Open Source Software Development Process for the Development of Open Source E-Learning Systems

Aarthy Krishnamurthy
M.Sc.

Thesis Submitted for the degree of
Master of Engineering

School of Electronic Engineering,
Dublin City University,
Dublin 9, Ireland.

Supervisors: Dr. Jennifer McManis and Dr. Rory V. O'Connor

August 2012

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Engineering is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work..

Signed: _____ (Candidate) ID No: **59117061**

(Aarthy Krishnamurthy)

Date: _____

Abstract

Over the last decade, numerous educational institutions and corporate world have employed various kinds of e-learning software solutions. One of the major components of end-to-end e-learning solution is the learning management system (LMS). These LMS are either developed as open source software (OSS) or close source software (CSS) product. In this regard, CSS for e-learning systems has a major drawback of being expensive and this hinders its widespread use. On the other hand, OSS is virtually free and not restricted by the licensing costs. The benefits of OSS can be completely realized only if there is an effective contribution from OSS community towards its development.

It is clear from the literature that the OSS development community does not follow an explicitly defined and documented software development process. This in turn results in lack of detailed information in the literature about the problems arising due to the absence of a defined process. Nevertheless, some of the major issues with regard to OSS development for LMS that have been identified include software design issues, week user Interface, lack of complete and accessible documentation, lack of co-ordination between unknown developers, etc.

This research develops a generalized OSSD process that could be used for the development of an open source (OS) e-learning system. To begin, in order to understand the current development practises of the existing OS e-learning systems, a detailed analysis was carried out for three different and popular OS e-learning systems (Moodle, ILIAS and Dokeos). The result of this analysis was represented as an Activity Flow Diagram which enabled precise identification of the implicit software development stages. In the next stage, in order to identify the output produced for each and every stage of development, a DEMO methodology was applied and DEMO models were built for three e-learning systems (Moodle, ILIAS and Dokeos). This is a particularly novel contribution that helps enable the development of the generalized OSSD process.

In order to select the different stages of development for the proposed process, the output resulting from each stage of the DEMO model was compared with the outputs prescribed by the ISO/IEC 12207:2008 standard. Further, in order to validate the

proposed process, an expert review method was employed by preparing a web-based questionnaire and circulating it along with the proposed process to three different and geographically separated OS experts. The proposed process was subsequently refined based on the feedback received from these experts. It is anticipated that the proposed OSSD process had the potential to streamline the future development of OS e-learning systems.

Acknowledgements

The period of my research had been a great learning experience, both on and off the campus. There have been days when it had been very enjoyable while there had also been very challenging times. However, I almost always found support from many different people around. It is very difficult to acknowledge everybody. But still, I would like to thank atleast few of them.

To begin with, I would like to thank my two supervisors, Dr. Jennifer McManis and Dr. Rory V. O'Connor for not only motivating and inspiring me regularly, but also providing all possible technical support during the entire duration of my research project. My research area was highly inter-disciplinary and required strong technical inputs from both e-learning and software engineering domain. In this regard, I have been extremely fortunate to have two supervisors from two different schools (School of Electronic Engineering and School of Computing respectively). They not only guided me in my research, but also ensured that I meet the technical requirements of both domains. Secondly, I would like to thank all open-source practitioners and experts who provided their valuable inputs to understand the basics of the Open Source Software Development (OSSD) practices. Further, I am grateful to the different technical experts who helped me in validating the proposed OSSD process by providing their valuable and timely feedback. At this stage, I would also like to thank IRCSET (Irish Research Council for Science Engineering and Technology) for supporting this inter-disciplinary and highly challenging research work.

Importantly, I would like to thank the DCU staff members, particularly, Dr. Gabriel Muntean, Dr. Olga Ormond, Ms. Breda McManus, Mr. Johny Hobson for helping me at different stages. A special word to all my colleagues in the University, especially, Aakash, Andrea, Bogdan, Irina, Lejla, Paul, Quang, Ramona, Sabine (and all others whom I have skipped here) for their regular and continuous interaction; and for pulling me to participate in different social activities which in turn helped me in broadening up my mind. Further, I would like to thank my family back home, especially my parents, my sister and my in-laws for their constant motivation and support. Lastly yet most importantly, I would like to thank my husband Hrishikesh whose love, kindness, patience, and understanding during the last few years transcend mere words.

Table of Contents

List of Figures

List of Tables

Acronyms

ATD – Actor Transaction Diagram

C-act – Coordination act

C-fact – Coordination fact

CS - Closed Source

CSS - Closed Source Software

CSSD - Closed Source Software Development

DEMO - Design and Engineering Methodology for Organization

GPL – General Public License

ICT - Information and Communication Technology

ILIAS – Integriertes Lern-, Informations- und Arbeitskooperations-System or Integrated Learning Information and co-operAtion System

ISO/IEC – International Standard Organisation / International Electrotechnical Commission

LMS - Learning Management System

Moodle – Modular Object-Oriented Dynamic Learning Environment

OS - Open Source

OSS - Open Source Software

OSSD - Open Source Software Development

P-act – Production act

P-fact – Production fact

PSD – Process Structure Diagram

RQ - Research Question

UP - Unified Process

1. Introduction

1.1 Research Background

Over the last decade, the rapid advancements in Internet and multimedia technologies have resulted in e-learning techniques moving from a marginal education mechanism to being an accepted form of education - across all primary education, secondary education, university education, etc (Allen and Seaman, 2008; Allen and Seaman, 2010; Allen and Seaman, 2011). This gives an opportunity for the learners and the teachers to opt for technology enhanced education which could be delivered virtually over a long distance, without having any time barrier. In addition, e-learning provides an excellent opportunity for both the learners and the teachers to quickly learn and teach new and relevant topics. This has resulted in a continuous increase in the demand for high-quality e-learning systems (Selim, 2007; Bernard, *et al.*, 2007).

In order to meet the demand, many e-learning systems have been developed over the last decade or so. While some of them are developed as commercial closed-source software (CSS) product(s), others are developed as an open-source software (OSS) product(s). Both types of systems co-exist in the current market though they follow different development and business principles. Notably, most of the OSS e-learning products are developed in an ad-hoc manner and the software products are distributed free of charge by networks of large volunteering group of computer programmers. On the other hand, the CSS products are developed for-profit and for commercial purposes by trained software professionals.

Over the years, OSS products have gained considerable popularity and recognition as compared to CSS products (Paulson, et al., 2004; Nakakoji, et al.,2002). However, there are still numerous and significant software development issues especially in the context OS e-learning system development. In an OSS development environment, the individual/group of people initiates a project to meet their immediate requirement (Krogh and

Hippel, 2006). More often than not, OSS community does not follow a well-defined/ well-documented software process (Scacchi, 2003; Glosiene, and Manzuch, 2004; Jensen, *et al.*, 2006), which raises development problem within the OSS community during product development. Due to the absence of an explicitly defined and documented software development process for OSS development, the drawbacks which arise due to the absence of a process are also not documented elaborately in the literature. However, few specific problems were identified and debated in detail within the OS e-learning development community (Boufford, 2004). The major problems that were identified include *software design issues*, *lack of complete and accessible documentation* (technical as well as user documentation), *not addressing all user requirements*, etc. Since, the OS systems and its features are mainly developed to address the developer's immediate requirement; it mainly results in less attention being paid to design issues. The poor design and requirement analysis in-turn leads to factors like, *misunderstood features*, *poor user interface*, etc. Also, due to the absence of a defined process, the co-ordination between unknown developers might be difficult and the new comers to OSS development might find it complicated to understand the development process, etc. All these issues significantly affects the OS e-learning system development and thereby the product quality itself. This in turn could make the users to prefer commercially developed proprietary software products which are much easier to work with.

Importantly, an OSS development (OSSD) process has several advantages (Jensen and Scacchi, 2008). Having a defined process prepares the community in developing the OS e-learning system for likely eventualities that might arise during development due to unforeseen circumstances. Further, it would assist a new comer to the OS community to clearly understand the development practices/activities and also the required deliverables from each of the tasks. This would indirectly help the OS community to gain an increased amount of valuable contributions from the new comers. However, the above benefits of OSS can be realized only if there is an effective contribution from the OSS community towards the OSS product development. Further, a defined OSSD process will facilitate the developer to understand the gaps in development

practices followed, thereby enabling the OSS community to efficiently contribute towards the product development. Once an OSSD process is defined, it will be much easier for the core team to manage the development. Also, it will enable the core team to predict and validate the development of software and easily productizing the end product (Scacchi, 2001). Hence, the fundamental premise for this research is to develop a generalised open source software development process for the OSS community.

1.2 Research Problem and Research Questions

Having identified the downsides of not using a defined process; and at the same time, the benefits of following a defined development process, the research problem and the research questions are presented in this section. The main goal of this research is:

> ***"To develop a generalised open source software development process (OSSD process) that can be used for implementation of an OS e/m-learning system in an OSS environment."***

However, the fundamental problem that needs to be addressed in order to achieve the above mentioned research goal is:

> ***"What approach should be followed in order to design a generalised OSSD Process?"***

The research problem is further divided into three fundamental research questions:

> ***RQ1: What are the current development practices followed by the OS e-learning product development communities?***
>
> ***RQ2: How should the current development practices be assessed in order to design a generalised OSSD Process?***
>
> ***RQ3: How is the OSSD process designed based on previous findings?***

RQ4: What approach should be followed to assess the proposed process and also to evaluate results of the appraisal?

Answering these questions will provide a platform for improving the OS e-learning system and its development; thereby addresses the shortcoming of not having a defined OSSD process.

1.3 Research Context

The research work carried out in this thesis is based on three popular OS e-learning systems, i.e., Moodle, ILIAS and Dokeos. All three e-learning systems are developed as free OSS products – specially an OS learning management system (LMS). The three systems are selected based on their high popularity and also the OSS community's commitment in developing the e-learning system.

Moodle: Moodle is an abbreviation of Modular Object-Oriented Dynamic Learning Environment. It is one of the early and successful OS e-learning platforms that had been developed following strong pedagogical principles. Its focus is to help the educators with creating the course content and delivering it to learners keeping the interaction and collaboration as one of the major criteria. Notably, Moodle has 58,207,428 users as of 2nd June 2012 (Moodle, 2012).

ILIAS: ILIAS stands for Integriertes Lern-, Informations- und Arbeitskooperations-System (in German) which was released as an OSS in 2002 (ILIAS, 2012). It was the first open source learning management system to follow SCROM 1.2 compliance completely. Also, unlike other OS e-learning system, ILIAS does not restrict learning to be confined to courses but offer a flexible environment for learning and working online with integrated tools. Interestingly, more than 2000 new users log onto ILIAS every month on an average that on an average every month with constant increase in number of users each year (Balogh and Budai, 2009).

Dokeos: Dokeos provides both commercial and OSS for e-learning purposes. Dokeos strictly follows SCORM principles and has one of the largest user-base. Although Dokeos have both the commercial and OS version of the product, even the free/OS version provides all the features that are need for blended learning management – from authoring to reporting. Dokeos has 42,45,929 users using the system for e-learning purposes (Dokeos, 2012).

1.4 Research Contribution

The main contribution of this research work is to propose and develop a *generalised OSSD process*, which could be used during the development of an OS e-learning system. The main benefit of the proposed OSSD process is that it would streamline the development of next generation OS e-learning systems.

To begin with, the current problems in OSSD, especially in OS e-learning system development were identified. Secondly, in order to understand the current development practises of the existing OS e-learning systems, a detailed analysis was carried out. The results of the analysis were then interpreted using *Activity flow diagrams* for three different and popular OS e-learning systems (Moodle, ILIAS and Dokeos). Thirdly, in order to identify the output produced for each and every stages of development, DEMO methodology was applied and *DEMO models* were built for all three e-learning systems.

Subsequently, the generalised OSSD process and its various development stages were then proposed based on the conjunctive results of Activity flow diagrams, DEMO models and ISO/IEC 12207:2008 standard. This is a particularly novel and significant contribution of this research work.

1.5 Structure of the Thesis

The thesis is organised into eight chapters. Chapter 1 introduces the *research background* and the *motivation* of the work before dwelling onto the *research problem* and the *research questions*. Notably, the *research contribution* and *the structure of the thesis* are also presented in chapter 1. In chapter 2, the research background is elaborated and the main topics, i.e., *e-learning systems*

and *software development process* are discussed in detail along with related works. Chapter 3 provides an overview of the research approach followed in this work along with corresponding detailed information of the same. Chapter 4 describes the review process carried out for the three OS e-learning systems, i.e., Moodle, ILIAS and Dokeos and the results of this study is modelled and presented as activity flow diagrams. The comparisons of the results are detailed along with the advantages and drawback of such modelling technique and ways to overcome the same. Chapter 5 describes the modelling of current practices using DEMO methodology and how it overcomes the drawbacks of activity flow diagram. Further, the DEMO models are evaluated with the help of software implementation processes as described in ISO/IEC 12207:2008. Chapter 6 then elucidates the process of designing the OSSD process in detail. Chapter 7 details the procedure followed for validating the proposed OSSD process along with the validation results. Finally, chapter 8 concludes the thesis with information on major research finding, the limitations and future work.

2. Literature Review

2.1 Introduction

This chapter begins with identifying two important topics related to this research work; *e-learning systems* and *software development process*. The two topics are then described in detail followed by the challenges in developing e-learning systems. Further, various types of e-learning system and their development principles are discussed.

E-learning systems and *software development processes* are independent topics on their own. At the same time, software development process plays an important role in the development of e-learning systems. This is being depicted in Fig 2.1 and is highlighted in green colour. The entire research work focuses on this green shaded area.

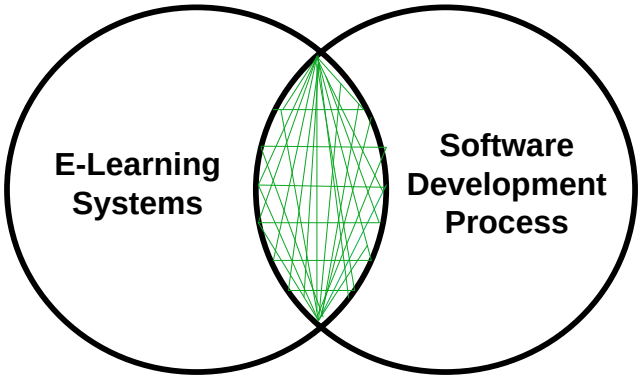

Fig. 2.1 Relationship between e-learning system and software development process

Software development process employed during the implementation influences the quality of the final product (Zahran, 1994; Clarke, P. and O'Connor, R. 2010). This is very crucial for an e-learning system as it directly affects the perceived satisfaction and usefulness of an e-learning system for both the educators and learners (Liaw, 2008). Therefore, it is important to make use of a suitable and appropriate development process during OS e-learning software implementation.

2.2 E-Learning Systems

E-learning can be defined as "technology-based learning in which learning materials are delivered electronically to remote learners via a computer network" (Stockley, 2003; Oguzor, 2011; Zhang, *et al.,* 2004; Yong, 2008). Further, e-learning systems are those software systems that supports e-learning such as, computer based learning, web based training/learning, virtual classroom, etc (Tavangarian *et al.*, 2004). These software systems can potentially remove barriers of space, time and location and importantly, provide knowledge in different media formats, anytime and anywhere (Shea, 2002; Milojevic, 2011). The usages of such systems also enables self-paced learning, provide consistent learning materials to its learners, allow the educators to easily and quickly update the learning materials and is usually less expensive to provide education as a whole. Also, it could potentially lead to an increased retention and a stronger grasp on the subject; while at the same time could be easily managed for large groups of students (Cantoni, *et al.,* 2004). Further, the advancement of computer and networking technologies provide highly diverse means to support learning in a more personalized, flexible, portable, and on-demand manner.

An effective e-learning system can be viewed as an integrated, end-to-end learning system comprising of three major components and is depicted in Fig. 2.2.

Component 1: The first component is the transmission network and seamless communication mechanism between the different electronic and handheld mobile devices and falls under the realm of wireless networks and Internet-supported solutions.

Component 2: The second component is the learning content. This includes course materials for different courses and modules, content authoring, etc. However, these aspects come under the category of content developers.

Component 3: The third component is the e/m-learning application software which is the learning management system (LMS). This is the bridge that links the first and second component in the end-to-end learning system. In other words, an effective LMS connects the different components of the integrated

learning system efficiently. Therefore, it is not only vital to develop this component efficiently but also imperative to have a structure/system that would facilitate seamless and flexible learning to the users, taking into account the diverse set of devices with different features and capabilities.

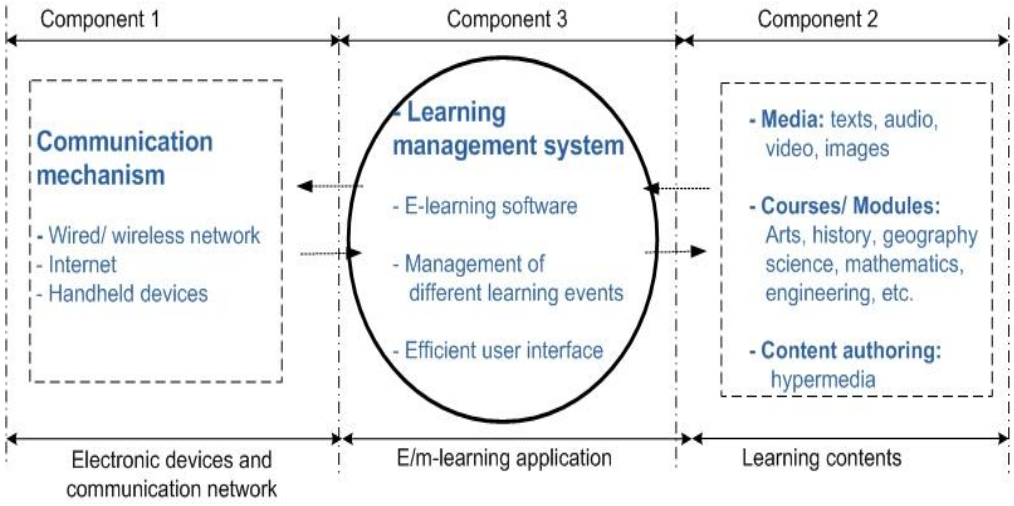

Fig. 2.2 Different components of end-to-end integrated e/m-learning system framework

These LMS's are not only an integral part of an e-learning system but also form a basis/platform to impart technology-based-education to the learners. There are several LMS's available and the most popular ones includes Moodle, Blackboard, Dokeos, Sakai, Blue apple, ILIAS, Adobe acrobat e-learning suite and several others. The LMS is briefly explained in sub-section 2.2.1. Notably, since LMS is an important component for an effective end-to-end e-learning system, these two terms would be used interchangeably and whenever e-learning is used, it in fact refers an LMS.

2.2.1 Learning Management System (LMS)

LMS is a software application for the administration, documentation, planning, delivering, tracking and managing the learning events within an organization, which include online/web based learning, virtual classroom, instructor-led courses, etc (Ellis, 2009). A robust LMS should be able to do the following six tasks efficiently and these include; centralize and automate administration, use self-service and self-guided services, assemble and deliver learning content rapidly, consolidate training initiatives on a scalable web-based platform,

support portability and standards, deliver online training and webinars and personalize content and enable knowledge reuse (Ellis, 2009).

It should be noted that, the selection of parameters (like mode of operation and usage, intended audience, etc.) and the system design will also depend on program goals, the infrastructure/facilities and importantly, the culture/background/diversity of the learners (Kruse, 2009). Based on these parameters, an LMS is selected for an institute/ organisation. LMS's are available either as a commercial CSS product or as an OSS product. Some of the popular CS e-learning systems are *Adobe e-learning suite*, *Blackboard*, *Blue apple*, etc. Likewise, popular examples of OS e-learning systems are *Moodle*, *Dokeos*, *.LRN*, *Sakai*, *ILIAS*, etc. However, not all LMS's have been successful. This is because, not all LMS are able to meet/satisfy the different challenges – educational challenges, technological challenges, sociological & cultural challenges, and psychological challenges (Cemal Nat, *et al.*, 2008). In this research however, only the technical challenges of LMS is focussed.

Importantly, in order to make an e-learning system successful it should satisfy the need of different types of users involved. An in-depth understanding of the benefits and drawbacks of LMS/e-learning systems from the perspective of each player - *a learner*, *instructor* and the *institution* is essential. These are summarised in Appendix A for further reading. The next section describes the software development process, its roles, goals and finally the broad classification of software development processes.

2.3 Software Development Process

A software development process is defined as "a coherent set of policies, organizational structures, technologies, procedures, artefacts and activities that are needed to conceive, develop, deploy and maintain software" (MingshuLi *et al.*, 2006; Fuggetta, 2000; Humphrey, 1988; Sommerville, 2004). Further, a software development process describes the internal relationships among different phases of development by expressing their order and frequency, as well as by defining the deliverables of the project. It also specifies criteria for

moving from one phase to another phase (Curtis, *et al.*, 1992). In addition, a software process also describes a series of actions or steps to be taken in order to achieve a particular goal (Fuggetta, 2000).

Any software development process includes various roles, goals and activities. In fact, there are four key roles and seven goals for any given software development process (Kruchten, 2000).

- The first and foremost role of a software process is to provide guidance in ordering and following various software development activities, as mentioned in Table 2.1.
- The second role is to clarify when and what are the different artefacts that are to be produced during and at the end of each activity.
- The third important role is to direct the tasks of the development team.
- Lastly, the software process should monitor and assess the project progress and henceforth its success.

These four roles are applied during the software development in order to achieve product goals like effectiveness, maintainability, predictability, repeatability, quality, improvement and tracking (Tyrrell, 2000). The process activities form a major aspect of software development processes. These activities form the basis to realize the process goals. The broad set of activities carried out during the development of a software product is represented in Table 2.1 (Sommerville, 2004).

Development Activities	Description
Inception	The software product is conceived and defined.
Planning	Initial schedule, resource and cost are determined.
Requirement Analysis	Defines what the software should do.
Design	Specifies the parts and how they fit.
Implementation	Software code is written.
Testing	Execute and test the application with test input data.
Maintenance	Repair defects and add capabilities/ functionalities.

Table 2.1 Software development activities

A software process model can be defined as a "simplified description of a software process that presents one view of a process and may also include activities that are part of software process and products, along with the constraints that apply to the process and roles of the people involved" (Sommerville, 2004). Further, a software development model differs from software development method; where the primary goal of a software development method is to "focus on how to navigate through each phase by determining data, control, or uses hierarchies; partitioning functions; allocating requirements and how to represent phase products such as structure charts; stimulus-response threads; state transition diagrams" (Boehm, B.W. 1988). Some of the popularly known and used software development process models and methods are *Waterfall model*, *Evolutionary development*, *Exploratory model*, *Component based development*, *Unified software development* and *Agile methodology*.

Notably, the software development processes can also be broadly classified into two categories – Closed Source Software Development (CSSD) process and Open Source Software Development (OSSD). The main difference between OSSD and CSSD are in their development principles (Devine, 2008; Raymond, 1998) which are presented in the following section.

2.3.1 Open Source and Closed Source Software Development

The development of software can also be broadly classified into OSSD and CSSD. CSSD can be defined as the one where, trained software professionals are employed in developing a software product (termed as CSS products). In many cases, these software professionals follow a defined and documented software development process. CSS products are developed for commercial purposes (for-profit) and only the executables are sold through sales team/person to the licensed customers. Also, the source code is not released to public and cannot be modified as most of the products would be protected under the copyright license or patents (Stephan Donovan, 1994; Tysver, 2008). Further, CSSD in general has a formalised organisation and structure. Some well known examples of CSS products are Microsoft Windows, Adobe Acrobat Suite, Oracle solutions, Blackboard, etc. On the other hand, OSSD is

oriented towards the joint development of a community of developers (Scacchi, 2001; Krogh, 2006). OSS products are developed by volunteers out of interest and any person could volunteer to play any role in its development, based on their skills and interest. Usually, the volunteers self assigns tasks that they would like to perform. Also, OSS is built as an open-source project initiated by an individual/group of people to meet their immediate requirement (Krogh, 2006). The people involved in OSS and its development share ideas, ideologies, technologies, source code and yet work independently in a geographically distributed environment and are spread across the world (Scacchi, *et al.,* 2006). They communicate through Internet forums, e-mails, informal chats or through any other communicative channels (Scacchi, 2007). Also, majority of the OSS does not have corporate owner or management staffs to organize, direct, monitor, and improve the software development practices that are followed for development (Scacchi, *et al.,* 2006). Some well known examples of OSS products are Linux, Firefox, Moodle, etc. The next section will specify the major similarities & differences between OSSD and CSSD.

2.4 Comparison between OSSD and CSSD Processes

There are several aspects in which the OSSD and CSSD process differ. These differences apply in the case of e-learning systems as well and are listed below (Open source initiative; Raymond, 1998; Ghosh *et al.,* 2002; Feller and Fitzgerald, 2002; Fuggetta, 2003; Ye and Kishida, 2003; Paulson, 2004).

Underlying principle: CSSD is purely for commercial purposes and focuses completely on business perspective and thereby, the financial growth. On the other hand, OSSD mainly aims at constantly providing software solutions and improving the software through open contributions from entire community of developers.

Availability of source code and software license: In any CSS developed using CSSD processes; only the executables/binaries are made available to the customers. The number of users is based on the number of licenses purchased by the customer. On the other hand, the source code of OSS is publicly

available. There is no need of any intermediate vendor for downloading the software. Further, OSS's are in general published under general public license (GPL) where anyone can download OSS products, make modifications and redistribute it under same GPL. Notably, there is no restriction on the number of user(s) licenses.

Structure of the organization: In case of CSSD, the team and their hierarchy are completely defined. Tasks are allocated to the team members and plans are drawn for the development. On the other hand, in case of OSSD, anyone interested in the proposed idea could join in and contribute based on their ability and interests. There are no strict hierarchy and the developer's self-assign tasks. Usually, the administrative executive has a weak control over the development. A rough plan would be drawn by the developer as a check point to check their output and to answer any queries that arises in the community.

People and location: In most of the cases, people working in CSSD know each other and may or may not be geographically distributed. On the other hand, in case of OSSD, unknown people work together from different part of the world on the same module.

Defined process: Many of the CSSD process follow a defined and documented software development process and most commonly it happens to be the conventional software development process or a customised form of the same. However, in an OSS environment, the software development process is not defined or documented. The OSS community follow their own development practices (ad-hoc practices) which reflect their expertise on software development.

Need for software: In CSSD, the products are developed, with main focus on the customer's requirement. Therefore, the development is user-oriented and the developers are paid for their efforts. However, in case of OSSD, most of the development is initiated due to the developer's personal requirement/need. It should be however noted that this difference is slowly fading out, especially in designing e-learning systems like Moodle, Sakai, ILIAS, etc.

Developer/Tester: In general, most of the popular OSS products are developed by more than hundreds of developers and testers. However, in CSSD, only

major software companies can afford to have a huge number of developers/testers for a single project.

S.No.	Attributes	OSSD	CSSD
1.	Formalised Organisation	No	Yes
2.	Defined structure	No	Yes
3.	Follow a defined and documented development process	No	Yes
4.	Most often the development happens in ad hoc fashion	Yes	No
5.	Source code made available to all its user	Yes	No
6.	Developed for commercial benefits and financial profits	No	Yes
7.	Wider space for testing	Yes	No
8.	Reliable and responsible 24X7 software support	No	Yes
9.	Up to date technical reports/documents	No	Yes
10.	Up to date user documents	No	Yes
11.	Very intuitive and outstanding software design	No	Yes
12.	Append many new feature to cope competition	No	Yes
13.	Burdened with license cost	No	Yes

Table 2.2 Comparison between OSSD and CSSD processes

The comparison between the OSSD and CSSD are tabulated and shown in Table 2.2. Also, this table enables pointing out the weaknesses of OSSD practices which is then subsequently addressed partially in this research work.

2.5 Summary

This chapter provided a brief definition of an e-learning system and discussed its various components, including a major component of an end-to-end e-learning system – the Learning Management System (LMS). This was followed by a definition of software development process, its activities, roles, goals along with a broad classification of software development processes. The underlying principles of the two main classifications of software development processes – OSSD and CSSD were then explained along with a comparison between the two. Since this research work focuses on OS e-learning system, a further analysis is subsequently carried out for OS e-learning systems.

3. Research Approach

3.1 Introduction

This chapter discusses the approach followed for carrying out this research work. The research approach is divided into two distinct stages as shown in Fig.3.1 and involves various tasks at each stage. The first stage is called the *foundation stage* (described inside a box with dotted line) while the second stage is called the *execution stage* (described within a box with regular lines). The two stages are described in detail in section 3.2 and section 3.3 respectively. Further, the methodology followed in proposing a generalised OSSD process – a major contribution of this research work is also described in section 3.3.

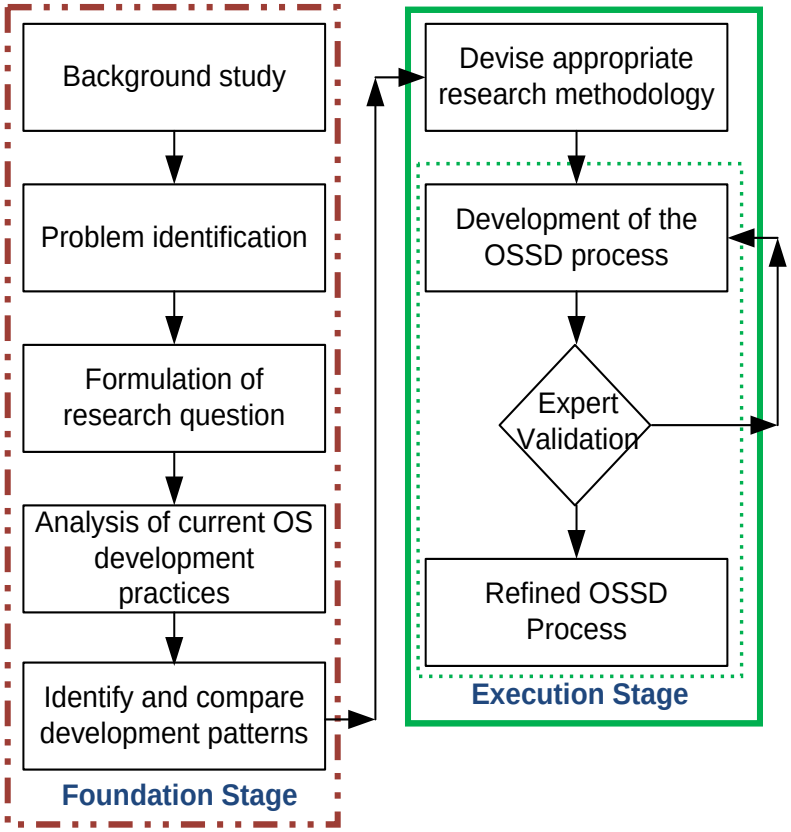

Fig. 3.1 Two stage research approach

3.2 Foundation Stage

The first task in the foundation stage was to carry out an initial background study of open source software, open source e-learning systems and open source software development. Based on this study, the research problems were identified which was followed by various research questions. The research problem and the precise research questions were described in detail in Chapter 1. Answering the different research questions led to the solution for the identified research problem(s); which are summarised in Chapter 8.

The first task towards the solution is to understand the current development practices; followed by the OS e-learning system development community. Unfortunately very little literature is available with respect to the OS e-learning system development. Only few of the OSS communities had updated their development practices in the form of *blogs*, *wiki pages* or as *guidelines to its members*. Also, this information was distributed randomly across their web-pages. This made it very difficult to capture all the required information. Hence, there was a need to do individual analysis on the development of each of the three OS e-learning systems.

For each of the three OS e-learning systems, an in-depth analysis was done in a manner that is comparable/ consistent with the case study. This approach was chosen as it is comparable with case studies and it answers questions like 'how' and 'why' (similar to case studies). This method is particularly beneficial for an OSSD environment where there is little/ no control over the events. The final task of the foundation stage was to evaluate the results obtained through analysis. The results are represented using the activity flow diagrams for easier and quicker understanding. Further, these results were compared and their similarities and differences were identified. This led to the identification of development patterns followed by these OSS development communities. A detailed explanation along with its advantages and disadvantages are described in chapter 4.

3.3 Execution Stage

The second stage of this research is called the execution stage. The first task under execution stage is to device an appropriate research approach to be followed that would answer the research questions. This research approach was developed and refined over number of iterations. It can be viewed as two sub stages towards the solution.

- Development of OSSD Process - This task forms the core of this research work and is the focus of this chapter. The tasks carried out are described in detail in section 3.3.1.
- Validation of OSSD Process - The second task is to validate the research outcomes using appropriate validation method. The validation methods selected for validating the proposed OSSD process is described in section 3.3.2.

3.3.1 Development of OSSD Process

In order to develop a generalised OSSD process, the best developmental practises from different OS e-learning systems needs to be incorporated. However, in-order to do that, it is essential to understand how it is being performed currently. As a part of foundation stage, analysis similar to case study was carried out and the development practices were identified. As a first step, the development practices are represented using an activity flow diagram. This representation was used as it would provide a dynamic aspect of an overall flow of the development practices followed by the OS communities. This type of representation is preferred for this research over the *state diagrams* or *event driven process diagrams*. This is because, what is required is not an abstract model or an exhaustive detail about various events carried out for each of the activities; but an overall flow of activities carried out within the community for its software development. In this regard, the activity flow representation of each OS e-learning system indicates different stages of its software development.

However, this representation was not independent of the technique followed during development and also does not identify the outcomes produced when these development activities are carried out. Further, the analysis done for activity flow diagram revealed that there are considerable variations in activities performed by various OS e-learning systems. Therefore, it is extremely complex to arrive at a generalized OSSD process for OS e-learning systems, based on the above analysis alone. Hence, there is a strong need for a level of abstraction in order to model the OSSD process (Lonchamp, 2005).

There have been couple of works carried out for modelling the OSSD process. However, each of them has its own limitations. For instance, the model proposed by Jensen and Scacchi (Jensen and Scacchi, 2008) for discovering the process followed for OSS development does not provide a complete clarification for investigating the results obtained. This inhibits its use for generalising the OSSD process. Likewise, the model developed by Basili and Lonchamp uses a multi-level approach (3 layer approach - definitional, general and specific) for modelling the OSSD process (Basili and Lonchamp, 2005). However, its main drawback is that it does not provide precise notations for specifying the relationship between the product and the role. In addition, both the models are dependent on the development activities carried out in modelling the process. Hence, an alternate approach - DEMO methodology - is considered in this research work. This model was selected because it could overcome the drawbacks of the activity flow representation and also, is independent of how the development activities were carried out. For deeper understanding, the DEMO methodology is further explained in section 3.4

Results of activity flow diagrams and DEMO models constructed for OS e-learning system development led to the identification of various implicit software development stages, activities carried out in each of its development stages and also the outcome of each such activity along with the actor who was responsible for producing an outcome. As mentioned before, various e-learning communities follow different approaches towards software development thereby differing in execution of various development stages and activities within each stage. Thereby, all of them produce a mix of various

other outcomes. Based on these finding, it is difficult to generalise the OSSD process for OS e-learning systems.

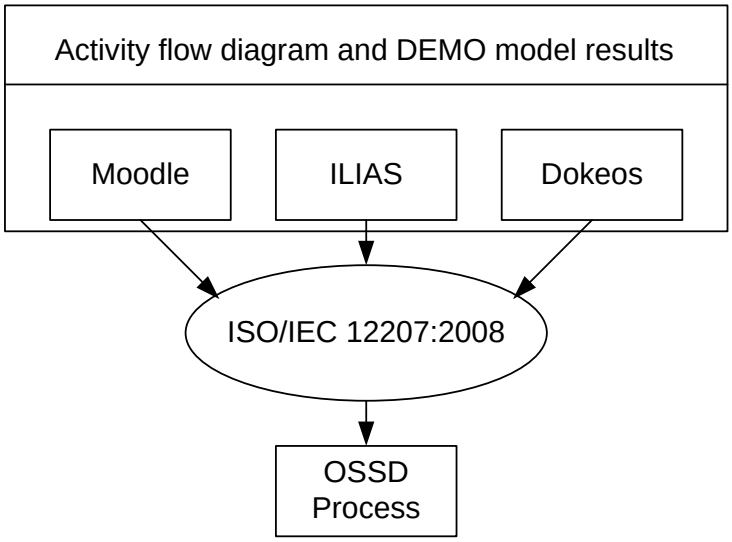

Fig. 3.2 Methodology for developing OSSD process

Therefore, there is need to use a well defined standard as a base tool in selecting different stages of development, ordering them and also to iterate as required. For this purpose, ISO/IEC 12207:2008 standard (ISO/IEC 12207:2008) was selected and was used as a foundation for proposing an OSSD process. This standard acts as a guide for both system life cycle processes and also software specific processes. The standard also lists various sub-processes along with its lower level processes. For each of these; the standard defines the *purpose*, *list of outcomes*, *activities* and *tasks*. This research work utilises the outcomes listed for software specific process and in particular the software implementation process as a base tool in selecting different software development stages. This is done by comparing the outcomes listed in the standard with the outcomes that are identified for each OS e-learning system development (activity flow diagrams and DEMO model results). In other words, these outcomes were selected in conjunction with the outcome achieved by the OS e-learning systems. The OSSD process is thereby generalised as shown in Fig. 3.2. These are explained in detail under chapter 5 and chapter 6.

3.3.2 Validation of the OSSD Process

Once the OSSD process is generalised, it has to be validated to see if it is feasible in the real-world OSS environment. This forms the second part of the execution stage, i.e., validation (Fig. 3.1). The best way to validate is to develop a feature following the recommendations provided in the proposed OSSD process and assess the proposed process as a post-mortem. Unfortunately, this method could not be followed due to time constraint and other practical problems. Another alternative approach for validation is to present the proposed OSSD process to the OS developers and ask them to follow the process recommendations during development. This was not practical as well, because in this case, we would not have had any control over the development activities and the way it might be adopted during development. Also, it would require considerable time in order to carry out a feasible longitudinal study. At this stage, it should be noted that the best person(s) to validate the OSSD process will be the OSS developers and/or its community member as they know exactly how it can be adapted to best suit the development of OS e-learning system; while also maintaining the integrity of the proposed OSSD process with OSS development. Therefore, another approach called 'expert review' method was selected for carrying out the validation process.

In the 'expert review' method, the experts (OSS developers or its community members) were provided with information about the proposed OSSD process and are asked to review it. Once reviewed, the experts were then provided with a web-based questionnaire and their feedback on the proposed process was then collected. Notably, the experts who could perform the validation were selected based on pre-set criteria's which are further discussed in chapter 7. This was considered to be a viable approach to validate the process since it did not require the experts to spend lot of time, thereby reducing the overall response time. Also, this approach was simple to implement and could be done at no monetary cost. An additional advantage of the 'expert review' method was that the analysis of the results was also easier, especially since the questionnaire had both objective and subjective questions. Moreover, the

questionnaire allowed sufficient space for the experts to provide their feedback/comments at all stages. The feedback obtained from the reviewers was then used to refine the OSSD process and is shown as iteration in Fig.3.1 under the execution stage.

3.4 DEMO Methodology

DEMO is abbreviated for **D**esign and **E**ngineering **M**ethodology for **O**rganisations and has its origin from organisational engineering domain. This methodology is used for developing high-level and abstract models of construction and operation of organizations. This methodology applies enterprise ontology theory and 'Ontology' can be simply defined as ''an explicit specification of a conceptualization" (Gruber, 1994). Enterprise ontology theory is described as the implementation independent essence of an enterprise (Dietz, 2006) and has a strong theoretical foundation. The strong theoretical foundation ensures that DEMO models can be claimed to be coherent, comprehensive, consistent, concise and essential (Albani and Dietz, 2011).

It is essential to understand briefly the enterprise ontology theory and importantly its terms in order to understand how DEMO methodology and its models can be used for modelling OS e-learning system development. Therefore, *Enterprise ontology theory and its axioms* are first explained along with the different terms used.

Enterprise Ontology Theory

The enterprise ontology theory consists of four axioms which form the basis for DEMO methodology and its models. They are *Distinction Axiom*, *Production Axiom*, *Transaction Axiom* and *Composition Axiom*. The distinction axiom differentiates between three human abilities which are required to fulfil certain actions - *datalogical actions*, *infological actions* and *ontological actions* (Stamper, 1973). Ontological actions are considered to be the fundamental human actions in a process flow. Since, the actions on the infological and datalogical level do not introduce new products/ services/

information, if is sufficient to focus on the ontological level actions in-order to describe its essence using DEMO.

The production axiom states that social individuals/ actors fulfil 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 transaction axiom 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 (Huysmans *et al.*, 2010). This authority can be attributed to individuals or groups of individuals.

Some processes may produce more than one P-fact for the organization. In that case, a DEMO transaction needs to be created for each P-fact. This requires coordination between transactions. The composition axiom describes how these transactions can interact. One aspect of interaction is how transactions are initiated. Any transaction is either initiated by an external party (e.g., a request for a bug fix by a user) or a time-based trigger (e.g., the nightly building of the software), or enclosed in another transaction. In the case of an

enclosed transaction, an information dependency usually exists between the enclosing and the enclosed transaction. The models created using the DEMO methodology for this research are based on these four axioms.

Further, DEMO methodology focuses on the communication pattern and various outputs produced during various software developments (Huysmans, *et al.,* 2010). From the context of this research, DEMO methodology gives 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. Yet, it identifies precisely who is responsible for producing an output. Also, the DEMO methodology has been already applied to OS systems and has been proved to provide a high quality, abstract model (Huysmans, *et al.,* 2010). Unlike other modelling methods used for modelling OSS development, DEMO exhibits two specific features within the context of OSSD process modelling that adds strength to this approach.

- DEMO analyses processes at the ontological level and provides high-level process descriptions, instead of focusing on the implementation level.
- DEMO studies the communication pattern between human actors, instead of the sequences in which activities are performed.

These characteristic of DEMO makes it extremely appropriate for modelling the development practices of software products and therefore OS e-learning systems by extension. The DEMO models and its application are explained in detail in chapter 5.

3.5 Summary

This chapter introduced the two-stage research approach and also explained each stage in detail. Further, the various tasks performed under each of these stages were explained that provided a comprehensive overview of this research work. A detailed and individual explanation of each tasks are provided in the remainder of this thesis under chapter 4, 5, 6 and 7.

4. Analysis of OS E-Learning System Development Practices

4.1 Introduction and background information

In this chapter, three OS e-learning systems, i.e., Moodle, Dokeos and ILIAS are analysed in detail. These three e-learning systems were selected mainly because of the following two factors:

- *Popularity*: All three e-learning systems selected are currently used and are quite popular among the institutions offering e-learning courses.
- *Development Activities*: The OSS communities constantly perform various development activities and have significant contributions towards its developmental.

The development activities of these three OS e-learning systems were identified using two different approaches. The first approach was to collect information from their websites, blogs, wiki pages and/or from any social network/media used by the community to broadcast the information. In addition to these, information was also collected from bug tracking system (or any other tracking systems), as some of the OSS communities track each of its development activities in such systems.

The second approach was applied only when the information collected from the first approach was either incomplete and/or ambiguous. This was in-fact a direct method whereby, questions were posted in public OSS community forums. The rationale was; anyone with the information can directly provide his/her answer(s) through the same forum or could even send e-mail or private messages. This helped in identifying many of its current development activities. However, the disadvantage of this approach was that, many-a-times, there were no clear consensus from the contribution of the community member's. In these scenarios, separate e-mails had to be sent to the core

members and other experienced developers within the OSS development communities. Importantly, no analysis was done until all the information was gathered. This was strictly followed to avoid any ambiguity or misconception due to wrongly assuming the current development practices.

Once the information on the development practices of each of the three e-learning systems was gathered, they were then modelled using activity flow diagrams. The flowing three sections will explain the development practices of Moodle, ILIAS and Dokeos. The development practices were then compared and are presented in section 4.5.

4.2 Moodle Development Activities

The different activities in case of Moodle development are shown in Fig. 4.1. The first step of development involves selecting the right candidate feature. For selecting a candidate feature, the community pools the entire feature requests raised in the Moodle moot discussions, user's feature request from forums and feature request from moodle vendors. These candidate features are then voted for entering into the release roadmap list. At this stage, it should be noted that there is no clear boundary between various development stages in Moodle when compared with ILIAS/Dokeos (these are explained latter in this chapter).

Any developer interested in developing the new feature listed in the release road map will initiate a discussion with other fellow community developers, in order to ensure that no one else is working on that requirement/feature. The developer(s) will then discuss their ideas with others, confirm the merits and the need for the particular feature, and importantly, evaluate theirs and other's ideas.

Once the feature is selected for development by a Moodle developer, he/she is expected to come with design documents along with other specification documentations. These documents are then posted in the Moodle wiki. In addition, a tracker item is created for the feature and assigned to the developer.

Subsequent changes are to be made, based on the feedback received by the developer in the respective documents which are then updated in the wiki.

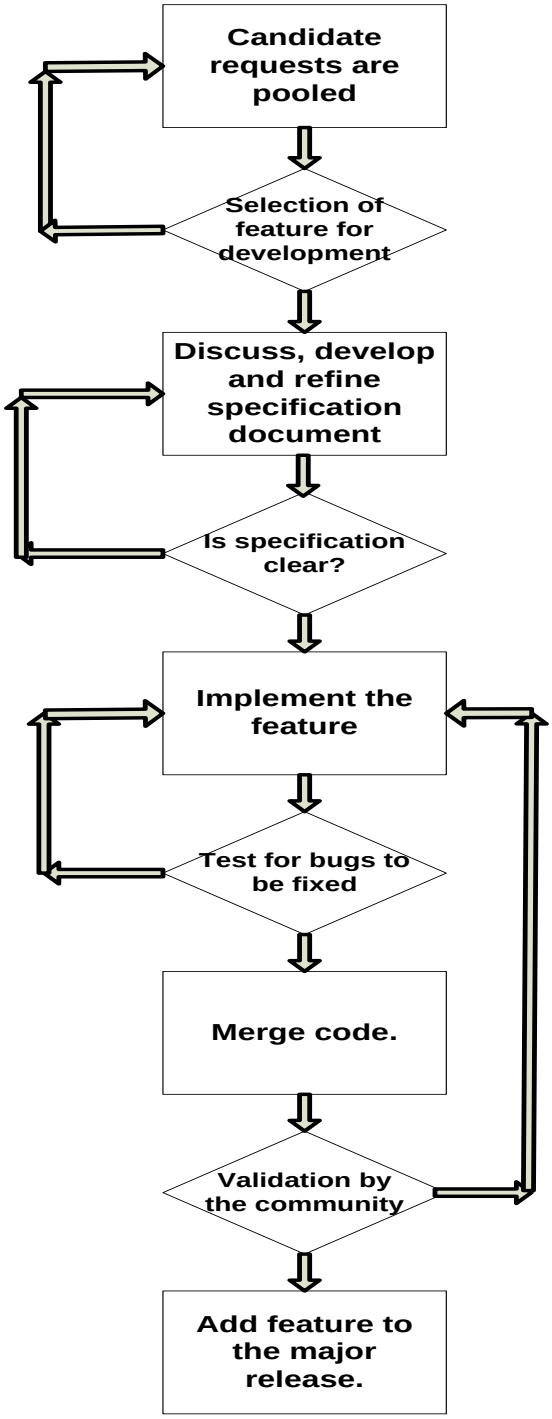

Fig. 4.1 Activity flow representation for Moodle development

Once the changes are made and agreed by the Moodle community, the developer begins coding. Once the development is completed or a major milestone is reached, it is the responsibility of the developer to advertise the feature for testing. Testing could be done by interested candidate(s) within the

moodle community. Subsequently, bugs (if any) are then reported and fixed. It is then integrated with the main version of Moodle and then released as a new version, which would be open and freely available.

4.3 ILIAS Development Activities

ILIAS (ENG: Integrated Learning Information and co-operAtion System) is one of the popular OS e-learning systems and comprises of six stages of development. They are; Vision/Concept, Specification, Implementation, Documentation, Testing and Release & Maintenance. In each of these stages, the OSS community perform various developmental activities which can be observed clearly in Fig. 4.2.

1st Stage: The first stage is about developing the vision or the concept. In this stage, ideas are proposed and published in wiki. The core development team will then decide on how to start the development. If the idea is already been put on to the feature wiki, people with similar interest are requested to work with them and develop the feature collaboratively.

2nd Stage: The second stage is the specification stage whereby, all major development is expected to have corresponding use cases or mock up screen-shots. For other minor developments/enhancements, developers would start with the feature wiki where it will describe the feature in detail, the purpose, etc.

3rd Stage: The third stage in the development of ILIAS is implementation. In this stage, the coding/programming is done by the developers. Each module that is developed in this stage is tested by the developer who also fixes the initial bugs that comes across. Further, the developer would either perform a unit-testing using PHP Unit, or get it done by a tester. Subsequently, the code is then merged with CVS.

4th Stage: The fourth stage is documentation. There are two types documentation prepared for a feature developed for ILIAS - technical documentation and user documentation. The technical documentation consists

of the class and functional documentation generated by PHPDoc. The user documentation will be mainly instructions for the average user on how to use it. The user documentation is only released at the time of release of the product.

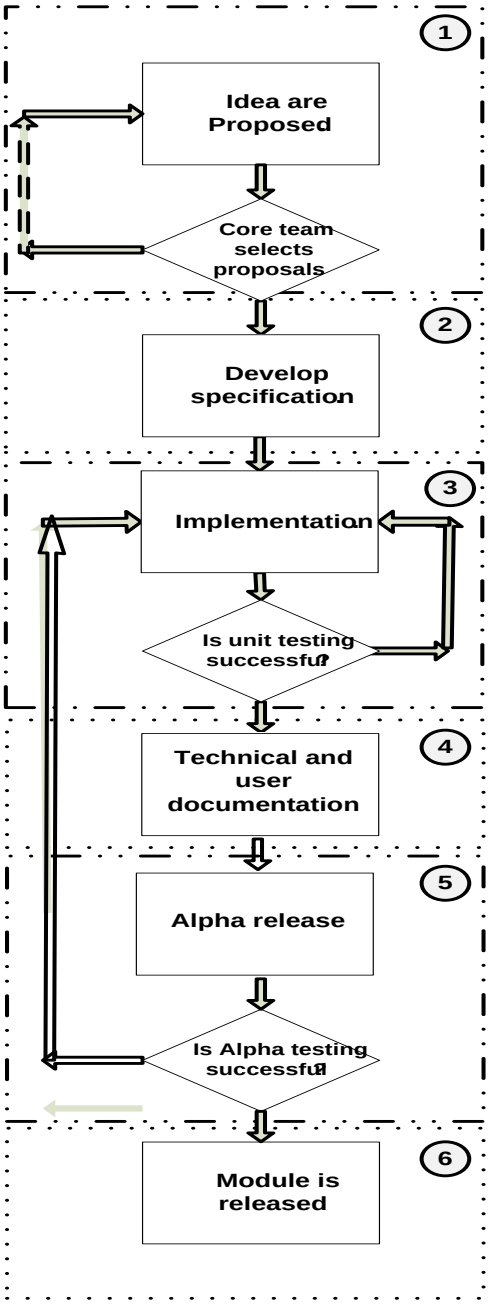

Fig. 4.2 Activity flow representation for ILIAS development

5th Stage: The fifth stage is the testing stage which mainly follows the implementation stage. In this stage, once the unit-testing and code merger is done, an alpha release is carried out for further testing and bug fixing. It is the

responsibility of the developer to appoint a tester to test the module developed by him. If the developer is unsuccessful in finding a tester to test his/her module, then the core team would carry out the required testing. However, in any case, the developer himself cannot be a tester for his own developed module.

6th Stage: The sixth and the last stage is the release stage wherein, the new modules that have undergone alpha testing are released under the beta version. Errors/bugs encountered after the beta release are then entered into the bug tracker (Mantis bug tracker). These bugs are then fixed and released as the main stable version.

4.4 Dokeos Development Activities

Dokeos is developed both as commercial and OSS version. The development of OSS version is the responsibility of the Dokeos community – from initiation of idea through release. Although there are two different existing systems, the OS version does provide all the basic features for free without any licensing cost to its users.

Dokeos community does not follow any defined stages as in ILIAS, but often, they do perform some activities in a particular order as shown in Fig. 4.3. Development of a feature starts with feature selection where the selected feature is added to the roadmap for development. The feature is then developed by the community of developers. The features are first tested before it is given it to the users for further testing. If any anomalies are found they are fixed and then passed on to the users for user testing.

The users would test the developed feature and if they do find any bug(s), they would report it. These bugs are then fixed and once again sent to the user for testing.

Once the user is satisfied with the features, they are subsequently released to the community as a stable version. All the users could then download it for free and use it.

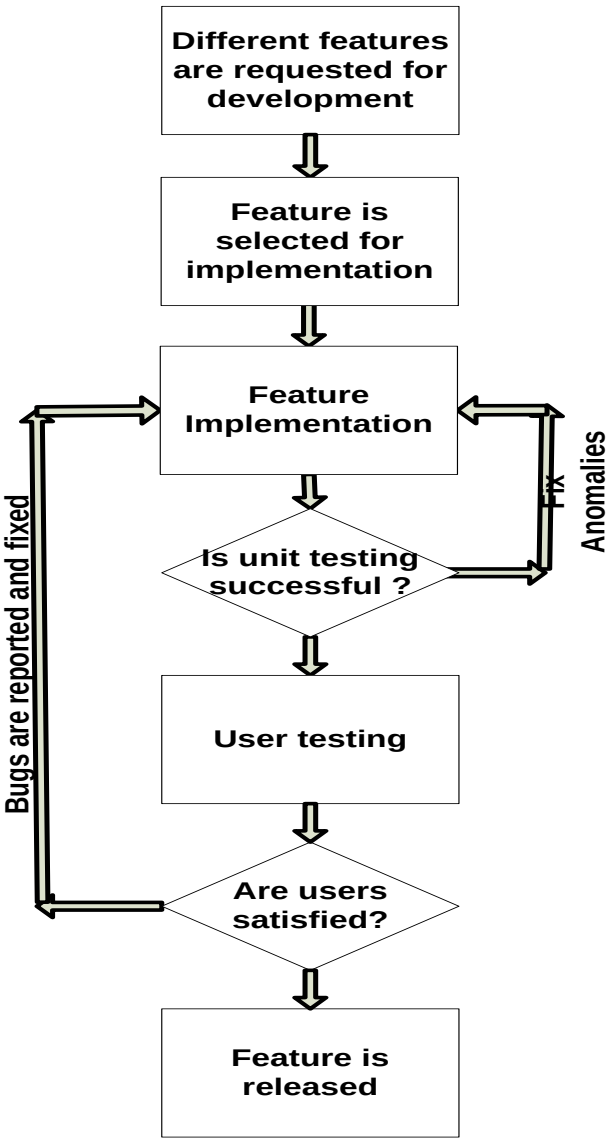

Fig. 4.3 Activity flow representation for Dokeos development

4.5 Comparison of OS E-Learning System Development Practices

The individual analysis of the three OS e-learning systems provide interesting insights into their software developed practices. 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. There are distinct *similarities* and *differences* between Moodle, ILIAS and Dokeos on different aspects. These are summarized in Table 4.1.

LMS / Parameter	Moodle	ILIAS	Dokeos
Number of development stages	Do not categorize development stages	Does categorize six development stages	Does not categorize stages
Who validates the proposed idea	Anyone can validate the idea and comment on it	Only the core team validates the proposed idea	Does not validate the proposed idea at this stage
Detailed development plan	No plan is produced	No plan is produced	No plan is produced
Person(s) responsible for development	A person who volunteered initially & the team that was formed latter on the fly.	A person who volunteered initially & the team that was formed latter on the fly.	Any interested volunteer engages in developing the software.
Testing	Anyone can test at any time.	Anyone can test at anytime.	Anyone can test till the product is released.
Release	Two stage release process is followed.	Two stage release process is followed.	Once the testing is done & bugs are fixed, the product is released. There is no beta release.

Table 4.1 Comparison between three OS e-learning system development

The comparative analysis is based on the development activities carried out by the OS e-learning communities. It begins with differences in number of developmental stages (as defined in chapter 2). The common developmental activities in each of the stages are then compared, based on factors like, *how it has been performed*, *who performs it*, etc. Each of these differences and similarities are discussed briefly and is described as an *observation* and *critique*. The critique is one of the inputs (recommendation) towards the development of the proposed OSSD Process.

- **Number of software development stages**

  Observation: In ILIAS, it is easy to identify different development stages /phases during development. However, Moodle and Dokeos do not categorize different software development stages, even though it has many tasks similar to ILIAS.

Critique: Having defined stages or phases of development are important as it aids in easy tracking of the development activities as well as assists in planning and testing different phases independently.

- **Scrutiny of the proposed idea**

  Observation: New ideas proposed to Moodle and ILIAS is scrutinized immediately after its proposal. At the same time, there is one major difference between Moodle and ILIAS. In case of Moodle, anyone who is interested in the new idea, including the core team, co-developers, testers, users, etc. can read the proposal document and comment on it. Based on the received feedback, the core team or the core members will signal the development. However, in case of ILIAS, only the core members will review the idea/feature and would decide its future. On the other hand in Dokeos, specifications are not detailed or developed for idea scrutinization.

  Critique: Assessing the features credibility and need even before the specifications are developed might lead to inappropriate judgment with regard to the features need and importance.

- **Person(s) responsible for specification scrutiny**

  Observation: In case of Moodle, the entire community could scrutinize the specification by reading the proposal document and commenting on it. Based on the feedback the core team/ members would either agree/ disagree with the idea. On the other hand, as compared to Moodle, ILIAS has a different approach. In case of ILIAS, only the core members would scrutinise the idea/feature decide its future. On the other hand, Dokeos does not have any such activity and therefore the community is not responsible for the same.

  Critique: Being open source and built by users for users, the specification validation should be kept open. This will make sure that the specification is acceptable from the OSS user's point of view. This is very important because, in all cases, development happens based on the specification. If the specification happens to be wrong, then the developed feature would go wrong. This is true for all the software

products including OS e-learning systems, irrespective of the development method followed.

- **Developmental plan**

  Observation: In all three systems i.e., Moodle, ILIAS and Dokeos, there are no explicit plans portrayed for its development. It is the responsibility of the person in-charge to develop the feature as agreed upon. At the same time, it is the individual or team's responsibility to answer all queries regarding the module/feature development.

  Critique: Even though having a defined plan is beneficial in tracking the development; it is very complicated to come up with plans and follow it strictly in the OSS environment where the volunteers develop the product during their free time.

- **Person responsible for development**

  Observation: In Moodle and ILIAS the person who agreed to develop the feature takes responsibility of its implementation. Further, the team formation happen on-the-fly based on the personal interest of the community member(s). If anyone is interested in its implementation, testing, documentation, etc. they volunteer to the working group/person.

  Critique: Even though having a defined plan for developing a feature may seem to be a 'failsafe' approach, it is not practical to follow such a structure in an OSS environment. This is especially so, when a feature is developed by geographically distributed community members who volunteer to do the same in their spare time not just for themselves but also for others.

- **Testing**

  Observation: In all the three OS e-learning systems, any individual from the community who is interested in a particular feature can test the developed code for any potential bug(s). However, there is one notable difference. In case of Moodle and ILIAS, the common ground testing could be carried out even after new versions are released. On the other

hand, in case of Dokeos, this type of common ground testing could be done only till the product is released.

Critique: Testing is one of the important activities in producing a quality software product. OS software products are usually well-tested due to the large number of user-base/testers, who are geographically distributed, have varied skill sets and could test the module/feature independently.

- **Product Release**

  Observation: A two-stage testing process is employed in case of Moodle and ILIAS. Once the initial testing is over, both Moodle and ILIAS release their features as a 'beta' version. Subsequently, this is tested again. Once the testing is completed, the features are then finally released along with other items as final version of the major product release. On the other hand, Dokeos does not have any beta release. The feature(s) are tested by users/community once it is developed and the bugs are reported. Once the encountered bugs are fixed, the feature is subsequently released.

  Critique: Having a beta test stage will enable identification of problems before the integration to the stable version. This would potentially save any additional costs (in most cases it's the time spent by the OSS community) that might have to be incurred if the stable version is corrupted.

4.5 Summary

This chapter provided a state-of-the-art overview of the developmental activities followed by three different OS e-learning systems. The development activities were presented using an activity flow representation, primarily because, it is easy to use, understand, interpret and compare. Following this, the corresponding developmental activities were compared. This demonstrated the clarity and explicitness of the different stages of development for each of the OS e-learning systems. At the same time, there were two main limitations with this type of representation. Firstly, this representation does not identify precisely which actor(s) were involved in carrying out a particular task/

activity. Secondly, the activity flow representation does not specify the outcome of a particular activity. In-order to overcome these drawbacks, a model that gives a high level view, needs to be constructed which will in-turn focus on the actors activity and outcome, instead of just looking at how a particular activity have been performed. Hence, DEMO methodology has been used subsequently. A DEMO model prescribes various models that can be drawn to depict the development practices followed; and is described in detail in chapter 5.

5. DEMO Methodology

5.1 Introduction

This chapter describes the DEMO models for OS e-learning systems, based on DEMO methodology and subsequently, explains in detail, the two critical models used to model all three OS e-learning systems selected for this research work. The resulting information from these models forms the basis for developing the proposed OSSD process.

5.2 DEMO Models

There are several ways (i.e., numerous diagram representations) for modelling a development process using DEMO methodology. They include: *State model*, *Action model*, *Interstriction model*, *Process structure diagram* (PSD) and *Actor transaction diagram* (ATD). Of these, the last two - PSD and ATD - enable in obtaining a high-level and abstract overview of the process used for development. Hence, they are very essential models that are always developed for a given process or organization.

The PSD details the interactions of each transaction and also between the transactions. On the other hand, the ATD shows the various actors' involvement in specific communication for executing a task. Also, it shows 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-order to make the diagrams compact, the act and fact related to each process step were merged into single symbol. A combination of a P-act and P-fact was represented by a diamond in a square, while a combination of a C-act and C-fact was represented by a circle within a square. Also, an arrow with a solid line represent the normal process flow, while an arrow with a dotted line represent a wait condition.

For an ATD, a single symbol was used for each transaction, which contained all the process steps. This symbol was represented by a diamond in a circle ⦻, in order to represent the combination of the P-fact and C-facts related to the transaction. The initiator was connected to the transaction symbol by a solid line (—). The executor was connected to the transaction by a solid line ending in a black square (—■).

DEMO models (ATD & PSD) were constructed for all three OS e-learning systems (Moodle, ILIAS and Dokeos) and elaborated in section 5.3, 5.4 and 5.5 respectively. Further, the achieved P-facts are also described for the corresponding activities carried out for each OS e-learning system development. Importantly, these diagrams were constructed under the assumption that all the activities carried out during the development of e-learning system have been successfully completed at once. This might not be the case in the real world as not all activities are successful until the activities are iterated/ customized whenever required.

5.3 DEMO Models for Moodle

The ATD for moodle development is shown in Fig. 5.1, 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. 5.1 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. Each of the transactions (T01 through T011) can be further expanded into individual PSD's.

In the PSD, each transaction is detailed with expressions of communication (rq, pm, st & ac) and indicates the execution phase (ex) which when

successful, produces a P-fact. Fig. 5.2, Fig. 5.3 and Fig.5.4 describes various activities carried out from the conception of the idea till the idea is productized and released. These PSD's are divided based on the general developmental stages such as *Moodle feature selection & requirement specification*, *construction*, *testing* and *release*.

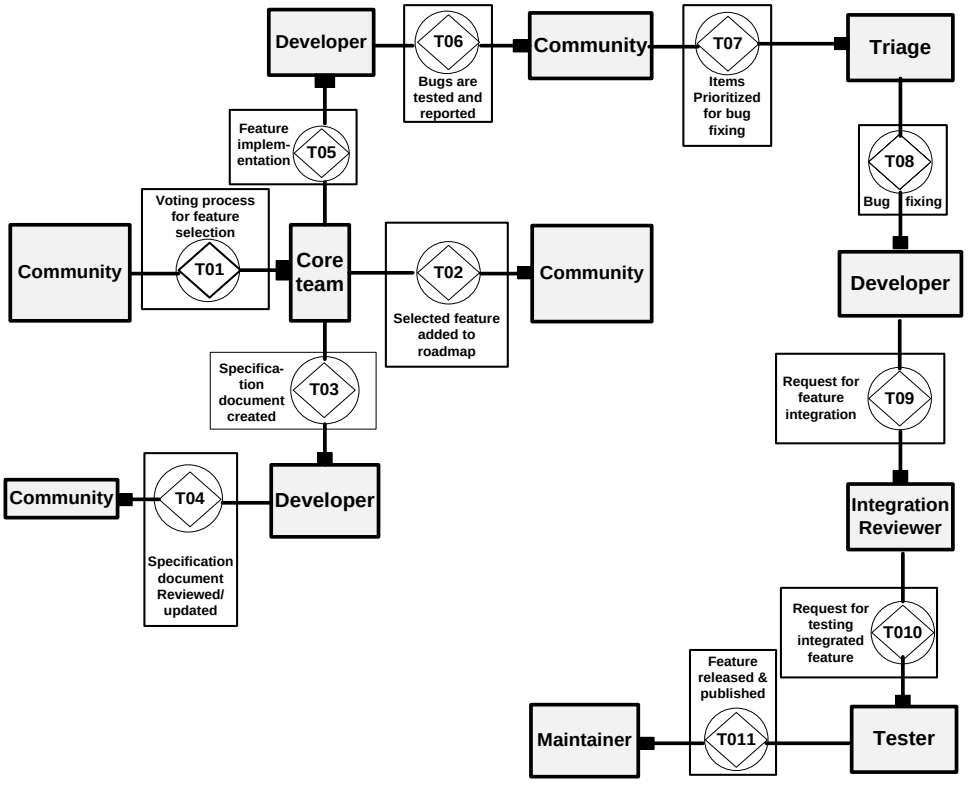

Fig. 5.1 ATD representation for Moodle development

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 T01 which 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 which the entire community provides, based on the specification released earlier.

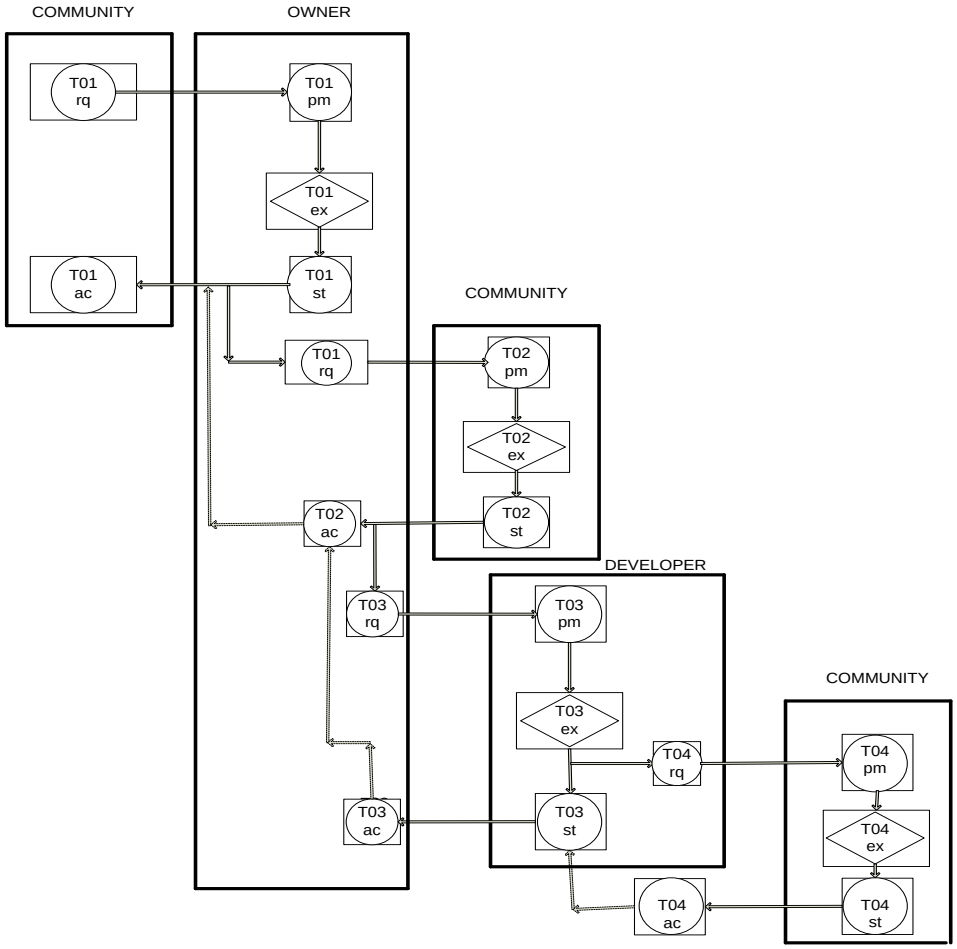

Fig. 5.2 PSD for Moodle feature selection and requirement specification

The next stage in moodle development is the implementation of the selected moodle feature. The PSD for Moodle development is shown in Fig. 5.3. 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.

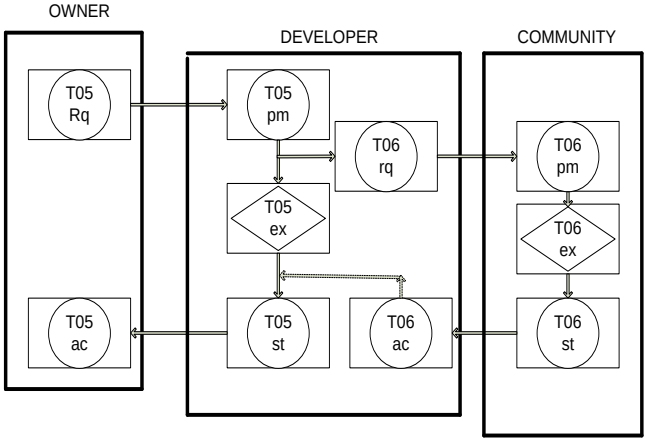

Fig. 5.3 PSD for Moodle implementation

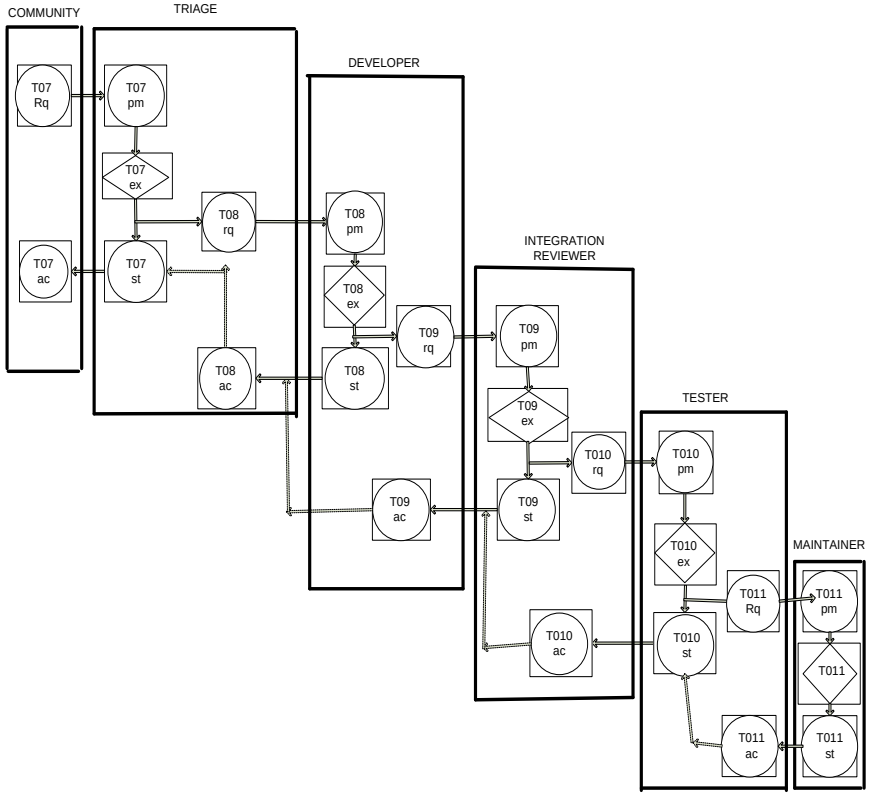

Fig. 5.4 PSD for Moodle testing and release

Transaction(s)	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	Developed feature is tested by the community and bugs are reported.
T07	Reported bugs are prioritised.
T08	Bugs are fixed.
T09	Features are added to the integration queue.
T010	Features are integrated and tested.
T011	A stable feature is released.

Table5.1 P-facts produced during Moodle development

Once the implementation was successfully finished, the feature is then tested and released to the Moodle-using community. Fig. 5.4 depicts the roles involved in carrying out the 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. For a quick review, the P-facts produced during Moodle development are summarised in Table 5.1.

5.4 DEMO Models for ILIAS

The ATD for ILIAS feature development is shown in Fig.5.5. The various actors' involved in its development are: the user community, core team, developer, tester and maintainer. The transactions carried out for its development are denoted from T01 through T09.

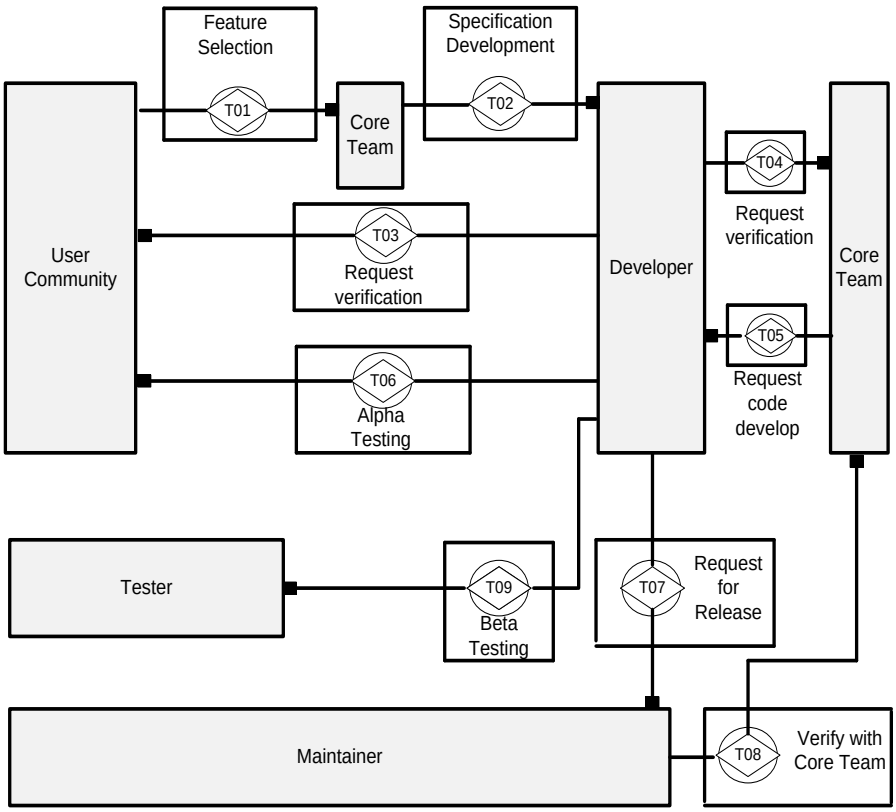

Fig. 5.5 ATD representation for ILIAS development

The PSD is divided based on the general software development phases. Fig.5.6 shows the PSD for ILIAS feature selection. 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.

Fig. 5.7 represents the PSD for ILIAS requirement specification development. The 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.

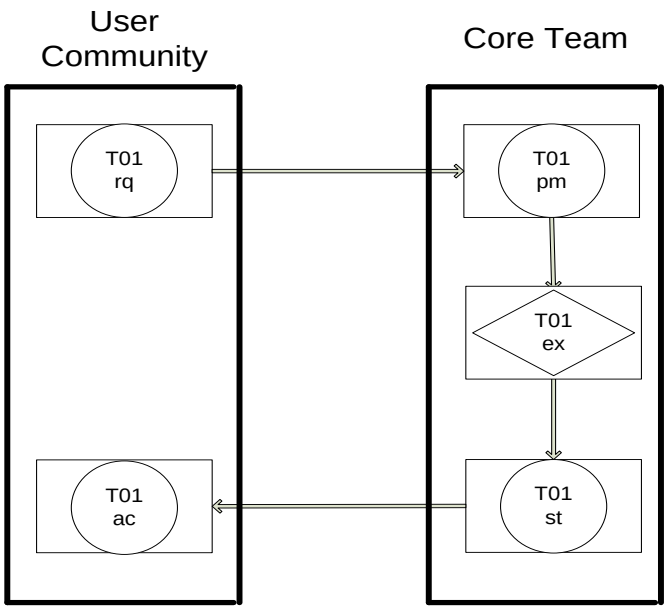

Fig. 5.6 PSD for ILIAS feature selection

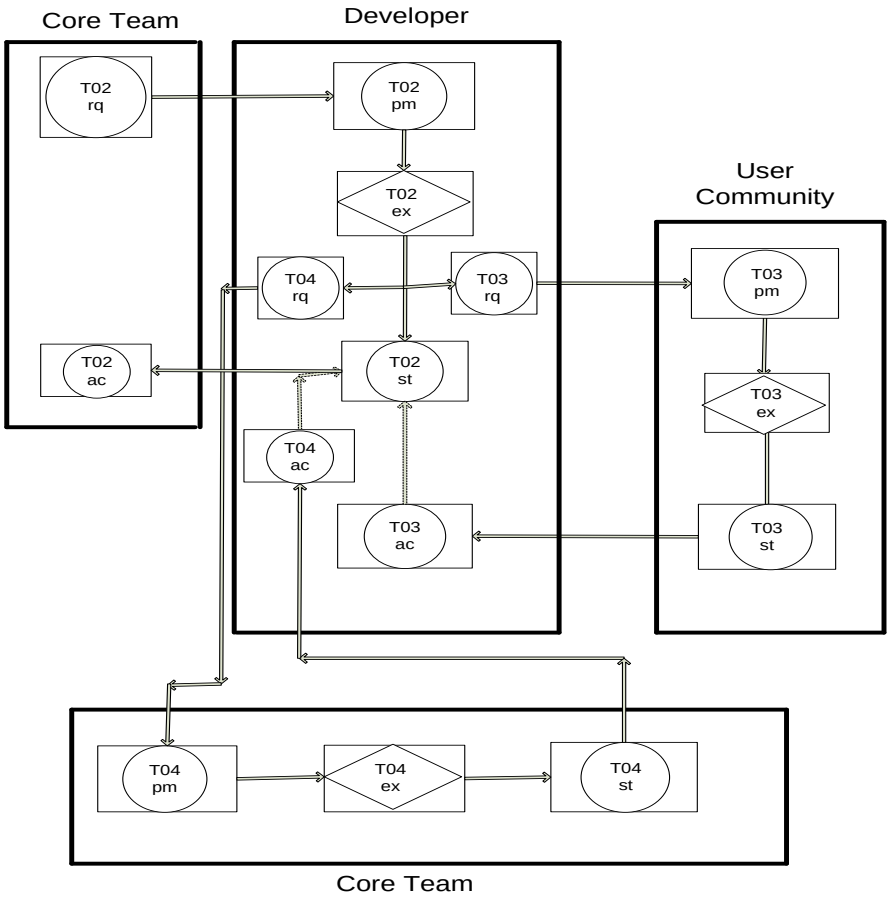

Fig. 5.7 PSD for developing requirement specification

Fig. 5.8 shows the PSD for feature implementation. 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.

Fig. 5.9 shows the transactions involved in testing and releasing the ILIAS feature. The actors involved 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.

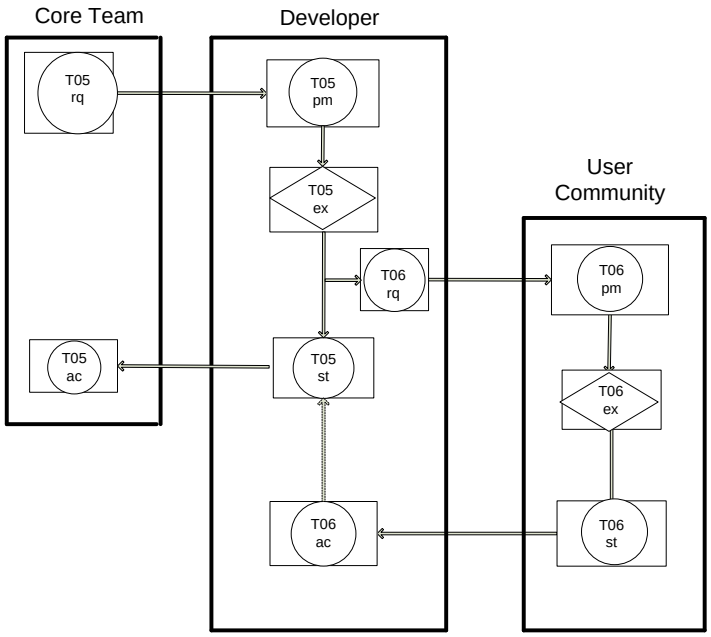

Fig. 5.8 PSD for ILIAS feature implementation

In this sub-section, each transaction represented in ATD is elaborated with corresponding process structure diagram for ILIAS. Also, the P-facts are highlighted for each of the transactions assuming they were successful. The P-facts have been summarised and are presented in Table 5.2 for a quick review.

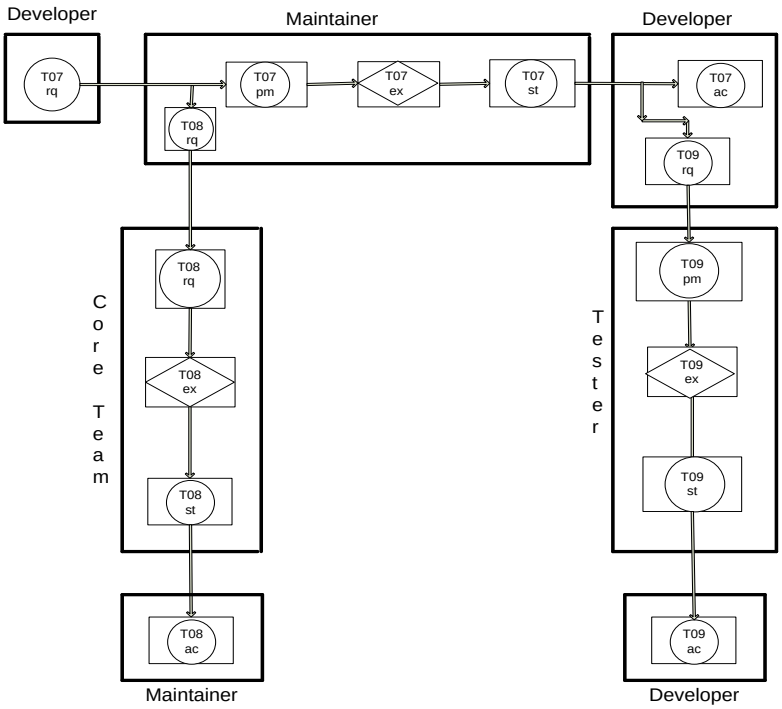

Fig. 5.9 PSD for ILIAS testing and release

Transactions	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.

Table 5.2 Summary of ILIAS P-facts

5.6 DEMO Models for Dokeos

The ATD for Dokeos development is shown in Fig. 5.10. 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). ATD will be followed by the PSD

and the ATD's are split into PSD's based on the general software development stages.

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 as shown in Fig.5.11. 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.

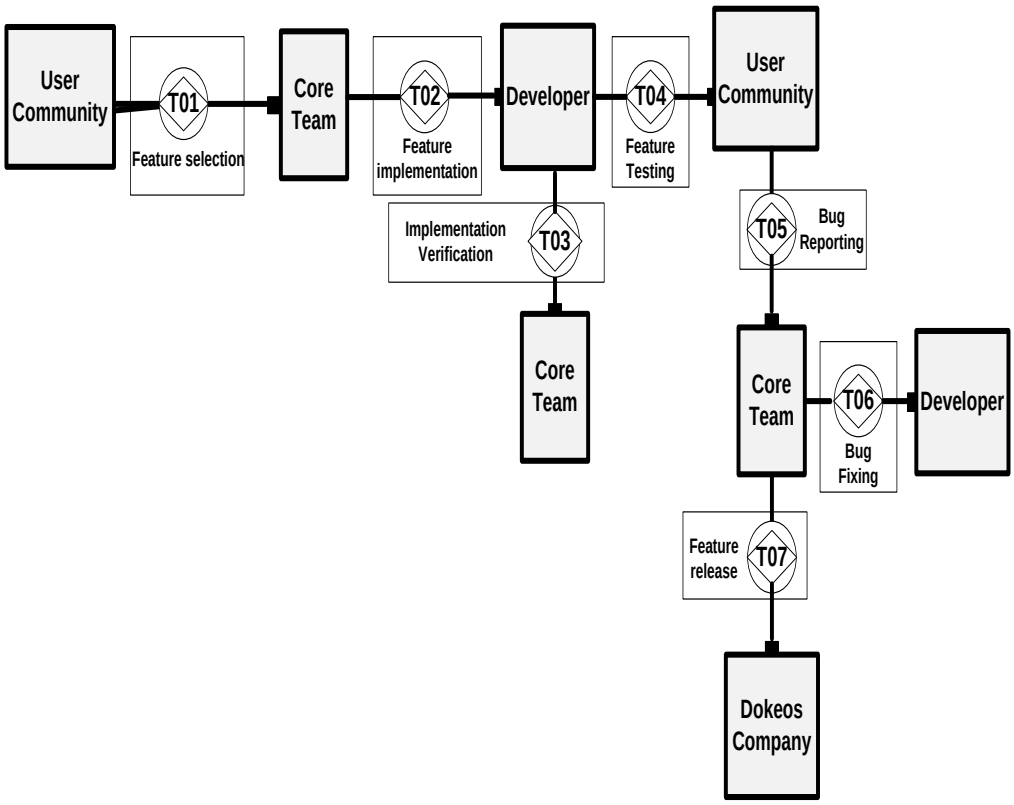

Fig. 5.10 ATD representation for Dokeos development

Once, the feature is selected by the core team for development, the developers are requested to build the feature which is depicted in transaction T02 in Fig. 5.12. 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.

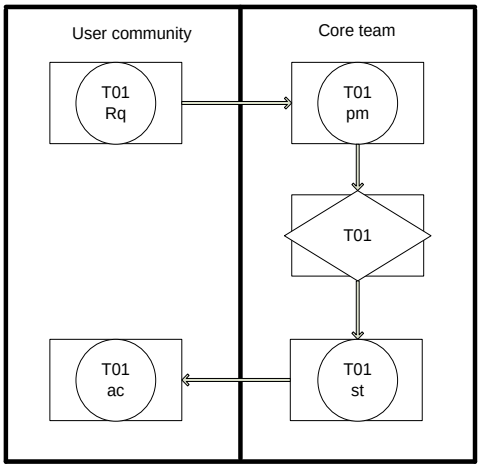

Fig. 5.11 PSD for Dokeos feature selection

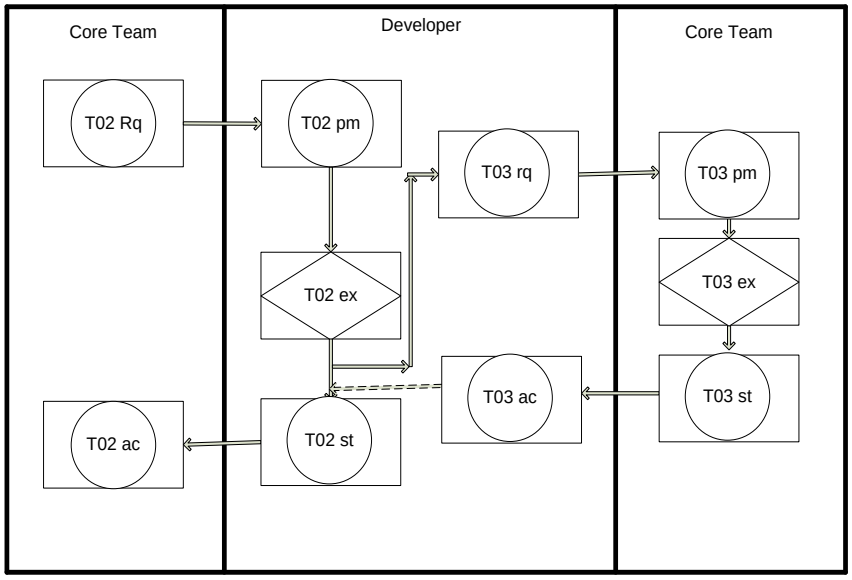

Fig. 5.12 PSD for Dokeos feature development

Fig. 5.13 shows the PSD depicting the communication pattern between the developer, core team and the user community for testing and fixing the bug. 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, categorised 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.

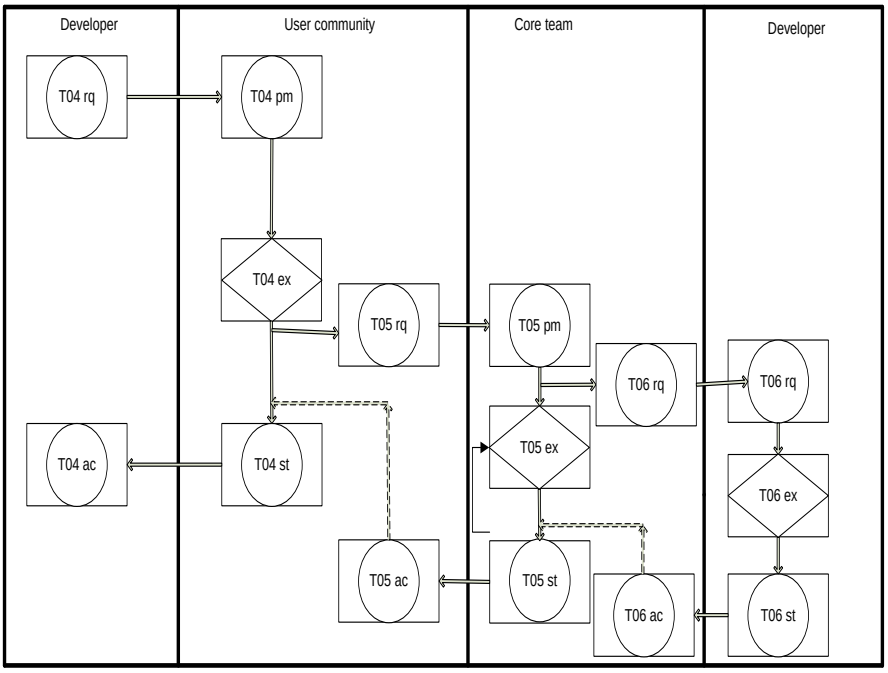

Fig. 5.13 PSD for Dokeos testing and bug fix

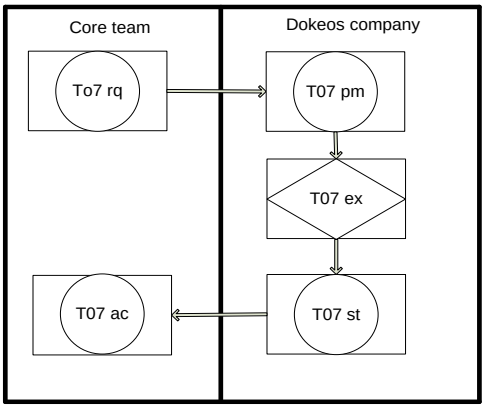

Fig. 5.14 PSD for Dokeos feature release

Transactions	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 prioritised.
T06	Bugs are fixed.
T07	Feature is released.

Table 5.3 Summary of Dokeos P-facts

Fig. 5.14 depicts the release process in the PSD. The core team initiate 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. Table 5.3 represents the summary of various P-facts that are produced during the development of Dokeos.

5.7 Discussion

Chapter 4 and chapter 5 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.

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 and maintenance	☑ [T09, T010, T011]	☑ [T07]	☑ [T07]

Table 5.4 Inputs for the proposed OSSD process

Table 5.4 presents various transactions executed (chapter 5) for different basic development stages (chapter 2, 4). For each of the three OS e-learning system development, if a particular development stage was identified as being executed (chapter 4 – Activity flow diagrams), then a tick mark '☑' is placed in the corresponding cell in Table 5.4; otherwise a cross mark '☒' is placed. Also, the transaction executed (Chapter 5 – DEMO models) 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, though it is an important input for the proposed process, the proposed OSSD process cannot be generalised based on this information alone.

5.8 Summary

This chapter described DEMO models and its associated terms. These include detailed information on the two key models (ATD & PSD) and its application on the three selected OS e-learning systems. Further, the application of DEMO methodology helped in identifying different actors involved in carrying out various development activities, along with the output of each such activity. Importantly, the results were found to be totally independent of how each of the development activities were carried out within each OS e-learning system community. The drawbacks of activity flow representation could thus be overcome. Further, a detailed discussion was carried out on the important inputs towards the proposed OSSD process. The next chapter discusses on how these results are used in conjunction with the standard ISO/IEC 12207:2008 in order to generalise the proposed OSSD process.

6. Development of OSSD Process

6.1 Introduction

This chapter begins with consolidating the results of DEMO model and activity flow diagram. This is followed by an overview of ISO/IEC 12207:2008 standard, the software-specific processes prescribed in the standard and the list of expected outcomes for each software development activity conveyed in the standard. Subsequently, the proposed generalized OSSD process is explained in detail; along with the different development stages, the ordering and the frequency at which each stage has to be carried out and the major activities in each stage.

6.2 Comparative Results of Various Development Stages of Three OS e-learning Systems

The activity flow diagrams and DEMO models constructed for Moodle, ILIAS and Dokeos had two major benefits. Firstly, it helped in identifying different implicit stages of development. Secondly, it helped in identifying the outputs of various activities in each stage of development and the actors involved in the same.

OSS Systems / Development Stages	Moodle	ILIAS	Dokeos
Requirement analysis	✓	✓	✓
Detailed design	✓	✓	X
Implementation	✓	✓	✓
Testing	✓	✓	✓
Integration	✓	✓	✓
Release	✓	✓	✓

Table 6.1 Developmental stages carried out by OS e-learning systems

Table 6.1 shows whether the three e-learning systems has carried out an activity pertaining to particular development stage. It can be seen from Table 6.1 th at Moodle and ILIAS have carried out few/many developmental activities for all six stages while Dokeos has not performed any activity with regard to detailed design stage. However, the results of activity flow diagram and the DEMO models do not specify the extent to which the different activities are carried out in each development stage. Hence, selecting different development stages for the proposed generalized OSSD process just based on the stages shown in Table 6.1 is not adequate. Before designing a generalized OSSD process, it is important to understand the extent to which the different activities are carried out for the three e-learning systems.

For proposing a generalized OSSD process, there are two key inputs that assist in identifying the extent to which each activity is carried out. The first key input is the result obtained from the DEMO models that identifies the output created by each of the development activities. The second key input for proposing the OSSD process is the ISO/IEC 12207:2008 standard. The standard provide complete details of various software development processes, different activities carried out in each processes and also their corresponding set of all outcomes. With this information, it is possible to judge how much effort has been spent by each of the OS e-learning system development community on these stages. At this point it should be noted that proposing a generalized OSSD process based on the ISO standard not only makes the process more consistent and reliable but also signifies its applicability in real world situation. The next section describes the ISO/IEC 12207:2008 standard.

6.3 ISO/IEC 12207:2008

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. Further, each of the process groups is divided into various lower level processes (International

Standard, 2008). The interesting domain for our research is the various outcomes listed for the software implementation processes which in fact, are a sub-division of software specific processes.

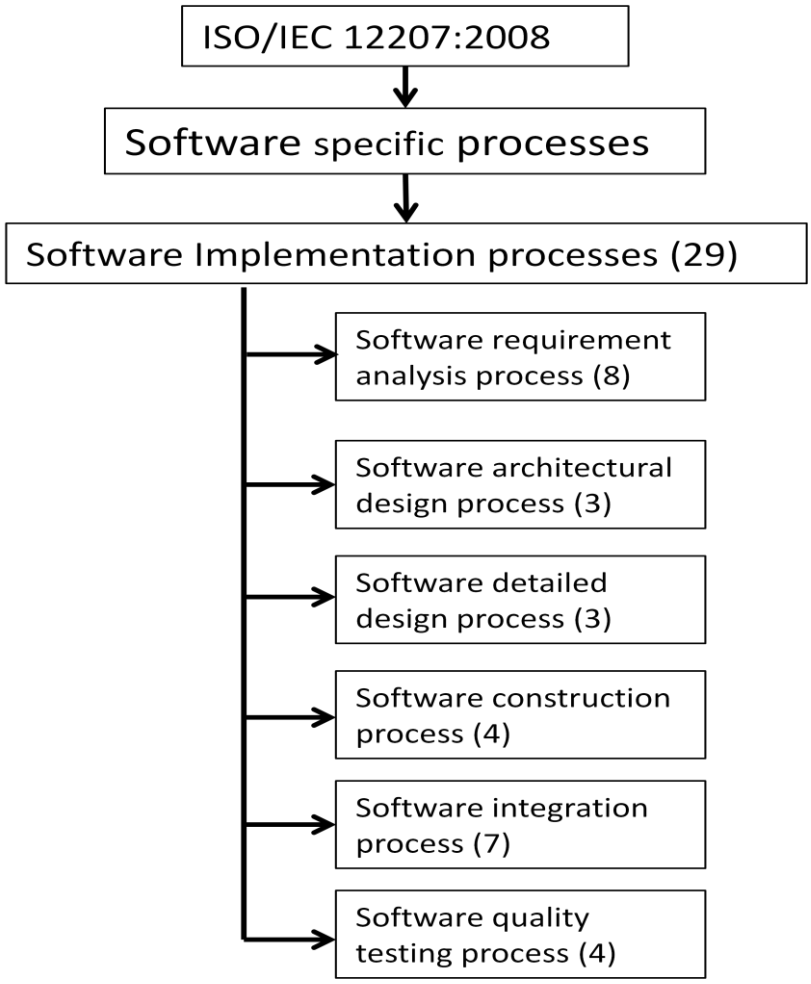

Fig. 6.1 Software lifecycle groups in ISO/IEC 12207

Software implementation processes is divided into six lower level processes as shown in Fig. 6.1. These are *software requirement analysis processes*, *software architectural design processes*, *software detailed design processes*, *software construction processes*, *software integration processes* and *software quality testing processes*. The numbers mentioned within the parentheses in Fig. 6.1 indicates the number of desired outcomes for each of the processes. According to ISO/IEC 12207:2208 standard, there are 29 outcomes that can be achieved by successfully carrying out the software implementation process and its corresponding activities and tasks. These 29 outcomes are divided among their six lower-level process. Table 6.2 lists all possible outcomes that can be expected when these lower level processes are completed successfully.

Lower Level Process		Possible Outcomes
Software Requirement Analysis Process	***RA1***	Requirements of software element & interfaces are defined
	RA2	Requirements analysed for correctness & testability
	RA3	Understand the impact of the requirement on operating environment.
	RA4	Consistency and traceability between s/w and system requirement are drawn
	RA5	Software requirement for implementation are defined
	RA6	Software requirements are approved and updated
	RA7	Changes to the s/w requirement are evaluated for cost, schedule & technical impact
	RA8	Software requirements are base-lined and communicated to all affected parties
Software Architectural Design Process	***AD1***	Software architecture is designed and base-lined
	AD2	Internal and external interfaces of each s/w item are defined
	AD3	Consistency and traceability is established between requirement and design
Software Detailed Design Process	***DD1***	Detailed design of each software component is defined
	DD2	External interfaces of each software units are defined
	DD3	Consistency and traceability are established between architectural design, requirement and detailed design
Software construction process	***CP1***	Verification criteria defined for all s/w units against their requirement.
	CP2	Software units defined by design are produced.
	CP3	Consistency and traceability are established between software unit, design and requirement.
	CP4	Verification of the software unit against requirement and design is accomplished
Software 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 and traceability are established between s/w design & s/w item.
	IP7	Regression strategy is developed and applied for re-verifying s/w items when change occurs in s/w unit
Qualification and 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 for re-testing the integrated software when a change in s/w item is made.

Table 6.2 ISO/IEC 12207 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	—	—	—

Table 6.3 Comparison with ISO/IEC 12207 process groups

The ISO/IEC 12207:2008 standard is used as a foundation for this research as it provides a detailed guideline for software specific processes. 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 6.3. 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.

It can be observed from Table 6.3 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 till have a huge scope for improvement in different stages of development. A percentage of achievement is calculated for each of the development stages based on 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 6.4 shows the percentage of achievement for each of the six stages for all three OS e-learning systems, along with the overall achievement ratio.

OS Systems / Development stage	Moodle	ILIAS	Dokeos
Requirement analysis process	50%	50%	25%
Architectural design process	0%	0%	0%
Detailed design process	66%	66%	0%
Construction process	100%	75%	75%
Integration process	57%	42%	42%
Qualification and testing process	50%	50%	0%
Overall percentage	**53%**	**47%**	**23%**

Table 6.4 Percentage of process coverage per stage

The achievements listed in Table 6.4 shows the achievement ratio (approx.) and thereby, 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*. The next section describes the proposed generalized OSSD process.

6.4 Proposed Generalised OSSD Process

A generalized OSSD process could be used by the OS community to develop new e-learning systems or could be applied to the existing e-learning system development. Particularly, according to the software development process definition (Chapter 2), the generalized OSSD process would specify the following:

- The different stage of development and their ordering
- The frequency with which each development stage is executed
- The important activities involved in each development stage.

At this stage, it should be noted that the proposed OSSD process does not specify on how a particular activity should be carried out. Further, it does not enforce the community on who should carry out a particular activity but rather provides guidelines on various stages of development, along with the major activities for each development stage that the OSSD community should follow, while developing OS e-learning systems.

6.4.1 Overview of Development Stages

The three OS e-learning systems considered in this study have activities performed in five out of six stages. However, the degree of completion with respect to each development stage is different for each OS e-learning systems. Hence, the percentage of outcome achieved by the OS e-learning systems for each of the lower-level processes stated by the standard is considered as an important criterion for selecting the different stages of development. In-order to do so, a four-level classification is considered for the percentage of outcome achieved; and is shown in Table 6.5. If a particular e-learning system has achieved 0 - 15% of the outcome it is considered to be '*NIL*'. If it is 15.01 - 50%, then it is stated as '*Partial*'. If an OS e-learning system has achieved 50.01 - 90%, it is stated as '*Major*'. Finally if a particular e-learning system

has satisfied 90.01 - 100% of the outcomes as stated by the standard then it is termed as '*Complete*'. If any of the three OS e-learning systems' outcomes prescribed for the lower level process in the standard is in the category, '*Complete*' or '*Major*', then that particular lower level process/development stage is *added to the proposed OSSD process*. In addition, if two out of three OS e-learning systems has performed a particular lower level process and the expected outcomes are categorized under '*Partial*', then again, it is added to the proposed OSSD process, with some suggestions for improvement.

Outcome Achieved in %	Category
0 – 15%	NIL
15.01 – 50%	Partial
50.01 - 90%	Major
90.01 – 100%	Complete

Table 6.5 Category based on percentage of process coverage achieved

For the proposed OSSD process, five development stages are selected from the existing OS e-learning systems. They are: design specification stage, implementation stage, software testing stage and integration & release stage. Notably, the architecture design stage is not selected from Table 6.4. There are two reasons for the same. Firstly, none of the three major OS e-learning systems have considered it in the design, Secondly, OS e-learning system is a continuously evolving software product and hence, there is no specific stage allotted for architecture design. However, at the same time, an additional stage is considered. This is the *feature selection stage* that is added as the first development stage. Though feature selection and its corresponding activities have been carried out implicitly by the OSS community, the proposed OSSD process makes this an explicit development stage. This is because; it is a crucial starting point for any feature to be developed for an OSS system. Hence, the proposed OSSD process has six development stages. They are:

- *Feature selection stage*
- *Requirement specification stage*
- *Design specification stage*
- *Implementation stage*
- *Software testing stage*
- *Integration and release stage*

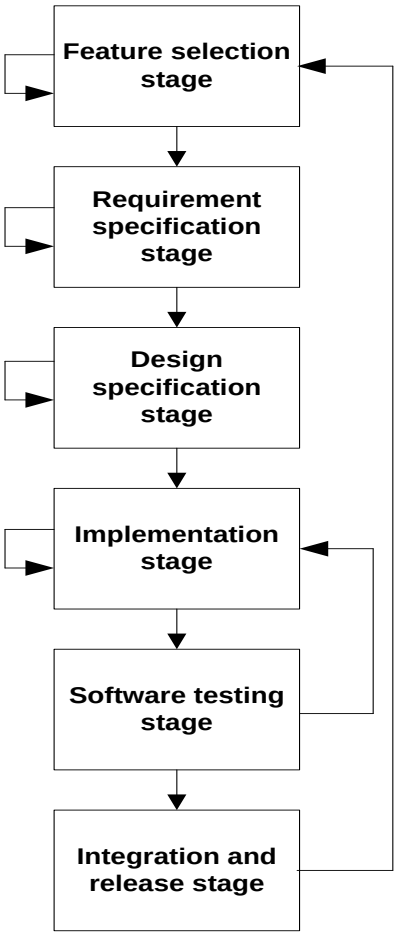

Fig. 6.2 Different stages in the proposed generalised OSSD process

Fig. 6.2 depicts the proposed OSSD process with six development stages. Of these six stages, the first four stages are iterated before proceeding to the next stage. Further, the iterations between the stages prescribed in the proposed OSSD process can be seen commonly in many development processes and also within the OSS development community's current development practices.

These iterations are proposed to be carried out based on the following:

- Iterations are proposed for the stages where many number of geographically dispersed community members work together in achieving a particular activity/task; or if more than one type of actor is involved in finishing a particular activity.
- Iterations are prescribed for those stages where the completion of an activity of one development stage depends upon the activity of another stage.

Further sections in this chapter will present each development stage along with the suggested important activities, followed by a detailed discussion on the same.

6.4.2 Stage 1 – Feature Selection Stage

Description: It is the first development stage where the right candidate feature(s), are selected for development by the OSS community.

Suggested Important Activities:

(a) Development of 'feature requirement document' and its further review, before they are selected for development.
(b) Selection of the feature by the entire community, based on the feature requirement document.
(c) Addition of all selected features to the feature development roadmap.

Discussion: A feature requirement document can be a wiki document/ general document that could be attached in the community forum or any other mechanism that the community is comfortable with. This document can be very brief and should state the purpose of the feature, the beneficiary of the feature and other user related and technical details (depending upon the proposed feature). This document should be reviewed by the core team and once reviewed and satisfied, these documents should be published openly to the entire community. This document not only helps in identifying the correct candidate feature but also helps the community as a whole to understand what is going to be developed for their OSS.

In addition, the selection of the feature should be based on the feature requirement document. The entire community members are required to participate in its selection and are a mandatory activity. The OSS community can adopt any mechanism to encourage its community to involve themselves in this activity (feedbacks, voting, etc.). Once the feature is selected by the community these are then added to the feature development roadmap. This roadmap would contain the entire list of features that are selected for further development. Further, a person/team is initially appointed / selected for each of

the selected candidate features. Having a dedicated contact person makes it easy to manage/ engage with the community at the initial stages of development.

Iteration: The feature selection stage can be iterated few times in order to make sure that the feature requirement document is clear enough for the community to understand and also to encourage more and more community members to participate in the selection process.

6.4.3 Stage 2 – Requirement Specification Stage

Description: Requirement specification is the second stage in the OSSD process where the software requirements are identified and elucidated before proceeding towards software development.

Suggested Important Activities:

(a) Development of the requirement specification documents; and subsequent iteration until it is widely accepted.
(b) Identifying the developers/team of developers who would work on the selected feature.
(c) Verifying and freezing the requirement-specification document for the latest product release.

Discussion: The requirement specification document that is developed in this stage should briefly reiterate the purpose of the feature, followed by identification of the clients/stakeholders/users. In addition, this document should list the constraints, along with functional and non-functional requirement. Further, they can also specify any open issues or any new problems in the software requirement specification document, so that the community members when reading might help the developer/developer's team with some suggestions. Also, the OSS community should start identifying the team members/developers (volunteers) who would be working on each of these selected candidate features for development. Notably, the core team should review the requirement specification documents along with newly formed team of developers before being published openly to the entire community. At this

stage, the core team can freeze these documents once the community is happy with the requirements.

Iteration: The requirement specification stage is iterated until the document is clear and satisfies the user requirement.

6.4.4 Stage 3 – Design Specification Stage

Description: Design specifications are developed based on the accepted requirements which will be the basis for software implementation.

Suggested Important Activities:

(a) Development of design specification document for the selected feature that would satisfy the requirements.

(b) Amendment of the design document, if required, based on the core teams feedback/suggestions and its publication to the entire community.

Discussion: The design specification that is developed for the selected feature should describe how the feature is going to be implemented (coding). It is totally up to the OSS community to decide upon the language that they are going to use within their community. The newly formed team (selected/volunteered in the development stage) should come up with this design document. The core team would review these design documents and would give comments/feedbacks on the same in order to improve the design document. The developers are required to make the necessary changes and make it available to the entire community. This gives a clear picture to the community members about the feature to be developed. Also, it would give an idea on how to use the feature and clear other basic doubts that they might have. In addition, design experts within the community could give their opinions/suggestions during development which might be helpful for the developer/developer's team.

Iteration: This stage is iterated few times in order to make sure that the design document is representing the feature requirement and is clear enough for the community to understand.

6.4.5 Stage 4 – Implementation Stage

Description: Implementation stage is a one where the developer/developers team implement the feature (coding) based on the design document to satisfy the user requirement and produce a workable software feature.

Suggested Important Activities:

(a) Development of a brief implementation document.

(b) Implementation of the selected feature based on its design specification.

(c) Development of unit testing strategy.

Discussion: The developers should be encouraged to decide how they are going to approach the implementation. They should ensure that it is clear enough to be written as a brief implementation document; along with initial and tentative deadlines for the feature to be implemented. Developing an implementation document encourages the community to actively participate in testing the feature once the local release is done. In-addition, the developer/ developer's team needs to update the community regarding the implementation periodically using community wiki's, blogs or any other social media that is used within the community. An important task in this stage is to actually implement the feature. In addition to this, the developer/ developer's team are required to come up with a simple unit testing strategy. The OSS community could identify/volunteer/elect/appoint a person/team from within the development team and can use their own template to develop the unit testing strategy. This unit testing strategy can be published publically for anyone to use it after the local release. Unit testing helps the OSS developer/developer's team to identify any issues early in the development cycle, facilitates any changes that need to be done, simplifies the integration process, etc. Defined unit testing strategy enables them to identify the bugs before the local release and can also be fixed.

Iteration: The implementation stage is iterated few times until the development team/developer is satisfied with what is already implemented. Also, these are iterated to fix any issues identified during unit testing.

6.4.6 Stage 5 – Software Testing Stage

Description: In the software testing stage, the implemented software is tested to satisfy the design document that was developed based on the user requirements.

Suggested Important Activities:

(a) Development and verification of initial and important test cases for the OSS community.
(b) Testing the locally released OSS feature.
(c) Reporting the bugs encountered and fixing the same.

Discussion: The OSS environment has the biggest advantage of having a huge number of testers/community members to identify any issues/bugs before the feature is released as a part of major product release. In order to take advantage of this, the initial and important test cases are published openly to the community members. This may help the community to head start the testing process and they may then further explore the feature through testing. In addition, this might help any new testers within the community to understand how to perform testing before the major release. The developer/developer's team could propose these test cases which the core team or any person appointed by the core team could approve/make changes as required and post it to the community. The entire community should be encouraged to take part in the testing activity. The community member could use the initial test cases to commence testing. Also the community could be given a time frame within which they could carry on testing and at their ease. For instance, the time frame can be until a week or two before a major product release. Also, the proposed OSSD process suggests the OSS community to have their bug tracking system in place. This will help the community to report all the identified bugs/issues in one common place from which the developers/volunteers from within the community can fix the issues without missing any important fixes they are suppose to do before the major product release.

Iteration: There is a need for multiple iteration of this stage for developing, correcting and approving the initial test cases. Once the bugs are identified, it has to be fixed by the developer/developer team and therefore there is a need for iteration between the implementation stage and testing stage until the developed feature satisfies the users.

6.4.7 Stage 6 – Integration and Release Stage

Description: This is the final development stage wherein, the developed and tested feature is integrated with the main OSS product and released as a part of the main product to all its community members and users.

Suggested Important Activities:

(a) Verifying the list of features under developed roadmap and making sure that they are developed and tested.
(b) Developing release roadmap before the actual release.
(c) Verifying the implemented feature before integration and release.
(d) Updating the release roadmap list if required; followed by integration and release of the OSS feature as a part of main OSS product.

Discussion: It is important to make sure that all the items listed under the development road map (6.4.1.1) are developed and tested successfully before the final integration and release. The release roadmap is then developed which lists all the items that would be integrated and released. These features should be released along with all the necessary supporting documents for all such items. Further, the core team has to verify if all the features implemented satisfies the requirement and design specification that are developed for that item (6.4.1.2, 6.4.1.3). Once these release items are verified and signed off by the core team/ responsible person, the final list of release items are published along with its supporting documents, which are then integrated and released in public domain.

Iteration: The OSS development supports continuous evolution. Hence, once the feature is integrated and released, it is iterated back to the ‘feature selection stage’ (6.4.1.1). The process starts again with identifying the right candidate

feature and its development as the successful OSS products are evolving products which will be always ready to address the need of its users/community at all given time.

6.4.8 Summary of OSSD Process - Stages and Activities

The proposed OSSD process has six development stages. Specific activities pertaining to each stage are suggested in the proposed process. In total, there are 18 important activities that are suggested. These are summarised in Table 6.6 for each development stage for a quick review.

Development Stages	Suggested Important Activities
Feature selection stage (single-phase stage with iterations)	Develop and further review the 'feature requirement document' before they are used for feature selection.
	Entire community should use the feature requirement document and also take part in selecting the right candidate feature for development.
	All selected features should be added to the development roadmap.
Requirement specification stage (single-phase stage with iterations)	Develop 'requirement specification document' and refine it in 'n' iterations, until widely accepted.
	Identifying the developers/team of developers who would work on the selected feature.
	Verify and freeze the requirement specification document for the latest product release.
Design specification stage (single-phase stage with iterations)	Develop design specification document for the selected feature that should satisfy the requirements.
	Update the design document based on the core teams feedback/suggestions (wherever necessary) and then publish it to the entire community.
Implementation stage (single-phase stage with iterations)	Develop a brief implementation document.
	Implement the selected feature based on design specification.
	Develop unit testing strategy.
Software testing stage (single-phase stage)	Develop and verify initial and crucial test cases for the OSS community.
	Test the locally released OSS feature.
	Report the bugs encountered and fix the same.

Integration and release stage (single-phase stage)	Verifying the list of features under developed roadmap and making sure that they are developed and tested.
	Develop release roadmap before the actual release
	Verify the implemented feature before integration and release.
	Update the release roadmap list if required; followed by integration and release of the OSS feature as a part of main OSS product.

Table 6.6 Important activities suggested for all stages of OSSD process

6.5 Summary

This chapter began with a brief comparison of the results of the activity flow diagram along with DEMO methodology results for all three OS e-learning systems. This was followed by a brief description of ISO/IEC 12207:2008 standard and a mapping between the standard and the outcomes achieved by the three OS e-learning systems. Subsequently, this chapter presented the proposed 'generalised OSSD process' in detail. The proposed process identified six development stages and its order of execution, along with the corresponding iteration pattern. Further, the major activities for each development stage were suggested along with a detailed discussion. The next step is the validation of the proposed OSSD process, which is explained in detail in the next chapter.

7. Validation of Proposed OSSD Process

7.1 Introduction

Any new software development process that has been proposed needs to be first validated in order to ensure that it is complete and acceptable. Hence, this chapter describes the validation of the proposed OSSD process. The validation approach is explained in detail along with its results and inference. In addition, this chapter addresses the feedbacks and comments received from the experts.

The proposed OSSD process can be validated in many different ways. The first technique that was considered was to handover the proposed OSSD process to the OS development community and develop an OSS feature for an e-learning system. Similarly, the second technique that was considered was to develop an OSS feature for an e-learning system in an academic environment. Both these techniques had the advantage of precisely pointing out the advantages and the drawbacks of the OSSD process. In fact, handing it over to the OSS community would have provided a very clear picture on the practical issues faced during the development of OSS feature.

However, the main drawback of both these techniques was the time constraint. It would take considerable time for the OS community/academic researchers to develop a new OS feature based on the proposed OSSD process; and then provide their suggestions and feedback. Further, since the OS development community is usually geographically distributed, it would require me to be personally involved in the development of the OSS feature, in order to meaningfully evaluate the results of the proposed OSSD process. However, this was again not possible because of the time constraint.

A third technique, known as 'expert review' method was therefore considered in this research work (Vredenburg, *et al*., 2002). It is a simple yet reliable approach with an added advantage of quick turn-around time. It is a well-

known approach in computing (Budgen, *et al*., 2008), specifically in software process area (Dyba, 2003). It is also described as an "Evaluation method" (Holz, *et al.*, 2006). Importantly, the 'expert review' method is seen as an ad-hoc method used by one or more experts for evaluation (Molich and Jeffries, 2003). Therefore, in this research, the OSS experts were requested to review the proposed OSSD process and give their feedback. Based on their reviews/feedback, the proposed 'generalised OSSD process' was then further improved.

7.2 Expert Review Approach

The expert review approach can be divided into six different phases as seen in Fig. 7.1. In the first phase, the experts were identified, taking into account three key criteria. These are:

- The experts should have sufficient knowledge about the various software development processes and models.
- The experts should actively participate or should have actively participated in the OSS development.
- The experts should have a good knowledge about how the OSS products are developed as a whole in an OSS development environment

While selecting the experts, preference was given to those people who had prior experience in OS e-learning system development. Once the experts were identified, individual request were sent to them along with a two-page document. This document briefly explained the back ground of this research, the expected outcome, the validation process in order to provide them a fair idea of what is expected from them. Depending on the received feedback and the willingness to serve as reviewers, three experts were selected for the validation process. The proposed OSSD process was then e-mailed to them so that they could study and examine it well in advance.

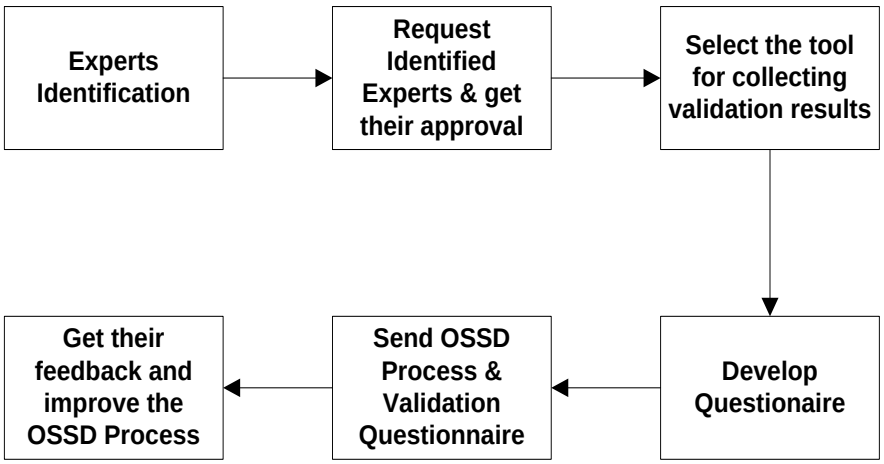

Fig. 7.1 Validation procedure

The next task for validation was to select an appropriate instrument/tool that could be used as a means to collect the validation results. There are various tools like diary method, interviews (online/telephonic/personal/mail), questionnaire, observation, etc. However, the questionnaire based approach was selected for this research work. This is because of the following three reasons:

- The experts were geographically distributed. Hence, having a questionnaire-based approach enabled them to give their feedback at their own time and was not confined to do in a particular time.
- The questionnaire-based approach formed a direct basis where they would have to answer specific questions with regard to the proposed OSSD process. This would enable us to streamline the received feedback; and improve the process accordingly. Further, this questionnaire facilitate in collecting detailed information as compared to interview/paper based surveys (constraint on time and length/pages of questionnaire). In addition having a questionnaire helps in structuring the questions especially when it has many branches, as in this work.
- Implementing a questionnaire was quick and cheap as compared to other methods (Munn and Drever, 1990; Basili, *et al.*, 1998).

Section 7.3 will present in detail the various aspects of the questionnaire and how it was being developed. Further, once the questionnaire was developed, it

was sent to the OS experts for validation. They were initially given two weeks time for completing the questionnaire and submitting the same. In case of any further delay, a reminder e-mail was sent to them every week until they responded to the e-mail or submits the questionnaire. Also, the experts were free to ask any doubts or concerns with regard to either the questionnaire or the proposed process itself. This helped in collecting the information as accurately as possible. Once the experts completed the review and clicked on the 'submit' button, the results were emailed to me and also, a copy of their feedback was saved for any future record. The results were then evaluated; and based on this evaluation; the proposed OSSD process was improved. The next section presents the format and the structure of the questionnaire for validation purpose.

7.3 Validation Questionnaire

The validation questionnaire was developed as a web-based questionnaire. The tool used to create the questionnaire was a web-service provided by 'JotForm' and can be accessed from www.jotform.com. The questionnaire was structured into different sections based on various development stages of the proposed OSSD process. This made it possible to collect and analyse the results easily for each individual development stage. The questionnaire started with a section dedicated for collecting personal/background information about each of the experts. This helped in identifying the experts. Each of the sections had a title, the instructions to answer the questions and followed by the question itself. At the end of each section, the experts were provided a choice of either going to the next section or go back to the previous section, in order to modify any details/information provided. Importantly, the experts had to finish all the sections and only then could submit their responses.

The questionnaire was composed of both close-ended questions like 'Yes/No'; 'Multiple choice' questions and at the same time, also had open-ended questions. The motive was to get their response on each and every development stage and also about the over all process. In-addition, the experts could provide their comments for each close-ended question. This helped in

getting a detailed feedback from them. The word limit was restricted to 100 words for close-ended questions while it was 500 words for open-ended questions. Importantly, all questions were mandatory. Hence, the experts had to answer all questions in each section, though providing comments were kept optional. For further details, please see *Appendix B*, which includes the entire questionnaire that was used by the expert reviewers for validation.

The final three phases of validation shown in Fig. 7.1 comprised of six tasks and is shown in Fig. 7.2.

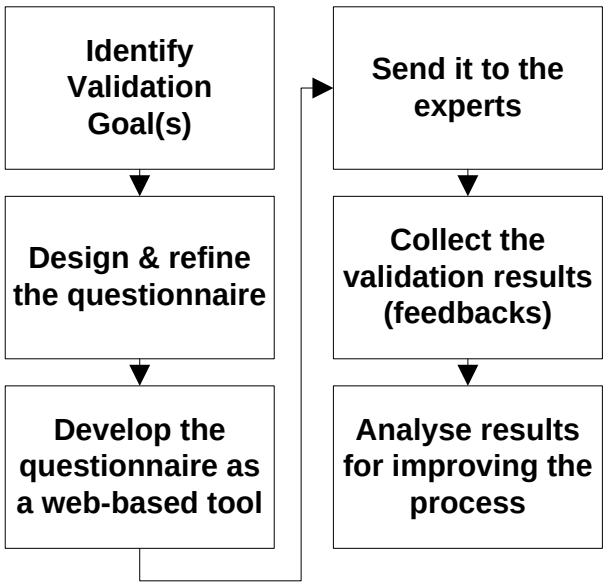

Fig. 7.2 Development and usage of the questionnaire

Before preparing a questionnaire for validating the OSSD process, it is important to understand the need and the aims of such a process. Hence, as a first task, the validation goals and aims were identified. The aim of validation is four fold and is listed below.

- Verify whether the proposed OSSD process is feasible to be used in an OS development environment.
- Verify whether the proposed OSSD process is 'complete' with all the required stages of development.
- Get the experts feedback on the proposed process so that it can be improved wherever required.

- Identify the concerns of the OS experts in using the process for developing an OSS product in an OS environment.

Once the goals have been made clear, the questionnaire was structured, developed and refined accordingly. The questionnaire was sent to three experts for validation via e-mail with a personalised request, along with a web-link. The professional experience of the three experts could be seen from Table 7.1. The first two experts had worked in OSS development for 4 years and 3 years respectively while the third expert had been working on OSS development as a developer for less than a year. However, all the experts had sufficient knowledge about various software development processes and also were well-informed about the OSS development practices. Notably, all three experts could directly access the questionnaire using the web-link. Further, the experts validated the proposed OSSD process and submitted their comments/feedback using the online tool.

Expert(s)	EXPERT 1	EXPERT 2	EXPERT 3
Years of experience	4	3	0.6
Number of OSS projects worked on	2	2	1
Knowledge on OSS Development Practises	Proficient	Proficient	Proficient

Table 7.1 Information on the expert's professional experience

The questions asked in the questionnaire are available in *Appendix C* which also includes the expert's feedback/answers in a tabular representation. Particularly, in case of objective-type questions; if the experts had contradicting answers for a particular question, then the decision on OSSD process alteration was done based on their comments.

7.4 Result Interpretation and Process Amendment

This section presents the feedback received from the experts for each of the development stages along with the corresponding comments/opinion given by the experts for improving the process. The experts gave their

opinion/comments for improving few stages which they were not completely satisfied.

The results are discussed for each of the development stages, taking into account the feedback received from individual experts. For each stage, the individual experts' feedback/concern is reported along with the reason for concern. Further, a response is provided for each concern raised by the experts. Finally, the proposed OSSD process is amended depending on the feedback/ comments and is presented accordingly. These amendments include - an alteration to the development stage as a whole, alteration with regard to the activities carried out or the frequency with which the development stage/ activities of a development stage are preformed, wherever necessary.

7.4.1 Feature Selection

7.4.1.1 Expert 1:

Feedback/Concern: According to the first expert, the 'feature selection' stage was not realistic, especially because the proposed process suggests that the entire community should participate in feature selecting before it is developed.

Reason for Concern: Going through the feedback of expert 1, it could be made out that that the expert had compared the proposed process with the OSS that the expert had been working with. His major concern was that the features were developed only when funding was approved.

Response to the Concern: The issue of funding approval is not true for most of the OSS products, as many of them are developed for addressing the immediate need and hence, the OS developer does not wait for funding. The OS community mainly invests time and effort in developing a feature. Hence, the expert's concern would not be significant, especially if the OS feature is of high importance.

7.4.1.2 Expert 2:

Feedback/Concern: According to the second expert, there should not be an over dependence on the core team.

Reason for Concern: According to this expert, the proposed way of feature selection might be appropriate for small and very crucial features but might not suit a very large feature. In fact, this might make the core team to spend lot of time in feature selection.

Response to the Concern: Although he points out the core team cannot be given too much authority, he also mentions that only the core team is capable and hence, should be responsible for selecting and finalising the feature for development. This was a self-contradicting statement.

7.4.1.3 Expert 3:

The third expert did not suggest any changes to this stage.

7.4.1.4 Amendment in Proposed OSSD Process:

The feature selection process was explicitly divided into two phases, based on the feedback of the second expert. In the first phase, the OSS community would take a lead in selecting the feature for development (for instance, voting and suggestions). In the second phase, the core team would take the final call. This would provide equal opportunity to both the core team and the community in selecting the feature for development and would not make anyone in the OSS community to feel less important.

7.4.2 Requirement Specification Stage

7.4.2.1 Expert 1 and Expert 3:

The first and third expert did not suggest any changes to this stage.

7.4.2.2 Expert 2:

Feedback/Concern: Freezing the feature requirement specification is too strict for an OSS environment.

Reason for Concern: Freezing the feature requirement might result in the feature not being improved/ elaborated. Further, in an OSS community, the

document should be kept open even during its development to the entire community.

Response to the Concern: The proposed OSSD process suggests freezing the feature requirement specification only for the immediately occurring feature release. Also, the core team can freeze the requirement only if the entire community or at least the working team agrees on the proposed feature specifications. In addition, it is important to understand that the proposed OSSD process visualises the OSS development as a continuously evolving process. Hence, freezing the requirement would assist the community to focus on developing the feature. Notably, if someone requires the feature to be elaborated later, the OSS community would allow them to raise a request for new development.

With regard to keeping the requirement document open at all stages to the entire community, it is not a major issue and could be decided by the respective OS community. However, an important point to be noted is that publishing an incomplete, unclear document might create confusion among the entire community. This was the main reason why the proposed OSSD process suggested making the document open to the community once the initial version is developed. The community could then provide suggestions/ recommendations for improving the same.

7.4.3 Design Specification Stage

7.4.3.1 Expert 1:

Feedback/Concern: The first expert again had a comment with regard to arranging the finances for the design activities.

Response to the Concern: As previously mentioned; not all the OSS developing community, aim to have finances arranged for developing an OSS feature.

7.4.3.2 Expert 2 and 3:

Feedback/Concern: Allowing the entire community to validate the design specification may be too tedious. Hence, only the core team should be given full authority to validate the design process.

Response to the Concern: The proposed OSSD process suggested a similar approach, with the additional prospect for the OS community to give their suggestions with regard to improving the design. This was because; there could be occasions where the unseen problems could be identified by third parties.

7.4.3.3 Amendment in Proposed OSSD Process: Having the design specification validated by the entire OSS community is not mandatory but would be highly recommended. The decision could be taken appropriately by the core team depending upon the feature under development.

7.4.4 Implementation Stage

All three experts stressed and acknowledged the importance of the unit-testing strategy adopted in this stage, in the proposed OSSD process.

7.4.4.1 Expert 1 and Expert 3:

The first and third expert did not suggest any changes to this stage.

7.4.4.2 Expert 2:

Feedback/Concern: The proposed OSSD process should emphasis on publishing the inter-mediatory milestones that are usually achieved during the software implementation stage.

Response to the Concern: This is an important point to be considered, as the proposed process only suggested periodic update about development stage.

7.4.4.3 Amendment in Proposed OSSD Process:

The proposed OSSD process is modified such that, as and when the community achieves a milestone (even if it is an intermediary one); it should

be published to the entire community. This would have an added advantage that the development progress would be periodically revealed to all stake holders.

7.4.5 Software Testing Stage

7.4.5.1 Expert 1:

The first expert did not raise any major concern and did not suggest any changes to this stage.

7.4.5.2 Expert 2 and 3:

Feedback/Concern: Providing explicit criteria for testing before the OS community start the actual testing might limit the overall testing scenario. Further, the community as a whole should use a bug tracking system.

Reason for Concern: The tester might not look beyond the test case that is already provided. The bug tracking system would ensure that all the bugs are fixed.

Response to the Concern: Providing an explicit criterion for testing would enable the new comers is the OS community to kick start their testing expedition.

7.4.5.3 Amendment in Proposed OSSD Process:

The proposed OSSD process would stress the OSS community to incorporate a bug-tracking system. Further, the process suggests performing testing activities during various stages of development especially when multiple components interact with each other.

7.4.6 Integration and Release Stage

One of the experts overlooked the necessity of verifying the software unit produced with the requirement and design before release. On the other hand,

the other two experts stressed the importance of verifying it against the requirement and design.

Hence, due to the majority of experts (two out of three) are in favour of verification, the verification stage was maintained as originally proposed.

7.4.7 Summarizing the Process Amendment

All three OSS experts were fairly content with the proposed OSSD process and also provided their feedbacks/comments for each individual development stage. Out of six development stages, significant concerns were raised for three stages. They were Feature Selection Stage, Design Specification Stage and Software Testing Stage. Further, there was a concern with regard to the Implementation Stage. These stages were amended in order to address the reviewers concerns and to improve the overall process. Table 7.2 lists the amendments with respect to the important activities carried out in major development stages.

Development Stages	Amendment(s) to the Process Activities
Feature selection stage (Two-phase stage with iterations)	Feature selection is divided into two phases. (i) In the first phase, community selects feature for development. (ii) In the second phase, core team takes the final call on all the selected features before adding it to the roadmap.
Design specification stage (single-phase stage with iterations)	Recommends that the entire community take part in validation of the design documents but is optional.
Implementation stage (single-phase stage with iterations)	(i) Publish all the inter-mediatory milestones to the entire community. (ii) Publish updates periodically regarding the development.
Software testing stage (single-phase stage)	(i) Stresses on the usage of bug-tracking systems. (ii) Recommends performing testing/review activities at different stages of development rather than putting it as a last one.

Table 7.2 Amendments to process activities

The amendment for Feature Selection stage includes dividing this stage into two sub-stages. In the first sub-stage, the OSS community would select the feature for development. In the second sub-stage, the core team would decide on the features based on the community's choice. For the Design Specification Stage, the proposed process recommends that the entire community take part in validation of the design documents. However it is not mandatory and could be decided by the core team based on the feature under development. The amendment to the Implementation Stage includes publishing the inter-mediatory milestones along with the periodic updates. Finally for the Software Testing Stage, the proposed process stresses on using bug-tracking systems. This would help in tracking all the important bugs so that it can be fixed before the release. Further, the proposed OSSD process suggests performing testing activities during various stages of development, especially when multiple components interact with each other.

Notably, a concern was raised by one of the expert on the Requirement Specification Stage and Integration and Release Stage. However the reason given by the expert was either inconsequential and/or biased. Therefore, no changes were recommended to these two development stages.

7.5 Summary

In this chapter, various validation approaches appropriate for this research work were discussed. Further, an explanation was provided on the selected 'expert review' approach followed by a detailed explanation on the questionnaire-based technique that was selected to realize the 'expert review' approach. The feedback from the experts were then interpreted along with their major concerns and the reasons for those concerns, Importantly, all the major concerns were addressed and the proposed generalized OSSD process was improved as when and where required. The next chapter would conclude the thesis and particularly, revisit the research questions. Notably, it would point out the advantages and limitations of this work, along with the potential future research direction.

8. Conclusion and Discussion

8.1 Introduction

This is the final chapter of this thesis. It provides a brief insight to the research work by revisiting the various research questions formulated at the beginning of this thesis. The implication of the proposed OSSD process is then described with respect to OS e-learning system development. Further, the limitations of this work are presented along with the possible future work pertaining to this research.

8.2 Research Insights - Revisiting Research Questions

The goal of this research work was to develop a generalised OSSD process that would enable the OS community to work together and develop more efficient OS e-learning systems. The fundamental question that defined the development of new OSSD process was, "*What approach should be followed in order to design a generalized OSSD process*?" Answering this broad question required it to be broken down into four basic yet important research questions. These are revisited in order to ensure that answers to these research questions were considered completely while developing the new OSSD process.

RQ1: *What are the current development practices followed by the OS e-learning product development communities?*

To begin with, different OS e-learning systems were sampled and three popularly used OS e-learning systems were selected as the basis. These include, Moodle, ILIAS and Dokeos. Subsequently, the development practices were analysed for each of the three OS e-learning systems which resulted in deeper understanding of their current practices.

RQ2: *How should the current development practices be assessed in order to design a generalised OSSD process?*

Each of the three selected OS e-learning systems was modelled using activity flow diagrams. However, it had two major drawbacks. Firstly, this representation did not identify exactly which actors were involved in carrying out a particular activity/ task. Secondly, the activity flow representation did not specify the outcome of a particular activity. In order to overcome these drawbacks, DEMO methodology was adopted and DEMO models were created for each of the three OS e-learning systems. These models facilitated in identifying the following:

- Various implicit and explicit development stages
- Various activities carried out for each development stage
- Different actors involved in the development activities and
- Outcome of each development activities.

RQ3: *How is the OSSD process designed based on previous findings?*

The activity flow diagram and the DEMO models assisted in identifying the output created by each activity. However, consolidating these results alone did not help in developing the generalized OSSD process. This is because, various e-learning communities followed different approaches towards software development. Thereby, they differed in the execution of various development stages and the corresponding activities. All of them produced a mix of various other outcomes which made it difficult to generalise. Therefore, a well defined standard, ISO/IEC 12207:2008 was used as a foundation tool in designing and proposing the generalised OSSD process. The ISO/IEC 12207:2008 standard provided a list of all desirable outcomes that one could produce when executing the lower level process. These outcomes were comparable with the outcomes identified using DEMO models. Subsequently, the result of DEMO models and activity flow diagrams were used in conjunction with the standard in order to develop the proposed OSSD process.

RQ4: *What approach should be followed to assess the proposed process and also to evaluate results of the appraisal?*

This question could also be written as, "*How should the proposed OSSD process be evaluated*?" In this research work, considerable deliberation was done with regard to selecting an appropriate validation mechanism. Though the proposed OSSD process was designed mainly for the OS development community, in this case, it has been developed as a Master's research work carried out in an academic environment. Hence, due to the time and resource constraints, the proposed OSSD process was validated using an '*expert review*' method. Accordingly, the proposed generalised OSSD process was presented to three external experts along with a detailed web-based questionnaire. Based on their feedback, the results were then analyzed and the proposed OSSD process was modified accordingly wherever required.

8.3 Implication

The proposed OSSD process described the different stages of development and their ordering, the frequency with which each development stage is executed and notably, the important activities involved in each stage. There were three major issues identified with respect to OS e-learning systems – issues with respect to software design, the user requirement not being addressed sufficiently and lack of proper documentation (Chapter 2). These issues are addressed in the proposed OSSD process and are mentioned below:

- The requirements should be verified not only by the core team but also by the entire community. In fact, this should start with the feature selection itself where the entire community should be encouraged to select the features based on the initial description provided to them.
- Implementing a detailed design stage should be mandatory, wherein the design documents would be produced by the development team. This would enable the core team to access the feature to be developed and also provide a clear picture of the feature, thereby advancing easily from design to development. Further, having the design document would assist the core team to verify whether the design had completely satisfied the user requirements before development.

- Documents should be developed at various stages, starting from feature selection stage till integration and release stage. However, in order to reduce the amount of documentation, the proposed process suggested keeping the documents brief, while encouraging the community to follow their own template in developing these documents.

8.4 Research Outcomes

This research work began with understanding the current development practices of three major OS e-learning systems (Moodle, ILIAS and Dokeos). An in-depth analysis (comparable with case studies) of the three e-learning systems was then performed. The result of this analysis was then presented using *activity flow diagrams*. These activity flow diagrams identified the implicit and explicit stages of development followed by the OS community. This is the first and one of the quintessential inputs towards the proposed OSSD process.

In order to build an abstract model independent of the development techniques, the DEMO methodology was adopted to model the development activities of Moodle, ILIAS and Dokeos. This identified both the outcome of each development activity and also the persons responsible for bringing such outcomes into existence. This type of modelling has not been done before for interpreting the development practices followed by the OS e-learning community. Hence, the DEMO models for OS e-learning systems are an important research contribution and form a crucial input towards developing a generalised OSSD process for e-learning systems.

Importantly, a generalized OSSD process has been proposed, taking into consideration the ISO/IEC 12207:2008 standard. The proposed process has six explicit development stages. These are:

1. Feature selection stage
2. Requirement specification stage
3. Design specification stage
4. Implementation stage

5. Software testing stage
6. Integration and release stage

It should be noted, that the *feature selection stage* mentioned in the proposed OSSD process is unique to the OSSD and has not been explicitly considered before. Significantly, specific activities pertaining to each stage have also been suggested for the proposed OSSD process. In total, 18 important activities have been suggested across the six development stages. However, the proposed OSSD process does not specify the techniques to be used for performing the various activities and keeps it flexible for the community to decide.

8.5 Limitations

It is important to understand that even though the proposed OSSD process has undergone iterations based on the feedback received from external OSS experts, the current form of the proposed OSSD process does have some limitations. Some of the limitations are:

- The proposed generalised OSSD process has been designed based on the comparison of current development practises followed in OS e-learning systems development and the standard's prescription. Subsequently, the proposed process highlights only the major activities under each development stage and does not list all the activities that have to be performed during development of OS e-learning system.
- The success of an e-learning system depends to a large extent on the ease-of-use/usability. This is because e-learning systems are used simultaneously by different users with varied skill sets. For instance, an e-learning system could be used simultaneously by a student with no prior experience, by a teacher experienced in developing learning contents; and also by an administrator who might be good in managing the system as a whole but might not have experience in developing the content itself. However, the proposed OSSD process does not explore the usability aspect. Though its importance is understood, due to the time constraint, the usability aspect is not explored in this research work.

- The proposed OSSD process provided a high level, abstract process for OSS community. But it did not investigate extensively on other process areas and activities. Specifically, it did not suggest precise activities pertaining to any of the six development stages that could enable the community to improve the product's usability and there by the product quality.
- In this research work, three different OS e-learning systems and their current development practices were considered for developing a generalized OSSD process for e-learning systems. However, other popular OSS product's development practises (e.g., OS web-browser) were not considered for developing the proposed OSSD process.
- With regard to validation, when the experts were questioned about the proposed process, they were inclined to compare it with the development practices followed in their current OSS project. This hinders obtaining a completely unbiased response.
- In this research, the validation was carried out by seeking reviews from three OS experts. However, it is slightly debatable whether three reviews are sufficient for improving an OSSD process. Further, the best method of validation would be to follow the process for developing an OS e-learning system feature. However, this could not be done in this work due to time and resource constraints.

Some of the above mentioned limitations could be overcome and the proposed OSSD process could be further adapted in the future.

8.6 Future Work

The proposed OSSD process is an initial, generalised, exemplification of the process that could be followed in developing an OSS product. The software process could be further adapted depending on the need and necessity of a particular OSS community and/or the feature developed by them. Some of the notable directions in which this research work could be extended are as follows:

- **Usability:** It is a non-functional requirement of a software development process. Also, it is an extremely important qualitative attribute that assesses the ease-of-use of interactive software like an e-learning system. This in fact has a direct influence towards the success of the e-learning system. In this regard, it should be noted that the OS community could be easily motivated to follow software development process as compared to motivating them to follow usability guidelines (Twidale and Nichols, 2005). Hence, a notable future direction would be to work on integrating the usability guidelines into the proposed OSSD process. This is quite a challenging task in itself. The two big questions that need to be answered here would be:
  a. How to consolidate the usability guidelines specifically for e-learning system?
  b. What aspects are to be considered in consolidating the usability guidelines for OS e-learning system?

- **Inclusion of all Tasks and Activities:** The generalised OSSD process could be further elaborated such that it lists all required activities that are to be performed by each of the OSS community during the different stages of development. Further, the process could include all specific tasks and activities pertaining to usability and all other quality attributes. This would not only help in improving the product quality but also enable the users to effectively use the product.

- **Inclusion of Other OSS development Practices:** In order to develop an efficient OSSD process, the best developmental practises of different OSS products needs to be incorporated. Currently, the best practices from popular OS e-learning systems are alone considered for developing the generalised OSSD process. However, other popular OSS product's (Apache, Mozilla, Linux, etc.) development practises should also be considered and their best practices should be incorporated to enhance the proposed generalised OSSD process.

- **Comprehensive Validation**: A best approach to validate the proposed OSSD process is to develop different OS features based on the proposed

process. This approach would provide a much clearer picture, aide in understanding the inherent weakness of the proposed process, provide a deeper understanding of real world issues and importantly, identify the areas where the proposed process would provide significant benefits.

References

Albani, A. and Dietz, J.L.G. 2011. Enterprise ontology based development of information systems. *Internet and Enterprise Management*. 7(1). pp. 41-63.

Allen, E.I. and Seaman, J. 2008. Staying the Course. *Online Education in United States*. United States of America: Babson Survey Research Group.

Allen, E.I. and Seaman, J. 2010. Class Differences. *Online Education in United States*. United States of America: Babson Survey Research Group.

Allen, E.I. and Seaman, J. 2011. Going the Distance. *Online Education in United States*. United States of America: Babson Survey Research Group and Quahog Research Group, LLC.

Balogh, A. and Budai, A. 2009. Organizational Integration of ILIAS Services at Dennis Gabor Applied University. *IN: 8th International ILIAS Conference, Nov 12-13, 2009*.

Boehm, B.W. 1988. A Spiral Model of Software Development and Enhancement. *Journal of Computer, Publisher: IEEE Computer Society.* 21(5). pp. 61-72.

Basili, V., Shull, F., Lanubile, F. 1998. Using experiments to build a body of knowledge. *IEEE Transaction on Software Engineering.* 25(4). pp. 456–474.

Basili, V. R. and Lonchamp, J. 2005. Open source software development process modelling *IN:* Acuña, S. T. and Juristo, N. (eds.). *Software Process Modelling, 10*. US: Springer. pp. 29-64.

Bernard, R.M., Abrami, P.C., Wade, C.A. 2007. A Summary of "Review of E-Learning in-Canada A Rough Sketch of the Evidence, Gaps,and Promising Directions". *Horizons policy research initiative*. 9(3), pp.32-37. [Online] Available from: http://www.horizons.gc.ca/doclib/HOR_v9n3_200702_e.pdf, [Last accessed 02 July 2012].

Boufford, B. 2004. Issues with Open Source Software Development. *Community discussion: Social forum* [online], 21 April. Available from: http://moodle.org/mod/forum/discuss.php?d=7208. [Last accessed: 23 July 2011].

Budgen, D., Turner, M., Brereton, P. and Kitchenham, B. 2008. Using Mapping Studies in Software Engineering. *IN 20th Annual Workshop. Psychology of Programming Interest Groups, Lancaster, UK.*

Cantoni, V., Cellario, M. and Porta. M. 2004. Perspectives and challenges in e-learning: towardsnatural interaction paradigms. *Visual Languages and Computing*. 15(2004). pp. 333–345.

Cemal Nat, M., Dastbaz, M. & Bacon, L. 2008. Research and Design Challenges for Developing Personalised eLearning Systems. *IN: C. Bonk et al. (Eds.), Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2008*. pp. 2536-2542.

Clarke, P. and O'Connor, R. 2010. Towards the identification of the influence of SPI on the successful evolution of software SMEs *IN: Dawson, R., Ross, M., Staples, G. (eds.). Proceedings of the 18th International Conference on Software Quality Management.* pp. 29 – 40.

Curtis, B., Kellner, M.I. and Over, J. 1992. Process Modelling. *Communications of ACM*, 35(9). pp.75–90.

Devine, J. 2008. Difference between Open, Free and Closed Source Software. Ezine article, *Computers and Technology: Software* [online]. Avalibale from: http://ezinearticles.com/?Differences-Between-Open,-Free,-and-Closed-Source-Software&id=1329290, [Last accessed on 4th July 2011].

Dietz, J.L.G. 2006. Enterprise Ontology. *Theory and Methodology*. Springer-Verlag, Berlin Heidelberg.

Dokeos. 2012. *A 4 Million User Community* [Online]. Available from: http://dokeos.com/en/community.php

Donovan, S. 1994. Patent, copyright and trade Secret protection for software. *IN: IEEE Potentials*. 13(3). pp. 20-24.

Dyba, T. 2003. Factors of software process improvement success in small and large organizations: an empirical study in the scandinavian context. *IN: Proceedings of the 9th European Software Engineering Conference held jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (ESEC/FSE-11).

Ellis, R.K. 2009. Field Guide to Learning Management Systems. *American Society for Training & Development - Learning Circuits* [online]. Available from: http://www.astd.org/NR/rdonlyres/12ECDB99-3B91-403E-9B15-7E597444645D/ 23395/LMS_fieldguide_20091.pdf. [Last accessed 10 July 2011].

Feller, J. and Fitzgerald, B. 2002. *Understand Open Source Software Development*. London: Pearson Education Limited.

Fuggetta, A. 2000. Software process: A Roadmap *IN: International Conference on Software Engineering: Proceedings of the Conference on Future Software Engineering*, Ireland. pp.25-34.

Fuggetta, A. 2003. Open source software – an evaluation. *Journal of Systems and Software*. 66(1), pp. 77-90.

Ghosh, R.A., Glott, R., Krieger and Robles, G. 2002. Free/Libre and Open Source Software: Survey and Study. *International Institute of Infonomics University of Maastricht, The Netherlands*.

Glosiene, A., & Manzuch, Z. 2004. Usability of ICT-Based Systems, State-of-the-Art Review. *Calimera Project Deliverable 9* [online]. Available from: www.calimera.org. [Last accessed on: 3rd July 2011].

Gruber, T. 1994. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *IJHCS*. 43(5/6): 907-928.

Holz, H.J., Applin, A., Haberman, B., Joyce, D., Purchase, H. and Reed, C. 2006. Research methods in computing: what are they, and how should we teach them?. *SIGCSE Bull*. 38(4). pp. 96-114.

Humphrey, W.S. 1988. Characterizing the Software Process: A Maturity Framework. *IEEE Software.* 5(2). pp. 73-79.

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.

ILIAS. 2012. *Open Source E-Learning System* [Online]. Available from: http://www.ilias.de/

International Standard 2008. *ISO/IEC 12207:2008 Systems and software engineering —Software life cycle processes*. Second Edition (01/02/2008) [Online]. Available from: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4475822

Jensen, C. and Scacchi, W. 2006. Experiences in Discovering, Modelling, and Re-enacting Open Source Software Development Processes. *IN: Li, M., Boehm, B. and Osterweil, L. (eds), Unifying the Software Process Spectrum*. Heidelber:Springer Berlin. pp. 449-462.

Jensen, C., Scacchi, W. 2008. Reference model based open source software process discovery. *Technical report*. [online]. Available from: http://rotterdam.ics.uci.edu/ papers/jensen-scacchi-oss07/Jensen-Scacchi-OSS2007.pdf.[Last accessed on 10 July 2010].

Krogh, G.V. and Hippel, E.V. 2006. The promise of research on open source software. *Management science*. 52(7). pp. 975-983

Kruchten, P. 2000. *Rational Unified Process. Reading*. Addison Wesley publication.

Kruse K. 2009. *The Benefits and Drawbacks of e-Learning* [online].Available from: http://e-learningguru.com. [Last accessed 9th October 2010].

Liaw, S. 2008. Investigating students' perceived satisfaction, behavioural intention, and electiveness of e-learning: A case study of the Blackboard system. *IN: Computers & Education*. Vol.51(2008). pp. 864–873.

Lonchamp, J. 2005. Open source software development process modelling. *IN: Software Process Modeling, International Series in Software Engineering.* Vol.10. pp. 29–64.

Milojevic, Z.M. 2011. Web based training as a factor of quality improvement for applying CNC technology. *IN: 5th International Quality Conference – Quality Research.* pp. 553-564.

Mingshu Li, Barry W. Boehm, Leon J. Osterweil. 2006. Unifying the Software Process Spectrum. *Journal of Software*. 17(4). pp.649−657.

Molich, R. and Jeffries, R. 2003. Comparative Expert Reviews. *IN: CHI '03 extended abstracts*, Ft. Lauderdale, FL. pp. 1060–1061.

Moodle. 2012. Moodle Statistics[Online]. Available from: http://moodle.org/stats/ [Last Accessed June 2012].

Munn, P. and Drever, E. 1990. Using questionnaires in small-scale research: a teacher's guide. *Scottish Council for Research in Education.*

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K. and Ye, Y. 2002. Evolution Pattern of Open-Source Software Systems and Communities. *IN: Proceeding of Int.Workshop on Principles of Software Evolution* (IWPSE 2002), Orland, Florida. pp. 76-85.

Oguzor, N.S. 2011. E-learning technologies and adult education in Nigeria. *Educational Research and Reviews*. 6(4). pp. 347-349.

Open Source Initiative. 2012. *Mission.* [Online]. Available from: http://opensource.org/

Paulson, J.W. 2004. An empirical study of open-source and closed-source software products. *IEEE transactions on software engineering*. 30(4). pp 246-256.

Paulson, J. W., Succi, G., and Eberlein, A. 2004. An Empirical Study of Open-Source and Closed-Source Software Products. *IEEE Transaction on Software Engineering*. 30(4). pp.246-256.

Raymond, E.S. 1998. The Cathedral and Bazaar. *First Monday, Peer Reviewed Journal*. 3(3). [online]. Available from: http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/578/499, [Last accessed: 04th July 2011].

Scacchi , W. 2001. Software Development Practices in Open Source Software development communities: A Comparative Case Study Making Sense of the Bazaar. *IN: First Workshop on Open Source Software Engineering, 23rd International Conference on Software Engineering.* [online]. Available from: http://opensource.ucc.ie/icse2001/scacchi.pdf. [Last accessed 05 July 2011].

Scacchi, W., 2003. Issues and experiences in modelling open source development. *3rd Workshop on Open Source Software Engineering*.

Scacchi, W., Feller, J., Fitzgerald, B., Hissam, S. and Lakhani, K. 2006. Understanding Free/Open Source Software Development Processes. *Software Process: Improvement and Practice*. 11(2). pp. 95-105. [online] Available from: http://onlinelibrary.wiley.com/doi/10.1002/spip.255/pdf. [Last accessed on 05 July 2011].

Scacchi, W. 2007. Free/Open Source Software Development: Recent Research Results and Methods. *Advances and Computers*. 69. pp. 243-295.

Selim. H.M. 2007. Critical success factors for e-learning acceptance: ConWrmatory factor models. *Computers & Education.* 49 (2007). pp.396–413.

Shea, R.H. 2002.E-learning today—As an industry shakes out, the survivors offer no-frills education for grown-ups.*U.S. News & World Report. October 28, 2002.*

Sommerville, I. 2004. *Software Engineering.* 7th ed. Published by Pearson Education: England.

Stamper, R.K. 1973. *Information in business and administrative systems.* B. T. Batsford: London and New York: Wiley.

Stockley D (2003). *E-learning Definition and Explanation*. ECampus.com.au.

Tavangarian D., Leypold M.E., Nölting K., Röser M.,Voigt, D. (2004). Is e-learning the Solution for Individual Learning? *Electronic Journal of e-Learning*. 2(2). pp. 273-280.

Twidale, M.B. and Nichols, D.M. 2005. Exploring usability discussion in open source development. *IN: Proceedings of the 38th Annual Hawaii International Conference on System Sciences,* HICSS'05.

Tyrrell, S. 2000. Many Dimensions of the Software Process. *The ACM Crossroad Student Magazine*, 6(4). pp.22-26.

Tysver, D.A. 2008. *Why Protect Software Through Patent*. [Online]. Available from: http://www.bitlaw.com/software-patent/why-patent.html

Vredenburg, K., Mao, J.Y., Smith, P.W. and Carey, T. 2002. A survey of user-centered design practice. *IN: Proc. of the SIGCHI Conference on Human Factors in Computing Systems:Changing Our World, Changing Ourselves* pp. 471-478.

Ye, Y. and Kishida, K. 2003. Towards the understanding of the motivation of open source software developers. *IN: Proceedings of 2003 International Conference on Software Engineering*, Oregon, pp 419-429, May 3-10.

Yong, J. 2008. Enhancing the Privacy of E-Learning Systems with Alias and Anonymity. *IN: Proceedings of CSCWD 2007, LNCS 5236.* pp 534-544.

Zahran S. 1994. The software process—what it is, and how to improve it. *IN: Proceedings of Software Quality Management II*. Vol.1: Managing Quality Systems, Computational Mechanics Publications: Glasgow, UK, 215–231.

Zhang, D., Zhao, J.L., Nunamaker, Jr., J.F. 2004. Can E-learning Replace classroom Learning?. *The Communications of ACM*. 47(5). pp. 75-79.

APPENDIX A

Benefits and Drawbacks of E-Learning Systems

In general, there are three classes of users for any e-learning system. This includes,

- Learner
- Instructor
- Organization intending to use and provide the e-learning system

It should be noted that the benefits and drawbacks of an e-learning system will be different for different classes of people.

A.1 Benefits

A.1.1 Benefit to the Learners

From the learner's perspective, there are several significant benefits of using an e-learning system. These include (Kruse, K. 2009; Rosenberg, M. 2009):

- On-demand availability of learning materials that enables the learner to learn anywhere and anytime.
- On-line learning materials enable the learner to learn at his/her own pace which not only leads to higher satisfaction but also assists the learner in achieving his/her learning objective, with significantly reduced stress.
- Interactively engage users, thereby creating inquisitiveness among the users to learn.
- Confidence building measure among the learners by providing them with quick reference materials.
- Decreased time to learn through an intelligent combination of different media formats and animations.

A.1.2 Benefits to the Instructor/Educator

The benefits of e-learning to the instructor are (Bates, A.W. 1997; Kruse, K. 2009).

- Assistance in improving access to education, training materials and support.
- Improvement in the overall quality of learning which benefits the instructor
- Reduction in the energy and cost of travelling (Fletcher, J.D. 1991)
- Training and knowledge transfer in very specific domains can be accomplished easily, efficiently and in a cost-effective manner. This is true in higher-educational institutions and also in industries/corporate world (Fletcher, J.D. 2009).
- Tedious and laborious mechanisms of learning like providing written materials, proofs, documentation, etc. can be automated.

Reduced cost is said to be one of the prominent factor in adopting e-learning systems and is seen not only in educational sector; but also in industrial level, e-learning is used extensively for training purposes.

A.1.3 Benefits to the Institution/Organization

The benefits hold to both the institution acquiring the e-learning system and also the organization that actually develops and provides the e-learning system (Kruse, K. 2009). The benefits are:

- Provide consistent, worldwide training to its employees/learners.
- Reduce delivery cycle time.
- Increase learner convenience.
- Reduce information overload.
- Improve tracking and
- Significantly lower expenses compared to multi-location class room coaching.

An important point to be noted from above is that, an e-learning system provides a clear and distinct benefit to each class of people.

A.2 Drawbacks

An e-learning system has few drawbacks as well, both due to technical and non-technical aspects. These are mentioned as follows:

A.2.1 Drawbacks to the Learners

The drawbacks of an e-learning system from the learner's perspective are: (Rosenberg, M. J. 2001; Kruse, K. 2009).

- Many learners, especially those who are not from the ICT background are techno-phobic, i.e., they either do not understand or are too awed by the technological component of the system design.
- Huge number of learners does not have access to adequate technological resources. This serves as a major hindrance to the potential learners.
- Reduced social and cultural interaction, which in-turn narrows the thinking ability of the learner.

A.2.2 Drawbacks to the Acquiring Institutions

- Acquiring an e-learning system requires huge upfront investments. This is usually a major bottleneck that prevents an institution in acquiring a high-quality learning system.
- The institution has to ensure that that the instructor or any 3rd party vendor produces sufficient learning content. This is a major task and requires huge investment in terms of time and energy.
- The technology and the infrastructure of the institution should be compatible with the requirements of the e-learning system.

Over the recent years, with wider acceptance of technology and with a greater understanding of the holistic benefits that could be accrued through e-learning, most of the drawbacks have been losing their importance, especially in developed countries and economies. However, the issues are still relevant and

cannot be ignored completely. Further, OS LMS, if and when effectively designed, could overcome several disadvantages. For instance, the upfront investments for using commercial e-learning system can be avoided by going for an OS e-learning systems. Continuing further, several individuals/organizations would switch/have been switching from CS to OS e-learning products if and when they find the e-learning product to be:

- Flexible
- Inclusive
- Authentic
- Relevant
- Effective and
- Globally accepted

APPENDIX B

B.1 OS Expert-Validation Questionnaire

B.1.1 Purpose of the Questionnaire for Validation

The purpose of this questionnaire is to validate the proposed OSSD process for developing OS e-learning systems. This questionnaire aims to extract information on the following:

- Whether the proposed OSSD process supports all necessary stages of development that are required for developing an OS e-learning system.
- Whether the developmental stages are in correct order.
- Whether the development stages iterated sufficiently.
- In the proposed OSSD process, some of the activities and their respective outcomes are made mandatory. The questionnaire aims to verify whether it is necessary and feasible in OS environment.

In addition to these, the experts are also requested to give:

- Details of any changes that they consider to be beneficial to the OSSD process.
- Critique on all the stages of development, highlighting both the negative and positive aspects of the OSSD process.

B.1.2 Structure of the Questionnaire

The questionnaire is developed as a web-form (using jot forms – www.jotforms.com) which the experts can access from anywhere and can submit it online once they are done with it. The questionnaire itself is divided into various sections for ease of analysis. It starts with expert's identification and their OS experience in order to identify their credentials. Then the questionnaire advances towards the validation of OSSD process where it is divided based on different stages of development.

Most of the questions are objective type questions where the experts are requested to select the answer(s). On an average, there are four objective type questions per section. The experts also have an option of providing comments/feedback for each of the questions and it can be of maximum 100 words.

There are few subjective type questions and its focus is to get the experts personal opinion/experiences, their critique on the OSSD process and further feedback. Maximum of 500 words is allowed for the same. Finally, the experts have to answer all of the questions and submit it once it is done.

B.1.3 Questionnaire

Expert Identification

First Name: _____

Last Name: _____

E-mail ID: _____

SECTION 1 - Background Details

[*Please answer all the questions.****]***

1. How many years of experience do u have in open source (OS) development?

_____ (in years).

2. For how long are you working in the current project?

_____ (in years).

3. What is your current project role?

- Project leader
- Developer
- QA analyst
- Others

Others (Please specify): _____

4. What are the activities do you currently work on? *(Please select all that applies)*

- Software requirement
- Design
- Coding & unit testing
- Testing/Integration
- Software quality assurance
- Process improvement

5. Do you follow any specific software development process?

- Yes
- No

If YES, please give details: _____

6. Do you have any previous experience in software process improvement?

- Yes
- No

If YES, Please brief on your role: _____

SECTION 2 – Software Practice – Feature / Requirement selection

[*Please answer all the questions. If you have answered "No" / "Don't know", please provide your comment(s) in the comment text box****]***

1. Do you think a particular feature should be developed, only after the selection and approval of the same by the core community?

- Yes
- No
- Maybe
- Don't know

Comments: _____

2. Do you think it is a good practice to demonstrate the necessity of a feature before selecting the same?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

3. Whom do you think is responsible for selecting and finalising the feature for development?

- Community
- Core team
- Both
- Don’t know

Comments: _____

4. Proposed OSSD process requires a brief description of each candidate feature to be published for better selection. Do you think it is a good practice to be carried out?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

5. Do you think it is necessary to publish the contact details of a person/team for each of the candidate feature (which might helps in further discussions) before the selection process?

- Yes
- No
- Maybe

- Don’t know

Comments: _____

6. Do you think the proposed OSSD process needs any change with respect to feature selection? If so, please provide details in the comment text box below. (Max. 500words)

Comment: _____

7. Please provide your detailed critique on OSSD process’s feature selection in the comment textbox below. (Max. 500words)

Comment: _____

SECTION 3 – Software Practice – Requirement Management

[*Please answer all the questions. If you have answered “No” / “Don’t know”, please provide your comment(s) in the comment text box****]***

1. Do you think that the OS community will appreciate the idea of developing the requirement specification for all selected feature before development?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

2. Do you think it is feasible in an OS environment to develop a requirement specification document for all selected feature before development?

- Yes
- No

- Maybe
- Don’t know

Comments: _____

3. Do you think the OS community will use the requirement specification document as a basis for developing/adjusting development plan/development activities that are to be carried out for implementing a feature?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

4. Do you think it is important to review the requirement specification before proceeding further?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

5. Do you think the proposed OSSD process needs any change with respect to the requirement management? If so, please provide details in the comment text box below. (Max. 500words)

Comment: _____

6. Please provide your detailed critique on OSSD process’s requirement management in the comment textbox below. (Max. 500words)

Comment: _____

SECTION 4 – Software Practice – Design Process

[*Please answer all the questions. If you have answered "No" / "Don't know", please provide your comment(s) in the comment text box****]***

1. Proposed OSSD process makes it necessary to develop a brief and clear design document. Do you think it is practicable in an OS environment?

- Yes
- No
- Maybe
- Don't know

Comments: _____

2. Do you think it is sufficient, if the core team alone validates the design documents?

- Yes
- No
- Maybe
- Don't know

Comments: _____

3. Do you think it is important to communicate and share these design documents with the entire community before proceeding towards implementation?

- Yes
- No
- Maybe
- Don't know

Comments: _____

4. Do you think it is practical to allow the entire community to validate the design document before implementation?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

5. Do you think the proposed OSSD process needs any change with respect to design process? If so, please provide details in the comment text box below. (Max. 500words)

Comment: _____

6. Please provide your detailed critique on OSSD process’s design process in the comment textbox below. (Max. 500words)

Comment: _____

SECTION 5 – Software Practice – Software Implementation

[*Please answer all the questions. If you have answered “No” / “Don’t know”, please provide your comment(s) in the comment text box****]***

1. Proposed OSSD process makes it necessary to develop and share a very brief implementation document before proceeding further. Do you think it is a good practice?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

2. Do you think that sharing implementation document will enable the community to easily and actively participate in the feature development?

- Yes

- No
- Maybe
- Don’t know

Comments: _____

3. Do you think developing a unit testing strategy and validation of the same by the core team is practical in an OS environment?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

4. Do you think the unit testing strategy should be shared with the entire community for better verification and validation of the feature?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

5. Do you think the proposed OSSD process needs any change with respect to software implementation? If so, please provide details in the comment text box below. (Max. 500words)

Comment: _____

6. Please provide your detailed critique on OSSD process’s software implementation process in the comment textbox below. (Max. 500words)

Comment: _____

SECTION 6 – Software Practice – Quality Assurance

[*Please answer all the questions. If you have answered "No" / "Don't know", please provide your comment(s) in the comment text box****]***

1. Do you think the proposed OSSD process provides sufficient room for quality assurance activities?

- Yes
- No
- Maybe
- Don't know

Comments: _____

2. Do you think the proposed OSSD process allows sufficient space for providing reviews/feedbacks by its community towards feature development?

- Yes
- No
- Maybe
- Don't know

Comments: _____

3. Do you think there is enough space for addressing the feedbacks/reviews before the feature is released?

- Yes
- No
- Maybe
- Don't know

Comments: _____

4. Should the community be provided with a defined set of criteria for testing?

- Yes

- No
- Maybe
- Don't know

Comments: _____

5. If 'YES', should the criteria be created/validated by the core team before giving it to the community members for testing?

- Yes
- No
- Maybe
- Don't know

Comments: _____

6. Do you think the proposed OSSD process needs any change with respect to software quality assurance? If so, please provide details in the comment text box below. (Max. 500words)

Comment: _____

7. Please provide your detailed critique on OSSD process's quality assurance activities in the comment textbox below. (Max. 500words)

Comment: _____

SECTION 7 – Software Practice – Software Integration and Release

[*Please answer all the questions. If you have answered "No" / "Don't know", please provide your comment(s) in the comment text box****]***

1. Do you think identifying the release-item(s) well before the release will help the community to plan/develop efficiently?

- Yes
- No
- Maybe
- Don't know

Comments: _____

2. Do you think developing and sharing an integration strategy before release will help the OS community?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

3. Proposed OSSD process requires verification of the software unit produced with the requirement & design before release. Do you think it is a necessary step to be carried out before release?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

4. Do you think it is a good practice to explicitly communicate about the release and the release items to all the affected parties?

- Yes
- No
- Maybe
- Don’t know

Comments: _____

5. Do you think the proposed OSSD process needs any change with respect to software integration and release? If so, please provide details in the comment text box below. (Max. 500words)

Comment: _____

6. Please provide your detailed critique on OSSD process's software implementation & release process in the comment textbox below. (Max. 500words)

Comment: _____

SECTION 8 – General Feedback on the Proposed OSSD Process

1. Do you think the proposed OSSD process have all the necessary stages of development? (Max. 500words)

Feedback:

_____

2. Do you think the development stages are ordered correctly and are iterated sufficiently? (Max. 500words)

Feedback:

_____

3. Please highlight both the negative and positive aspects of the proposed OSSD process. (Max. 500words)

Feedback:

_____

APPENDIX C

C.1 OS Expert-Validation Results

C.1.1 Expert – 1

Question	Answer
1. Full Name	Expert 1
2. E-mail	-
3. How many years of experience do you have in open source software (OSS) development?	4
4. How many different OSS projects have you worked on?	2
5. For how long are you working on your latest OSS project ? (in years)	4
6. What is your current project role? (select as many as applies and others please specify)	Others
Others:	E-Learning Consultant in university, using a open-source Learning Management system
7. What are the different software development activities do you currently work on? (select as many as applies and others please specify)	Software requirement Testing/Integration
8. Do you follow any documented software development process?	Yes
If 'YES" can you very briefly describe it?	At we do see the dev guideline of ILIAS. There we can see who the maintainers are, and who is testing the different components. Every summer, we do test some components, and for this we do get

	a large xls-document with all test-cases. At , we write all our feature-requests. Everyone of the community can then add comments. Then we can add the feature-request to a jour-fixe. Everyone can take part in this jour-fixe, but in most cases, only the core-team does join this jour-fixe. If a feature-request is accepted, then we must find funding, and a programming-"company". We do also have a feature-freeze. At , we do see the roadmap. At , we do report bugs. There are also accepted and implemented guidelines: . If a guideline is not used, we can add a bug-report.
1. Do you think a particular feature should be developed only after its selection and approval by the core community?	No
Comments / feedback:	No, not every feature must have an

	approval by the core community. But every big feature (like e-Portfolio in ILIAS) must be accepted by the core community. But the community should have the possibility to add comments to a particular feature.
2. Do you think it is a good practice to demonstrate the importance and need of a particular feature before selecting the same?	Don't know
Comments / feedback:	It is a good procedure that every feature for the core must be open to the community, so every feature must be described in a wiki or another tool. It is also important, that feature-descriptions do explain why this feature is important.
3. Whom do you think is responsible for selecting and finalizing the feature for development?	Both
Comments / feedback:	Both. The community does have the needs, the ideas, the end-users. But only the core-team does know exactly the whole concept of a system.
4. The proposed OSSD process requires a brief description of each candidate feature to be published for selecting the right feature. Do you think that performing this task is a good practice?	No

Comments / feedback:	This is overkill. In (our) open source community, most features will only be developed if there is some funding. So selecting the right feature is in most cases not an option. OS-Communities can only say: "Good idea - we like and accept it / and give comments", or "Bad idea - develop it in a branch, or develop a plug-in".
5. Do you think it is necessary to publish the contact details of a person/team for each of the candidate feature (which might help in further discussions) before the selection process?	Both
Comments / feedback:	
6. Do you think the proposed OSSD process needs any change with respect to feature selection? If so, please provide your details in the text box below.	The first stage is not realistic. There is really (in our LMS) no selection possible. Important features will be financed with funding of the institutions-members of the community. All other features will only be developed if someone has the funding. And all feature-requests can be open to the community from the beginning.
7. Please provide your detailed critique on the proposed OSSD process's feature selection in	No more comments

the text box below.	
1. Do you think that the OSS community will agree with the idea of developing requirement specification for all selected feature before development?	Maybe
Comments / feedback:	
2. Do you think it is feasible in an OSS environment to develop a requirements specification document for all selected features before the development starts?	Maybe
Comments / feedback:	Not for all developments, but for the big ones, a requirement specification document is useful.
3. Do you think the OSS community will use the requirement specification document as a basis for developing/adjusting development plan/development activities that are to be carried out for implementing a feature?	Yes
Comments / feedback:	
4. Do you think it is important to review the requirements specification by the OSS community even before the design process?	Maybe
Comments / feedback:	
5. Do you think the proposed OSSD process needs any change with respect to requirements management? If so, please give us your feedback / comments in the text box below.	No
6. Please provide your detailed critique on the proposed OSSD process's requirements management in the text box below.	The question is: Who does finance the design-experts in the design specification stage. This is an important thing, but i do not see a solution yet. Programming

	companies do have knowhow in writing specifications and programming, the community knows their needs, but we do need a usability and design-team, who can give good feedback.
1. The proposed OSSD process makes it necessary to develop a brief and clear design document. Do you think it is practicable in an OSS environment?	Yes
Comments / feedback:	
2. Do you think it is sufficient for the core team alone to validate the design document?	Maybe
Comments / feedback:	
3. Do you think it is important to communicate and share these design documents with the entire community before proceeding towards the implementation phase?	Yes
Comments / feedback:	
4. Do you think it is practical to allow the entire community to validate the design documents?	Yes
Comments / feedback:	
5. Should the entire community be involved in the validation of design documents?	Yes
Comments / feedback:	
6. Do you think the proposed OSSD process needs any change with respect to the design process? If so, please give us your feedback / comments in the text box below.	Already written.
7. Please provide your detailed critique on the proposed OSSD process's design process in the text box below.	-
1. The proposed OSSD process makes it	Yes

necessary to develop and share a very brief implementation document before actually implementing (coding). Do you think it is a good practice?	
Comments / feedback:	
2. Do you think, that sharing the implementation document will enable the community to easily and actively participate in the feature development?	Yes
Comments / feedback:	
3. Do you think developing a unit testing strategy and validation of the same by the core team is practical in an OSS environment?	Yes
Comments / feedback:	
4. Do you think the unit testing strategy should be shared with the entire community for better verification and validation of the feature?	Yes
Comments / feedback:	
5. Do you think the proposed OSSD process needs any change with respect to software implementation? If so, please give us your feedback / comments in the text box below.	-
6. Please provide your detailed critique on the proposed OSSD process's software implementation in the text box below.	-
1. Do you think the proposed OSSD process provides sufficient room for quality assurance activities?	Yes
Comments / feedback:	
2. Do you think the proposed OSSD process provides sufficient space for providing reviews/feedback by its community towards feature development?	Yes
Comments / feedback:	
3. In the proposed OSSD process, do you think there is enough space for addressing the	Yes

feedback/reviews before the feature is released?	
Comments / feedback:	
4. Should the community be provided with explicit criteria (on - what they should look for) before they start the testing?	Yes
Comments / feedback:	
5. If 'YES', should the criteria be created /validated by the core team before giving it to the community members for testing?	Yes
Comments / feedback:	
6. Do you think the proposed OSSD process needs any change with respect to software quality assurance? If so, please give us your feedback / comments in the text box below.	-
7. Please provide your detailed critique on the proposed OSSD process's quality assurance activities in the text box below.	-
1. Do you think identifying the release item(s) well before the release will help the community to plan/develop the feature efficiently?	Yes
Comments / feedback:	
2. Do you think developing and sharing an integration strategy before release will help the OSS community?	Yes
Comments / feedback:	
3. The proposed OSSD process requires verification of the software unit produced with the requirement and design before release. Do you think it is a necessary step to be carried out before release?	Yes
Comments / feedback:	
4. Do you think it is a good practice to explicitly communicate about the release and the release items to all the affected parties (entire community, commercial users, etc.)?	Yes

Comments / feedback:	
5. Do you think the proposed OSSD process needs any change with respect to the software integration and release? If so, please give us your feedback / comments in the text box below.	-
6. Please provide your detailed critique on the proposed OSSD process's software implementation & release process in the text box below.	-
1. Do you think the proposed OSSD process has all the necessary stages of development?	Yes
2. Do you think the development stages are ordered correctly and are iterated sufficiently?	Stage1 is in some OS-Software-developments not possible.
3. Please highlight both the negative and positive aspects of the proposed OSSD process.	Negative: Testing is not fun, but necessary. Positive: Design process! - this is really a necessary stage.

C.1.2 Expert – 2

Question	**Answer**
1. Full Name	Expert 2
2. E-mail	-
3. How many years of experience do you have in open source software (OSS) development?	3
4. How many different OSS projects have you worked on?	2
5. For how long are you working on your latest OSS project ? (in years)	1
6. What is your current project role? (select	Project leader

as many as applies and others please specify)	Developer
Others:	
7. What are the different software development activities do you currently work on? (select as many as applies and others please specify)	Design Coding & unit testing
8. Do you follow any documented software development process?	No
If 'YES" can you very briefly describe it?	
1. Do you think a particular feature should be developed only after its selection and approval by the core community?	May be
Comments / feedback:	if the developer want to do that in his spare time, why not; if the result is ok it can be proposed to the community and will be included in the process with some improvements if necessary
2. Do you think it is a good practice to demonstrate the importance and need of a particular feature before selecting the same?	May be
Comments / feedback:	as i said above, if someone want to work on that in his spare time why demonstrating ? if the feature is already requested by the community than it's value is already validated; if the implementation if difficult than it might be useful to make a demo as a proof of concept and recruit more developers;
3. Whom do you think is responsible for	Core team

selecting and finalizing the feature for development?	
Comments / feedback:	they already know the inner workings of the existing code and could see where the tweaking has to be done
4. The proposed OSSD process requires a brief description of each candidate feature to be published for selecting the right feature. Do you think that performing this task is a good practice?	Yes
Comments / feedback:	
5. Do you think it is necessary to publish the contact details of a person/team for each of the candidate feature (which might help in further discussions) before the selection process?	Both
Comments / feedback:	I'm not sure if the questionnaire options are correlated with the question; anyway the contact coordinates have to be published so that a prospective person interested in finding more info regarding the feature know whom to contact;
6. Do you think the proposed OSSD process needs any change with respect to feature selection? If so, please provide your details in the text box below.	depending on the project i would propose switching between core team and community as that in the first phase community should propose and vote a specific set of features to be submitted for analysis to the community and in the second phase the core team will give their feedback on what can/can't be done and

	why; in a large project you solution can waste the time of the core team and anyway if the project has a big community (eg. Drupal) there are a lot of skilled developers who can spot interesting features; you solution gives to much power to the core team and they can misguide the project by not implementing some of the features at their free will;
7. Please provide your detailed critique on the proposed OSSD process's feature selection in the text box below.	see above
1. Do you think that the OSS community will agree with the idea of developing requirement specification for all selected feature before development?	Maybe
Comments / feedback:	it depends of the project and the adopted development methodology; most of the small projects won't use that and some of the large projects won't use either because not having requirements gives them more freedom to implement the feature as they wish and make some shortcuts in the development
2. Do you think it is feasible in an OSS environment to develop a requirements specification document for all selected features before the development starts?	Maybe
Comments / feedback:	see above

3. Do you think the OSS community will use the requirement specification document as a basis for devoloping/adjusting development plan/development activities that are to be carried out for implementing a feature?	Maybe
Comments / feedback:	if they agree (see question 3.1) they should use it because they have it; if someone from the core team will advocate 'pro' this methodology it will be used for one feature and depending on how it's working (if it is good for the final outcome) they will decide to use it or not for the following features
4. Do you think it is important to review the requirements specification by the OSS community even before the design process?	No
Comments / feedback:	
5. Do you think the proposed OSSD process needs any change with respect to requirements management? If so, please give us your feedback / comments in the text box below.	the requirements should be public entirely; it's and OS process and the development documents are part of the project so why not showing to the community that you are working on something...; as i said in section 2 and 3 my opinion is to involve the community before the core team; it does not worth investing time and resources for developing all this documentation if the community does not want this particular feature; regarding the final

	statement of this section: it is too early to freeze anything; we are not a corporation and we are working for our own pleasure; if you will impose too strict guideline you risk loosing people endangering the project
6. Please provide your detailed critique on the proposed OSSD process's requirements management in the text box below.	see above
1. The proposed OSSD process makes it necessary to develop a brief and clear design document. Do you think it is practicable in an OSS environment?	Maybe
Comments / feedback:	
2. Do you think it is sufficient for the core team alone to validate the design document?	Yes
Comments / feedback:	they know the inner workings of the project; they are the only responsible for the well being of the project and can understand possible problems that will arise from a particular design; the community is not interested on how it will implemented but only that is will be there, when and how will they be able to use it;
3. Do you think it is important to communicate and share these design documents with the entire community before proceeding towards the implementation phase?	Maybe

Comments / feedback:	to share but not to discuss; in a large project there will be always 'wise' people with ideas but if they don't want to work on the particular feature let them keep their ideas; if they want to work then they will be on the dev team and obviously they will discuss the design decisions; the documents should be published because they prove that something is happening on the feature;
4. Do you think it is practical to allow the entire community to validate the design documents?	No
Comments / feedback:	see above
5. Should the entire community be involved in the validation of design documents?	No
Comments / feedback:	see 3
6. Do you think the proposed OSSD process needs any change with respect to the design process? If so, please give us your feedback / comments in the text box below.	...
7. Please provide your detailed critique on the proposed OSSD process's design process in the text box below.	...
1. The proposed OSSD process makes it necessary to develop and share a very brief implementation document before actually implementing (coding). Do you think it is a good practice?	Maybe
Comments / feedback:	depending on the type of project and structure of the dev team;

	on a small project it is useless and also on a geographically localized one or when the coders have a good communication between them; there might be 5 devs which already know each other and work symbiotically; in large projects or when the coders don't work together it is a good document;
2. Do you think, that sharing the implementation document will enable the community to easily and actively participate in the feature development?	Maybe
Comments / feedback:	the selected coders will work most actively; if in the middle of the project someone wants to join them it may benefit from such a document but is quite unlikely;
3. Do you think developing a unit testing strategy and validation of the same by the core team is practical in an OSS environment?	Maybe
Comments / feedback:	the strategy as a political guideline but nothing specific;
4. Do you think the unit testing strategy should be shared with the entire community for better verification and validation of the feature?	Yes
Comments / feedback:	
5. Do you think the proposed OSSD process needs any change with respect to software implementation? If so, please	...

give us your feedback / comments in the text box below.	
6. Please provide your detailed critique on the proposed OSSD process's software implementation in the text box below.	in this phase it might be useful to implement some clear and publicly available milestone system because usually after the coding starts it never ends :) community should be able to clearly see that something is happening and exactly what is happening;
1. Do you think the proposed OSSD process provides sufficient room for quality assurance activities?	Yes
Comments / feedback:	
2. Do you think the proposed OSSD process provides sufficient space for providing reviews/feedback by its community towards feature development?	Yes
Comments / feedback:	
3. In the proposed OSSD process, do you think there is enough space for addressing the feedback/reviews before the feature is released?	Yes
Comments / feedback:	
4. Should the community be provided with explicit criteria (on - what they should look for) before they start the testing?	Maybe
Comments / feedback:	depending on the specific feature and phase of the dev; obviously in the first cycles there will be interest in validating the functional requirements of the feature and there should be more effort on testing that as it is

	obvious that the non critical parts will be buggy
5. If 'YES', should the criteria be created/validated by the core team before giving it to the community members for testing?	Don't know
Comments / feedback:	i selected Maybe on the previous question
6. Do you think the proposed OSSD process needs any change with respect to software quality assurance? If so, please give us your feedback / comments in the text box below.	you should emphasize on bug tracking; bugs are the physical manifestation of any problem that should be covers by unit testing; but you may have the best unit testing and don't catch some bug because it's caused by the interaction of the components; if you have a bug you should solve it regardless of unit testing, and this is where the community of good; they will report bugs because they can see and understand them; unit testing is for the dev team, bugs are for mere mortals :))
7. Please provide your detailed critique on the proposed OSSD process's quality assurance activities in the text box below.	all ready stated above
1. Do you think identifying the release item(s) well before the release will help the community to plan/develop the feature efficiently?	Yes
Comments / feedback:	
2. Do you think developing and sharing an integration strategy before release will help the OSS community?	Maybe

Comments / feedback:	depends of the complexity of the project and the size of the development team
3. The proposed OSSD process requires verification of the software unit produced with the requirement and design before release. Do you think it is a necessary step to be carried out before release?	Maybe
Comments / feedback:	only for internal usage and/or personal satisfaction;
4. Do you think it is a good practice to explicitly communicate about the release and the release items to all the affected parties (entire community, commercial users, etc.)?	Yes
Comments / feedback:	
5. Do you think the proposed OSSD process needs any change with respect to the software integration and release? If so, please give us your feedback / comments in the text box below.	...
6. Please provide your detailed critique on the proposed OSSD process's software implementation & release process in the text box below.	...
1. Do you think the proposed OSSD process has all the necessary stages of development?	it has
2. Do you think the development stages are ordered correctly and are iterated sufficiently?	i would iterate more on the design and implementation, implementation being interleaved with testing; it there are really good architects the design can be good from the start but on a lot of smaller projects there is also an adventurous/exploratory

	part when the team is not sure of what and how they want to achieve;
3. Please highlight both the negative and positive aspects of the proposed OSSD process.	good: introducing order, structure and discipline in OS dev; usually there are a bunch of people writing code and this should give them some guidelines not so good: to much order can deter them form the project; if the core team or the leader if a methodology-nazi some of the prospective contributors can decide not to join the project only because of the rules especially if they are not having training in software development;

C.1.3 Expert – 3

Question	**Answer**
1. Full Name	Expert 3
2. E-mail	-
3. How many years of experience do you have in open source software (OSS) development?	.5
4. How many different OSS projects have you worked on?	1
5. For how long are you working on your latest OSS project? (in years)	.5
6. What is your current project role? (select as many as applies and others please specify)	Developer
Others:	
7. What are the different software	Design

development activities do you currently work on? (select as many as applies and others please specify)	Coding & unit testing
8. Do you follow any documented software development process?	No
If 'YES" can you very briefly describe it?	
1. Do you think a particular feature should be developed only after its selection and approval by the core community?	Yes
Comments / feedback:	
2. Do you think it is a good practice to demonstrate the importance and need of a particular feature before selecting the same?	Yes
Comments / feedback:	
3. Whom do you think is responsible for selecting and finalizing the feature for development?	Both
Comments / feedback:	
4. The proposed OSSD process requires a brief description of each candidate feature to be published for selecting the right feature. Do you think that performing this task is a good practice?	Yes
Comments / feedback:	
5. Do you think it is necessary to publish the contact details of a person/team for each of the candidate feature (which might help in further discussions) before the selection process?	Both
Comments / feedback:	
6. Do you think the proposed OSSD process needs any change with respect to feature selection? If so, please provide your details in the text box below.	No change needed in my opinion.
7. Please provide your detailed critique on	Due the amount of work

the proposed OSSD process's feature selection in the text box below.	for a new feature for participating institutions it leads to custom implementations that never find their way into the OO-project.
1. Do you think that the OSS community will agree with the idea of developing requirement specification for all selected feature before development?	Yes
Comments / feedback:	
2. Do you think it is feasible in an OSS environment to develop a requirements specification document for all selected features before the development starts?	Yes
Comments / feedback:	
3. Do you think the OSS community will use the requirement specification document as a basis for developing/adjusting development plan/development activities that are to be carried out for implementing a feature?	Yes
Comments / feedback:	
4. Do you think it is important to review the requirements specification by the OSS community even before the design process?	Yes
Comments / feedback:	
5. Do you think the proposed OSSD process needs any change with respect to requirements management? If so, please give us your feedback / comments in the text box below.	No
6. Please provide your detailed critique on the proposed OSSD process's requirements management in the text box below.	Nothing
1. The proposed OSSD process makes it necessary to develop a brief and clear	Yes

design document. Do you think it is practicable in an OSS environment?	
Comments / feedback:	
2. Do you think it is sufficient for the core team alone to validate the design document?	Maybe
Comments / feedback:	Not sure whether the community should be part of this as well.
3. Do you think it is important to communicate and share these design documents with the entire community before proceeding towards the implementation phase?	Yes
Comments / feedback:	
4. Do you think it is practical to allow the entire community to validate the design documents?	Yes
Comments / feedback:	
5. Should the entire community be involved in the validation of design documents?	No
Comments / feedback:	Just some core community members
6. Do you think the proposed OSSD process needs any change with respect to the design process? If so, please give us your feedback / comments in the text box below.	-
7. Please provide your detailed critique on the proposed OSSD process's design process in the text box below.	-
1. The proposed OSSD process makes it necessary to develop and share a very brief implementation document before actually implementing (coding). Do you think it is a good practice?	No

Comments / feedback:	I think the requirements and design specs are enough
2. Do you think, that sharing the implementation document will enable the community to easily and actively participate in the feature development?	Don't know
Comments / feedback:	
3. Do you think developing a unit testing strategy and validation of the same by the core team is practical in an OSS environment?	Yes
Comments / feedback:	
4. Do you think the unit testing strategy should be shared with the entire community for better verification and validation of the feature?	Yes
Comments / feedback:	
5. Do you think the proposed OSSD process needs any change with respect to software implementation? If so, please give us your feedback / comments in the text box below.	-
6. Please provide your detailed critique on the proposed OSSD process's software implementation in the text box below.	-
1. Do you think the proposed OSSD process provides sufficient room for quality assurance activities?	Yes
Comments / feedback:	
2. Do you think the proposed OSSD process provides sufficient space for providing reviews/feedback by its community towards feature development?	Yes
Comments / feedback:	
3. In the proposed OSSD process, do you think there is enough space for addressing	Yes

the feedback/reviews before the feature is released?	
Comments / feedback:	
4. Should the community be provided with explicit criteria (on - what they should look for) before they start the testing?	No
Comments / feedback:	Explicit criteria makes that the community members only test those things.
5. If 'YES', should the criteria be created /validated by the core team before giving it to the community members for testing?	Yes
Comments / feedback:	
6. Do you think the proposed OSSD process needs any change with respect to software quality assurance? If so, please give us your feedback / comments in the text box below.	-
7. Please provide your detailed critique on the proposed OSSD process's quality assurance activities in the text box below.	-
1. Do you think identifying the release item(s) well before the release will help the community to plan/develop the feature efficiently?	Yes
Comments / feedback:	
2. Do you think developing and sharing an integration strategy before release will help the OSS community?	Don't know
Comments / feedback:	
3. The proposed OSSD process requires verification of the software unit produced with the requirement and design before release. Do you think it is a necessary step to be carried out before release?	Yes
Comments / feedback:	

4. Do you think it is a good practice to explicitly communicate about the release and the release items to all the affected parties (entire community, commercial users, etc.)?	Yes
Comments / feedback:	
5. Do you think the proposed OSSD process needs any change with respect to the software integration and release? If so, please give us your feedback / comments in the text box below.	-
6. Please provide your detailed critique on the proposed OSSD process's software implementation & release process in the text box below.	-
1. Do you think the proposed OSSD process has all the necessary stages of development?	Yes
2. Do you think the development stages are ordered correctly and are iterated sufficiently?	Yes. Iterations are fine as long the features aren't too big. Else there are bigger iterations needed (over multiple stages, not only within a single stage).
3. Please highlight both the negative and positive aspects of the proposed OSSD process.	-