 4 transactions to be executed in order to select a feature and develop requirement specification for the selected feature(s). They are T01, T02, T03 and T04. The roles that execute the tasks corresponding to these transactions are the

Moodle community, owner/core team and the developer. P-fact is produced on successful execution of T01which implies successful completion of voting process for selecting the feature. Once the voting is done, the features with highest number of votes are selected (immediate requirement) and are added to the roadmap list. Therefore, the P-fact of T02 is the roadmap developed for feature implementation. In Moodle, specification document are to be created for each of the feature added to the roadmap. Hence the corresponding P-fact produced by executing T03 is the specification document. Finally, the P-fact for the transaction T04 is the suggestions and discussion on the specification document that the entire community provides, based on the specification released earlier.

The next stage in Moodle development is the implementation of the selected Moodle feature. Two transactions were executed for implementing and verifying the implementation of the Moodle feature (T05 & T06). The owner/core team starts communicating with the developer by placing a request 'T05 rq' for developing a particular feature. The developer promises to do the work which is indicated as 'T05 pm' and executes the task denoted by 'T05 ex'. The developer then requests the community to verify his work before merging the code 'T06 rq'. The community promises to verify the code 'T06 pm', verifies it and changes its status as verified 'T06 st'. Further, it sends the feedback to the developer who in turn acknowledges the work, 'T06 ac'. It then changes the status 'T05 st' and sends the code to the owner/core team. They in turn acknowledge the developer 'T05 ac'. The P-fact of transaction, T05 implies the successful implementation of the Moodle feature. P-fact of T06 is the completion of initial testing and bugs found in this testing are then reported for a fix.

Once the implementation was successfully finished, the feature is then tested and released to the Moodle-using community - Transactions T07 through T011 (for testing and releasing the Moodle feature developed). The P-fact of T07 is the prioritized list of items developed by the triage for fixing & testing. These are then sent to the developer. The developer then fixes the issue and tests it. The bugs that are fixed form the P-fact of T08 and are then added to the integration queue. The integration reviewers are responsible for integrating the same - the P-fact of T09. In transaction T010, the integrated code is tested and verified. The corresponding P-fact is the updated tracker item. The P-fact of the final transaction T011 is latest version of the software, which would be freely available for download from production repository. The P-facts produced during Moodle development are summarized in Table 2.

Table. 2. P-Facts produced during Moodle development.

Transaction	P-facts
T01	Voting process is completed.
T02	Development road map is created.
T03	Specification document created.
T04	Selected features are discussed.
T05	Feature is developed.
T06	Feature is tested by the community & bugs reported.
T07	Reported bugs are prioritized.
T08	Bugs are fixed.

T09	Features are added to the integration queue.
T010	Features are integrated and tested.
T011	A stable feature is released.

DEMO Model (ATD) for ILIAS: Various actors' involved in ILIAS development are: the user community, core team, developer, tester and maintainer. The transactions carried out for its development are denoted from T01 through T09 and the ATD for ILIAS is shown in Fig. 2. For selecting a feature in ILIAS, the user community and the core team communicate with each other and subsequently, the core team executes the transaction T01. The P-fact produced for this transaction is a feature wiki page which includes the selection decision along with the discussions that led to the final decision. The next step in ILIAS development is the development of requirement specification. Various actor's involved in developing and verifying the requirement specifications are: core team, user community and the developer. There are three transactions involved in developing the specification (T02, T03 and T04). The P-facts produced for each transaction (T02, T03 & T04) are the creation of requirement specification document, discussions on the specification document. Subsequently, the core team improves the specification doc by implementing some of the suggestions.

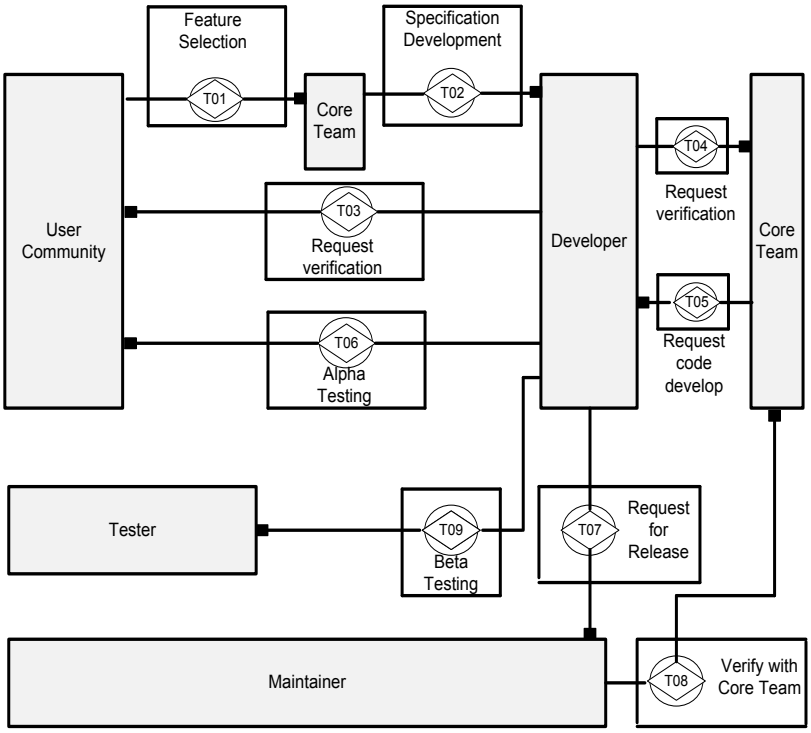

Fig. 2 ATD for ILIAS Development

The next step is feature implementation and this involves 3 main actors: the core team, the developer and the user community over 2 transactions T05 and T06. The P-fact produced by successful execution of T05 is the successful implementation of the

feature selected. The P-fact of T06 is the bug reported on that feature in their bug reporting system. Once the feature is developed, it has to be tested and released and the actors involved in this are developer, maintainer, core team and tester. There are three transactions T07, T08 & T09 executed by these roles. The P-facts achieved by the transactions are released working feature, updated roadmap with the released feature included in it and the bugs reported after the release in the bug tracking system. The P-facts have been summarized in Table 3.

Table 3. P-Facts produced during ILIAS development.

Transaction	P-facts
T01	Feature wiki with selected features is created.
T02	Specification document is developed.
T03	Specification document is discussed.
T04	Specification document is improved.
T05	Feature is developed.
T06	Feature is tested and bugs are reported.
T07	Accepted feature is released.
T08	Release road map is developed.
T09	Tested the released feature and bugs are reported to bug tracking system.

DEMO Model (ATD) for Dokeos: The ATD for Dokeos development is shown in Fig. 3. The actors involved in Dokeos development are user community, core team and the Dokeos Company. In all, 7 transactions are executed in developing a feature successfully for Dokeos (T01 through T07).

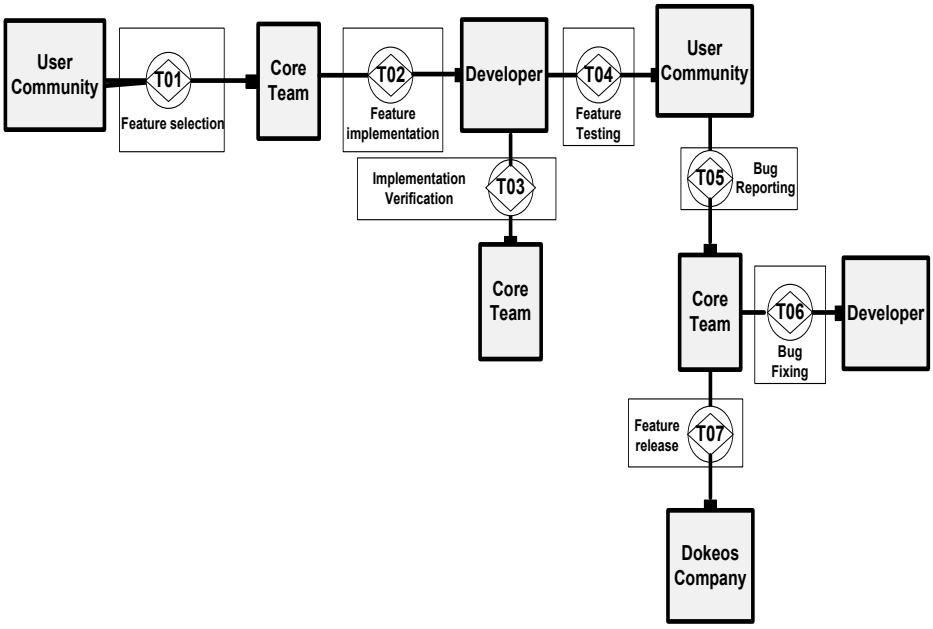

Fig. 3. ATD representation for Dokeos

Dokeos features are selected by the core team from the dream map (user community requests are polled in dream map) to road map. This is done in a single transaction T01. The transaction is initiated by the user community by adding the feature's request to the dream map. The core team would then select the feature and add it to the roadmap - the P-fact of the transaction T01. Once, the feature is selected by the core team for development, the developers are requested to build the feature which is depicted by transaction T02. The P-fact for T02 is the developed feature itself. Once the feature is developed, the developer requests the core team (T03) to verify and fix anomalies, if any. The P-fact of T03 is the verified and fixed feature. For testing and fixing the bug, the developer, core team and the user community communicate with each other. The developer requests the user community to carry out testing on the newly developed feature (T04). Once the user finishes testing, the bug fixes are reported to the core team which is the P-fact of T04. The core team in turn verifies, categorizes and organizes all the reported bugs. This list of verified, categorized and organized bugs is the P-fact of T05. These are then forwarded to the corresponding developer to fix the issues (T06). The fixed and working feature becomes the P-fact of T06. The next step is releasing the feature and the core team initiates the release process by requesting the Dokeos Company with a request. Then the feature is released by the Dokeos Company which is executed in transaction T07. The P-facts produced during Dokeos development are summarized in Table 4.

Table 4. P-Facts produced during Dokeos development.

Transaction	P-facts
T01	Feature is selected for development.
T02	Feature is implemented.
T03	Implemented feature is verified.
T04	Feature is tested and bugs are reported.
T05	Bugs are prioritized.
T06	Bugs are fixed.
T07	Feature is released.

2.1 Comparison

The previous sections in this paper provide sufficient details with regard to the development practices followed by the three OS e-learning systems. The activity flow diagrams provided information about the implicit/explicit software development stages and also helped in classifying the same. On the other hand, DEMO models provided information about what outcomes have been produced in each of the development stages (by executing a particular transaction) and by whom was that transaction executed. Table 5 presents various transactions executed for different basic development stages identified from the background study. For each of the three OS e-learning system development, if a particular development stage was identified as being executed then a tick mark '☑' is placed in the corresponding cell in Table 5; otherwise a cross mark '☒' is placed. Also, the transaction executed under a particular development which produces a successful outcome is mentioned inside the parentheses '[ ]'. However at this stage it is not clear that, to what extent each of the

OS e-learning systems had carried out each of the activities corresponding to various development stages. Therefore, there is a need for ISO/IEC 12207:2008 which helps in getting a deeper insight into the development processes of these OS e-learning systems.

Table 5. Summary of the research findings

Development stages	Moodle	ILIAS	Dokeos
Inception	☑ [T01, T02]	☑ [T01]	☑ [T01]
Planning	☒	☒	☒
Requirement Analysis	☑ [T03, T04]	☑ [T02, T03, T04]	☒
Design	☑ [T03, T04]	☑ [T02, T03, T04]	
Implementation	☑ [T05, T06]	☑ [T05, T06]	☑ [T02, T03]
Testing	☑ [T07, T08]	☑ [T08, T09]	☑ [T04, T05, T06]
Release & maintenance	☑ [T09, T010, T011]	☑ [T07]	☑ [T07]

3 ISO/IEC 12207:2008 Mapping

ISO/IEC 12207:2008 standard is a fully integrated suite of system and software life cycle processes which explains seven process groups, forty three processes, hundred and twenty one activities and four hundred and six tasks. Each of the processes within those process groups is described in terms of its (a) scope, (b) purpose, (c) desired outcomes, (d) list of activities and tasks which need to be performed in order to achieve the outcomes. The Software implementation processes are divided into six lower level processes with 29 outcomes that can be achieved by successfully carrying out the software implementation process and its corresponding activities and tasks [8]. Table 6 lists all possible outcomes that can be expected when these lower level processes are completed successfully.

Table 6. ISO/IEC 12207:2008 Process Groups

Lower Level Process		Possible Outcomes
Requirement Analysis Process	RA1	Requirements of software element & interfaces are defined
	RA2	Requirements analyzed for correctness & testability
	RA3	Understand the impact of the requirement on environment
	RA4	Consistency & traceability between system requirement are drawn
	RA5	Software requirement for implementation are defined
	RA6	Software requirements are approved and updated
	RA7	Changes to the requirement are evaluated
	RA8	Requirements are base-lined & communicated to all parties
Architectural Design Process	AD1	Software architecture is designed and base-lined
	AD2	Internal & external interfaces of each s/w item are defined
	AD3	Consistency & traceability is established
Detailed Design Process	DD1	Detailed design of each software component is defined
	DD2	External interfaces are defined
	DD3	Consistency and traceability are established between architectural

		design, requirement & detailed design
Construction Process	CP1	Verification criteria defined against requirements
	CP2	Software units defined by design are produced.
	CP3	Consistency & traceability are established
	CP4	Verification against requirement and design is accomplished
Integration Process	IP1	Integration strategy is developed
	IP2	Verification criteria for s/w items are developed
	IP3	Software items are verified using defined criteria
	IP4	Software item defined by integration strategy are produced.
	IP5	Results of integration testing are recorded.
	IP6	Consistency & traceability are established
	IP7	Regression strategy is developed and applied when change occurs
Qualification & Testing Process	QT1	Criteria for the integrated software are developed that demonstrates compliance with the software requirements
	QT2	Integrated software is verified using the defined criteria
	QT3	Test results are recorded.
	QT4	A regression strategy is developed and applied

The major advantage of using ISO/IEC 12207:2008 standard is that the outcomes mentioned by the standards can be compared directly with the P-Facts that were identified from the DEMO models. The comparative details are presented in Table 6. For each outcome mentioned by the standard, the corresponding transaction for Moodle, ILIAS and Dokeos have been mapped. Further, any particular outcome stated in the standard that is not met by the OS development community is denoted with an '-'. Notably, in case of RA8, all three OS e-learning systems produce data logical information (marked with '*') whereas outcomes of other transactions correspond to ontological information.

Table 7. Comparison with ISO/IEC 12207:2008 Process Groups

Outcomes	**Moodle**	**ILIAS**	**Dokeos**
RA1	T02	T02	-
RA2	T01	T03	-
RA3	T01	T01	T01
RA4	-	-	-
RA5	-	-	-
RA6	T01 & T02	T04	T01
RA7	-	-	-
RA8	Road maps*	Feature wiki*	Road maps*
AD1	-	-	-
AD2	-	-	-
AD3	-	-	-
DD1	T03	T02	-
DD2	-	-	-
DD3	T04	T03	-
CP1	T04	-	-
CP2	T05	T05	T02
CP3	T06	T06	T03
CP4	T06, T07 & T08	T06	T03
IP1	T09	-	-

IP2	-	-	-
IP3	T09	T07	T04
IP4	T010, T011	T07	T07
IP5	T010	-	T05, T06
IP6	-	T08	-
IP7	-	-	-
QT1	-	-	-
QT2	T010	T09	-
QT3	T010	T09	-
QT4	-	-	-

It can be observed from Table 7 that Moodle meets 16 out of 29 outcomes mentioned by the standard by executing 11 transactions. On the other hand, ILIAS meets 14 out of 29 outcomes by executing 9 transactions while Dokeos meets only 8 out of 29 outcomes by executing 7 transactions. Even though Moodle and ILIAS has achieved higher number of outcomes as compared to Dokeos, all three OS e-learning systems still have a huge scope for improvement in different stages of development.

4 Conclusions

The mapping of the process outcomes and the DEMO models results have identified that none of the OS e-learning systems have achieved all the outcome described by the process. However, it is important to know the extent to which each of the OS e-learning systems have performed. This is done by calculating the percentage of achievement for each of the development stages for all the three e-learning systems.

Table 8 Percentage of achievement by Moodle, ILIAS and Dokeos

	Moodle	**ILIAS**	**Dokeos**
Requirement analysis	50%	50%	25%
Architectural design	0%	0%	0%
Detailed design	66%	66%	0%
Construction	100%	75%	75%
Integration	57%	42%	42%
Qualification & testing	50%	50%	0%
Overall	53%	47%	23%

The percentage of achievement here is defined as the ratio between the number of outcomes achieved and the number of outcomes listed in the standard. For instance, in case of requirement analysis, the standard had prescribed eight outcomes as desired outcome of which Moodle satisfied four. Therefore, the achievement for Moodle under RA is 50%. Table 8 shows the percentage of achievement for each of the six stages for all three OS e-learning systems, along with the overall achievement ratio. This table also shows the weakness in the different development stages of all three OS e-learning systems. Moodle with 53% has the highest achievement rate. On the other hand, with an achievement rate of only 23%, Dokeos performs very poorly. Notably, all three OS e-learning systems have significant weakness in most of the development stages, except for construction stage. Without ISO/IEC 12207:2008, it would have not

been possible to identify the underlying weaknesses in each of the development stages for these three OS e-learning systems.

Having identified the whether the OS e-learning system had performed any activities pertaining to a development stage and the extent to which it has performed; it is possible to come up with a generalized OSSDP. Hence the next step would be come up with a strategy on selecting different stages of development, the frequency with which each stage could be performed, various important tasks and activities pertaining to each development stages.

Acknowledgments. Supported by Irish Research Council (IRC) - Embark Initiative Program.

References

1. Scacchi, S., Feller, J., Fitzgerald, B., Hissam, S. and Lakhani K., Understanding Free/Open Source software Development Process, Software Process Improvement and Practice, 11 (2),. pp. 95-105 (2006)
2. Krishnamurthy, A., O'Connor, R., McManis, J., Usability in Software Development Process for Open Source e/m-Learning Systems, Proceedings, 4th Irish Human Computer Interaction Conference (iHCI 2010), 2010.
3. Krishnamurthy, A and O'Connor, R.V.: Analysis of Software Development Processes of Open Source E-Learning Systems. In: McCaffery, F., O'Connor, R.V. and Messnarz R (eds.) EuroSPI 2013. CCIS, vol. 346. Springer, Heidelberg (2013).
4. Jensen, C., and Scacchi, W., Guiding the Discovery of Open Source Software Processes with a Reference Model, Third IFIP International Conference on Open Source Systems, Limerick, Ireland (2007)
5. Basili, V. R. and Lonchamp, J. 2005. Open source software development process modeling IN: Acuña, S. T. and Juristo, N. (eds.). Software Process Modelling, 10. US: Springer. pp. 29-64.
6. Gruber, T. 1994. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. IJHCS. 43(5/6): 907-928.
7. Huysmans, P., Ven, K. and Verelst, J. 2010. Using the DEMO methodology for modeling open source software development processes. Information and Software Technology. 52(2010) pp. 656–671.
8. Clarke, P., O'Connor, R.V.: The situational factors that affect the software development process: Towards a comprehensive reference framework. Journal of Information and Software Technology 54(5), 433–447 (2012)