Digital Innovations & Contemporary Research in SCIENCE & ENGINEERING
Vol. 5 No. 2, June. 2017

Article Citation Format
A.O, Balogun, M.A. Mabayoje, E.O. Adeniyi & S.A. Salihu (2017). A Framework for Coordinating Usability Engineering and Software Engineering Activities in the Development of Content Management Systems. Journal of Digital Innovations & Contemp Res. In Sc., & Eng Vol. 5, No. 2.

Article Progress Time Stamps

Article Type: Research Article
Manuscript Received: 13rd September, 2016
Review Type: Blind
Review/Acceptance Information Sent : 16th Dec, 2017
Final Acceptance:: 1st May, 2017
DOI Prefix: 10.22624

A Framework for Coordinating Usability Engineering and Software Engineering Activities in the Development of Content Management Systems

A.O, Balogun, M.A. Mabayoje, E.O. Adeniyi & S.A. Salihu
Department of Computer Science
University of Ilorin, Ilorin, Nigeria.
bharlow058@gmail.com, mmabayoje@gmail.com, iadeniyiolufemi@yahoo.com, shaksoft@yahoo.com

ABSTRACT
Due to the expansion of the internet in recent years, we have witnessed an increase in popularity of web applications and its technologies. A particular technology of web application, Content Management system, has also gained relevance as they facilitate the distribution of wide varieties of content. The process involved in designing and developing content management system is a complex procedure due to the variability of its requirement over time which has effects on its architecture and design. Currently, the Usability Engineering (UE) and Software Engineering (SE) processes are practiced as being independent of each other. However, several dependencies and constraints exist between these two frameworks, which make coordination between the UE and the SE teams crucial. Failure of coordination between the UE and SE teams leads to CMS that often lacks necessary functionality and impedes user performance. At the same time, the UE and SE processes cannot be integrated because of the differences in focus, techniques, and terminology. We therefore propose a development framework that incorporates SE and UE efforts to guide current CMS development. The framework characterizes the information exchange that must exist between the UE and SE teams during CMS development to form the basis of the coordinated development framework. The UE Scenario-Based Design (SBD) process provides the basis for identifying UE activities. Similarly, the Requirements Generation Model (RGM), and Structured Analysis and Design are used to identify SE activities. We identify UE and SE activities that can influence each other, and identify the high-level exchange of information that must exist among these activities. We further examine these interactions to gain a more in-depth understanding as to the precise exchange of information that must exist among them. The identification of interacting activities forms the basis of a coordinated development framework that incorporates and synchronizes the UE and SE processes.

Keywords: CMS, Coordination, Integration, Software Engineering, Usability Engineering, Requirements Generation, Framework, Development Process computation

1. BACKGROUND TO THE STUDY

Since the beginning of web engineering, many web modeling methods have been proposed in order to provide efficient and structured ways of developing web applications. However, these web modeling methods only provided design primitives for building web applications from scratch, and not including the possibility to design pre-existing components related to a particular web domain. Content Management Systems (CMS) are software that enable one to add and/or manipulate content on a Web site. Web Content Management (WCM) systems are software products used for the management and control of web content. WCM systems have gradually merged with web applications resulting in CMS-based web applications [27]. The implementation and continuous usage of WCMSs is a complex task, since one has to cope with fast-changing requirements which have a high impact on architecture and design. Key in this process is the translation of business requirements into pre-existing components. In order to make this process more transparent and effective and to improve the usability of CMS, the usability engineering and the software engineering processes must be well defined and incorporated.

The Usability Engineering and Software Engineering life cycle activities help develop "usable" and "functionally satisfactory" software systems. The factors that make a system more or less usable and functionally satisfactory are complex. A usable system should enhance human performance and be easy to learn. At the same time, a usable system should be easily adaptable and should provide a satisfying user experience. A functionally satisfactory system should possess the necessary functionality to satisfy the requirements of all users when deployed in the users' environments. Additionally, the system should interface well with the other systems already in use. The ultimate goal of UE is to create systems with a measurably high usability; the practical objective is to provide interaction design specifications to software engineers [20]. Usability Engineering is the process that incorporates a set of methods and techniques used by the usability engineers to design a "usable" system. The process helps the usability engineers in iteratively evaluating how usable the software is, and improving the design to make software more usable.

Software Engineering is the application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. Software Engineering addresses the application of engineering methods to software development. With the application of sound software engineering principles, the software developed is reliable, maintainable, and efficient. The software engineering process identifies the problems for which a software solution is to be generated. It also identifies the characteristics of the software solution that can solve these problems. Software Engineering further identifies the paths to construct a software solution, as well as defines strategies for error detection and downstream maintenance. Software engineers have introduced a structure to the overall software development process through a better understanding of the various processes involved in software development. These processes have been expressed as software development models. The Waterfall Model [24], the Incremental Model [16], and the Spiral Model [2] are examples of software development models. Software development models decompose the software development process into five major phases: the requirements engineering phase, the software design phase, the implementation phase, the integration phase, and the maintenance phase. The ultimate goal of software engineering is to engineer software systems that possess the necessary functionality to support user requirements.

As with any web based application, basic usability principles should be followed. But web interfaces are different from normal software Graphical User Interfaces (GUIs), as there are a variety of different technologies that can be used. The technical abilities of someone creating a web interface can be much lower than that of a normal software engineer as the technologies and means to create a website are easier to get hold of. A huge problem with this is that, if the user does not like a site because it is unusable, difficult to read, or poorly designed, he tends to look for alternatives. This is a usability problem. A usable WCMS back-end saves money and time on training users, makes implementation and upkeep of a site easier. It enhances integration and automation of processes that contribute to efficient dissemination of information on the Internet.

1.1 Statement of Problem

Traditional software engineering lifecycles do not explicitly address usability [15]. Usability Engineering (UE) and Software Engineering (SE) are still typically separate and are applied independently in CMS development. For example, it is not uncommon to find usability engineers being brought into the development process after the implementation stage. They are asked to 'fix' the usability of an already implemented system, and even then most changes proposed by the usability engineers that require architectural modifications are often ignored due to budget and time constraints. Those few changes that actually get retrofitted incur huge costs. The inadequate coordination between the usability and software engineers often leads to conflicts, gaps, and miscommunication. This research work therefore focuses on defining a development framework that enhances effective/efficient coordination and synchronization of UE and SE processes during CMS development [11][13]

1.2 Objective

The objective of this research work is to define a framework based on the definition of information exchange among influencing Usability Engineering and Software Engineering activities in Content Management System development.

2. RELATED WORKS

In this section of the chapter, we present an overview of the other researchers' efforts directed towards the integration of Usability Engineering into the Software Engineering process. We also highlight the important observations that have emerged through these research efforts.

2.1 Milewski

Milewski [18] focuses on usability and software engineering education and addresses some issues related to integration of the UE and SE processes. Milewski claims that integrating the UE and SE teams (to become a single entity) is not possible for some reasons. However, he was not able to establish the fact that optimum coordination can be achieved among these two teams. This coordination allows the teams to work concurrently with regards to each other in software development processes[14][15][17].

Some of the reasons he highlighted for the impossibility in integrating the teams are:

- Several application functions are neither directly related to the user, nor would benefit from user involvement. Such functions may exist in the software functionality.
- Having the user advocate semi-separated from the schedule and budget demands of the rest of the project is best.
- Too much work may be required to combine positions and to interface others' roles in the project (systems engineers, developers, designers, marketers, etc.), and this work is typically too much for a single role (combined usability and software engineer) to handle.

Milewski further claims that creating a common integrated process model will not make the situation more manageable because:

- Process models in practice are adapted to fit the specifics of the environment and the needs of the specific project.
- Process models are more descriptive of what actually happens rather than what must happen.

2.2 Natalia Juristo

Juristo et al [11] explore the use of architectural patterns designed to enhance the usability of software products. Their research promotes the use and application of architectural patterns to enhance software usability. Juristo and Lopez advocate a forward engineering approach to integrate UE into the SE process by suggesting the use of architectural patterns for software to enhance usability of the software product. The approach taken by these researchers differs from the traditional approach of measuring usability after the development of the software product. They consider usability as a quality attribute of a software system and address the problem of integrating usability characteristics into the general software architecture. The general UE quality attributes are too high to integrate into the software architecture. To solve this problem, Juristo and Lopez have decomposed the UE process into levels of abstraction progressively closer to SE. The two intermediate levels are usability properties and usability patterns[5][6][10]The usability properties make software usable and usability patterns act as mechanisms to incorporate usability properties into software architecture. The usability patterns do not offer a software solution, but suggest an abstract mechanism to incorporate usability patterns into software architecture. The implementation of usability patterns into software architecture is therefore a problem. Juristo and Lopez claim that architectural patterns will reflect a possible solution to the problem of implementation of usability patterns. The usability patterns are unique for every usability property. The architectural patterns are therefore the last link of the UE attribute-property-pattern chain connecting software usability with the software architecture[19][21][22]

2.3 Xavier Ferré

Xavier [8] states observations made by his research team working on the "STATUS" project. The aim of the STATUS project is to study and determine the connections between software architecture and the usability of the resultant software system. Ferré claims that the use of UE techniques is not straightforward, because they are not integrated into the SE process. Therefore, his research tries to introduce a handy group of increments, which when included in the software engineering process, would ensure higher usability of the resultant software. Ferré presents a survey of the UE literature to identify activities that were best suited, and agreed upon by researchers, for inclusion in the software engineering process [23][24][25]

To identify the best-suited activities, the researchers listed UE activities that enjoyed general acceptance in the Human-Computer-Interaction field. They also checked for activities that were less alien to SE, had low integration costs, and had a higher general applicability. The "findings" from the UE field were then mapped to the activities in the SE field, adapting their results to the SE concepts and terminology. Grouping of similar activities formed the increments or "deltas." The paper discusses in detail the phases of the integrated process and the usability increments applicable to the SE process. Ferré however, did not highlight the necessary interactions that need to take place among the individual activities of both UE and SE. This is necessary for the definition of a development framework for the coordination of UE and SE processes [28].

3. CURRENT PRACTICES

In spite of the extensive research in Software and Usability Engineering, there has been a marked deficiency of understanding between the two generally in software products development and particularly in CMS development. In general, the two teams do not usually seek to understand the other's goals and needs and have inadequate appreciation for the other's area of expertise. Douglas [7] opined that the apparent reason for this situation is the way computer science courses are typically offered in colleges, he stated that SE courses often omit any references to user interface development techniques and UE courses hardly discuss the SE implications of usability patterns. Douglas' claims remain to be validated. The simple fact remains that these two teams do not work hand-in-hand with each other in the development of CMS; their activities are carried out independently and in parallel manner. In most cases, usability engineers come into action after the CMS has been developed and, in some cases contents getting published, their role is now reduced to fixing already made errors that greatly affect the usability and also functionality of CMS. In some cases, this absence of coordination results in failed software projects.

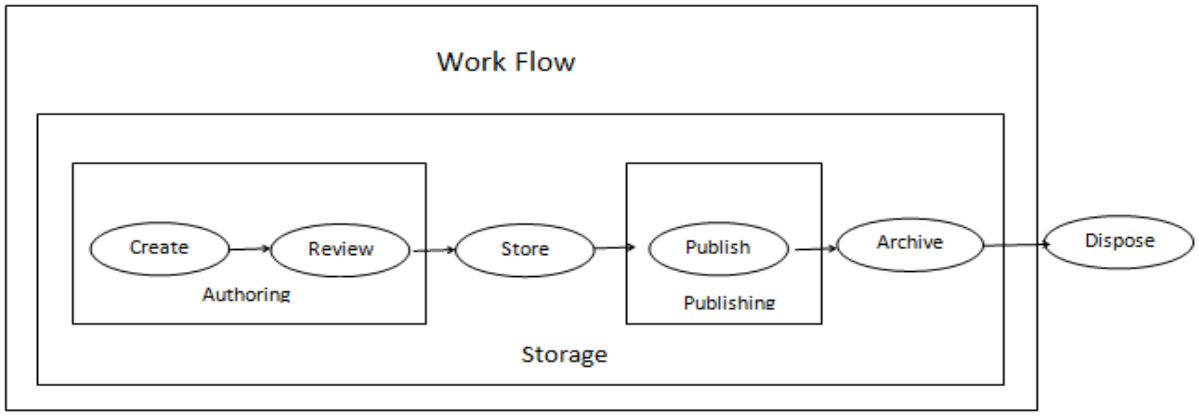

Figure 1: Content Management Process Model

3.1 UE and SE Interaction Framework

Both UE and SE processes focus on developing a "usable" and "functionally satisfactory" software product respectively. A usable software product should be easily understood and adaptable. A functionally satisfactory software product satisfies user requirements.

The activities of the UE and SE processes have common goals, and, are therefore, similar in nature. The objectives of these activities and their adopted techniques are, however, significantly different. For the UE and SE process activities to achieve their objectives in CMS, a considerable information exchange need exist among their activities. Ordinarily, the UE and SE processes are practiced as being exclusive and non-interacting. The underlying framework that guides content management development should promote the exchange of information among activities of the UE and SE processes by employing the processes as coordinated and synchronized practices. In this chapter, we discuss a paramount understanding of the exchange of information that normally should take place among the various phases of the UE and SE processes. This exchange of information helps in identifying essential interactions that must exist among the activities of the UE and SE processes. We conceptualize the exchange of information among activities of the UE and SE processes as interactions. To identify the interactions, we search for relationships among objectives of activities and introduce the concept of activity awareness, which implies continuous interactions among the activities of the UE and SE processes to exchange design information [4]

3.2 Interactions Between The SE And UE Processes

The identification of relationships among the objectives of both UE and SE can be useful in identifying the exchange of information that should exist among them. In this section, we present a high-level understanding of the exchange of information that should exist among the phases of the UE and SE processes. There exist some similarities and differences between the UE and SE processes. At a high level, UE and SE share the same objectives which are:

i. Seeking to understand the user's wants and need;
ii. Translating these needs into system requirements;
iii. Designing a system to satisfy these requirements; and
iv. Testing to help ensure their realization in the final product.

4. INFORMATION TRANSFER BETWEEN UE AND SE PROCESSES

As stated earlier, UE and SE have constituent activities that make them up, each of which has its own objectives. To improve the usability of CMS, there needs to be interaction between these activities as these activities need be carried out concurrently and in sync, that is, with respect to each other. Information exchange needs to take place between the UE and SE teams as the CMS project progresses, some of the important information transfer that need to take place between UE and SE processes in CMS development include:

4.1 Information about Constraints and Dependencies That Affect the System

Both usability and software engineering teams identify the constraints and dependencies that affect the CMS usability. The constraints and dependencies identified by the usability engineers may, however, have a different focus from those identified by the software engineers. Information about constraints and dependencies identified by each team should be useful to both teams during the later stages of their design. Therefore, this information should be exchanged among the activities of the UE and SE processes. [1][3]

4.2 Information about User Categories

Knowledge about user categories need be synchronized between the usability and software engineers. The usability engineers identify user categories during requirements and stakeholder analysis, while the software engineers do the same during their initial interaction with the customers, earlier in the vision and overview stage. This exchange of information about user categories should help to maintain a common understanding between the two teams with respect to the various possible user categories of the CMS.

4.3 Information about User Needs

The UE stakeholder analysis activity identifies the needs of the users that should be met by a usable system. These identified needs are useful to the software engineers during their design process. The software engineers identify the needs of the users through needs generation process performed in consultation with the stakeholders. User needs generated by the usability engineers during stakeholder analysis are available for reference during the SE needs generation. The SE team takes these needs into consideration during the implementation stage for easy creation, storage, editing, archiving and disposing of content on the content management systems. Also, the software engineers perform needs evaluation to distinguish the needs to be incorporated into the CMS from those to be left out due to non-necessity. These distinct needs should be conveyed to the usability engineers for use during the identification of the CMS functionalities.

4.4 Information about Candidate System Services

During the interaction design stage, the usability engineers identify the basic functionality to be offered by a software system. This basic system functionality is a high-level understanding of the activities to be supported by the CMS which include creating, reviewing, publishing, archiving and disposing of content materials. The usability engineers analyze current practices to identify new opportunities for improvement and identify the assumptions that constrain design activities. The knowledge about opportunities for improvement and the constraints imposed on the design are used to identify the basic functionality offered by the system under development. This information about these candidate activities to be supported by the system would be useful to the software engineers during their needs generation and should be conveyed to them.

4,5 Information about Supported Activities

The usability engineers design the details of activities and perform a refinement of the design through stakeholder involvement. They also transform current practices to new activities by constructing alternate activity design scenarios and by evaluating design features. The refinement of supported activities is performed using the point of view of system objects and the elaboration of activities through participatory design with stakeholders. The usability engineers verify the designed activities for coherence and completeness. The CMS development effort requires the usability and software engineers to have a common understanding of supported activities early in the project. Information about the identified supported activities obtained by the usability engineers should be conveyed to the SE team.

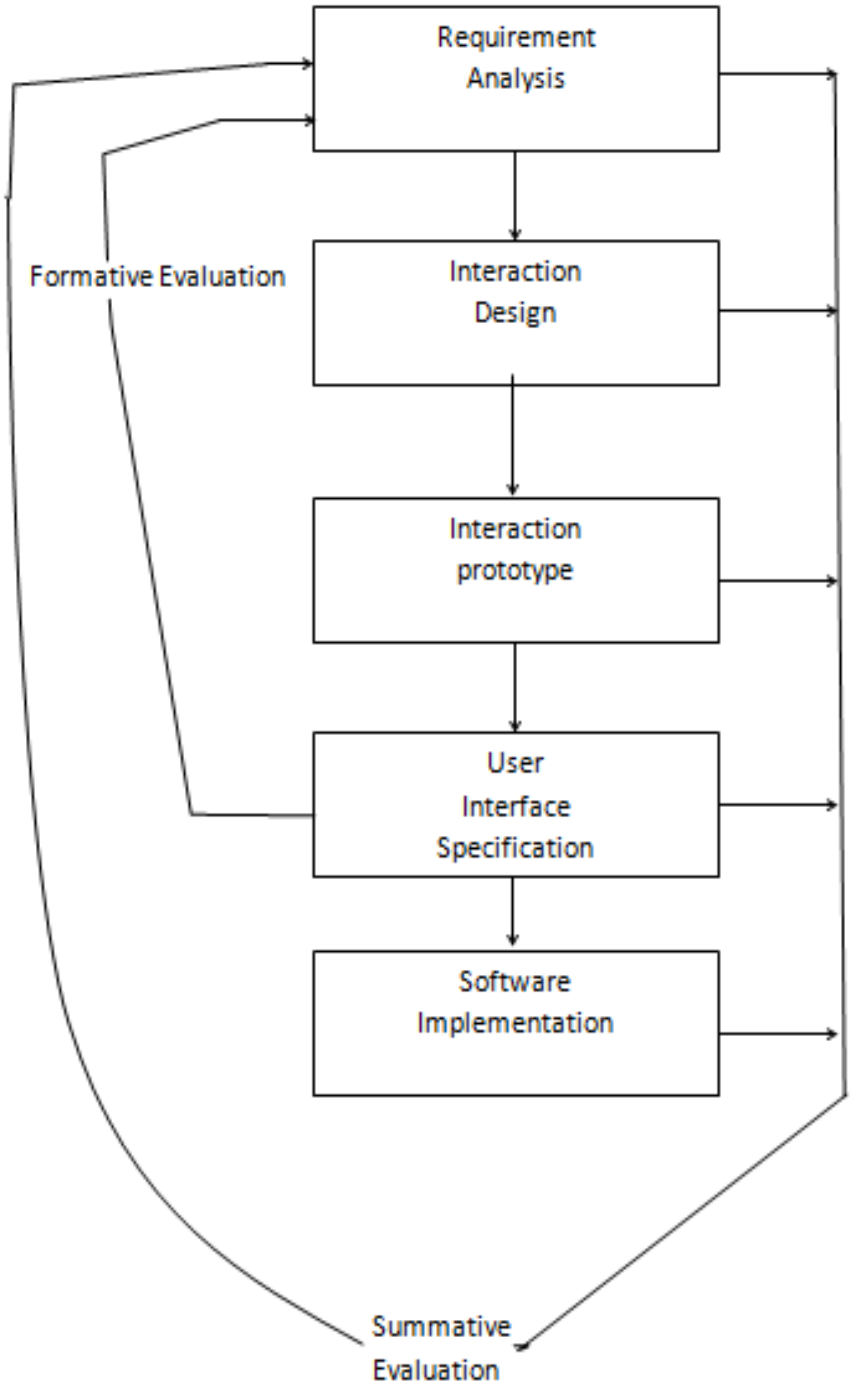

Figure 2: Scenario Based Usability Process Model

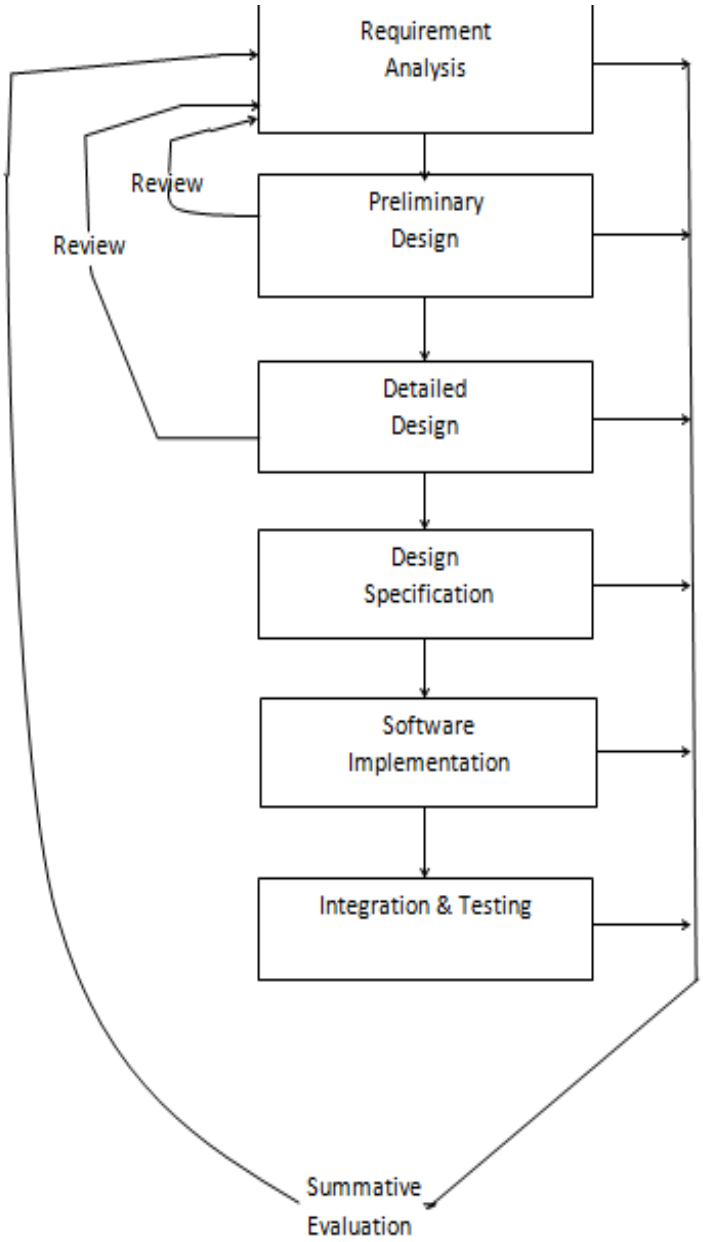

Figure 3: Software Requirement Generation Process Model

5. PROPOSED FRAMEWORK

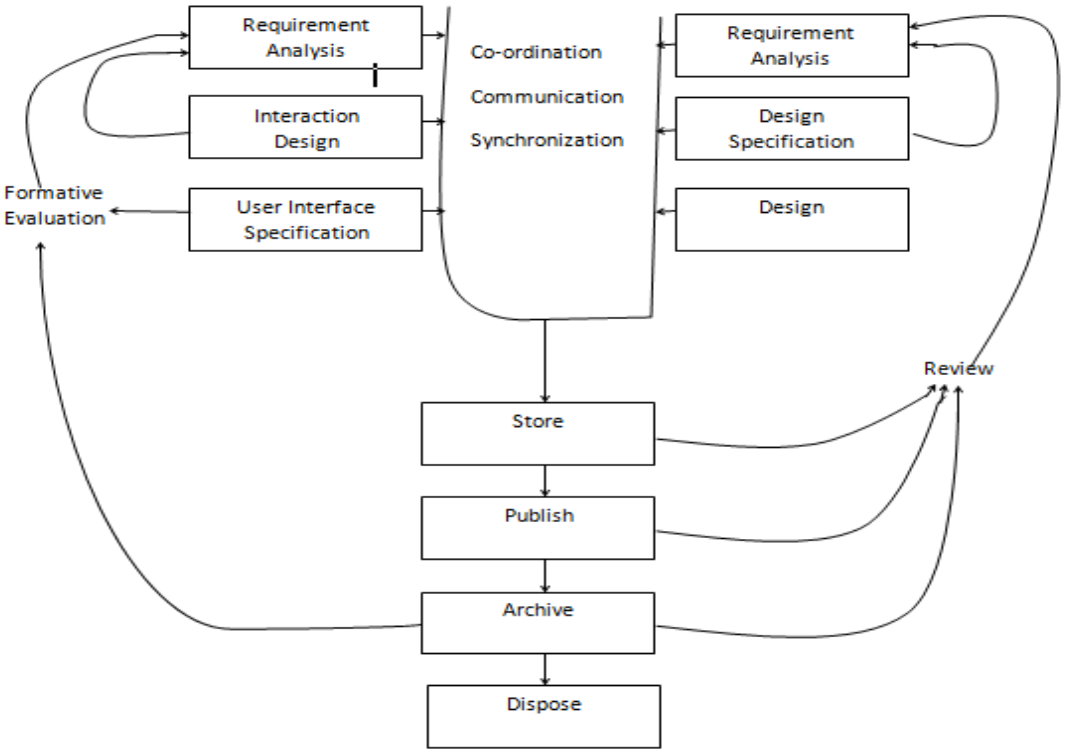

Figure 4: Proposed framework for CMS

5.1 Activity Awareness and Lifecycle Independence

Using the developed framework (figure 4.0), each developer role can see their own and the other's lifecycle status, activities, the iteration of activities, the timeline, techniques employed or yet to be employed, the artefacts generated or yet to be generated, and the mappings between the two domains if present. The view of each role would show only those activities that are relevant to that role. Each role views the shared design representation through its own filters so that, for example, the software engineers see only the software implications that result from the previously mentioned iterativeness in UE, but not the techniques used or the procedure followed. Similarly, if software engineers need iteration to try out different algorithms for functionality, it would not affect the usability lifecycle. Therefore, the process of iteration is shielded from the other role, only functionality changes are viewable through the UE filter. Each role can contribute to its own part of the lifecycle and the model allows each role to see a single set of design results, but through its own filter. Our framework places these connections and communication more on product design and less on development activities. This type of 'filter' acts as a layer of insulation, between the two processes, i.e. the process model helps isolate the parts of the development processes for one role that are not a concern of the other role. The layer needs to be concrete enough to serve the purposes, but not over specified so as to restrict the software design that will implement the user interface functionality. This prevents debates and needless concerns comparing processes and distrust on the other's techniques.

Due to the fact that the developed framework does not merge, but integrates, the two development processes, experts from one lifecycle need not know the language, terminology, and techniques of the other, and therefore can function independently.

5.2 User Interface and Functional Core Communication Layer

The framework advocates the need for the two development roles to specify a common communication layer between the user interface and the functional core parts. This layer is similar to the specification of the communication between the model and the other two parts (view and controller) in the Model View Controller (MVC) architecture [12]. This communication layer describes the semantics and the constraints of each lifecycle's parts. For example, the usability engineer can specify that an undo operation should be supported at a particular part of the user interface and that in the event of an undo operation being invoked by the user, a predetermined set of actions must be performed by the functional core. This type of communication layer specification, which will be recorded by our process model, allows the software engineers to proceed with the design by choosing a software architecture that supports the undo operation [1]. How the undo operation is shown on the user interface does not affect the SE activities. This type of early specification of a common communication layer by the two lifecycles minimizes the possibility of change on the two lifecycle activities. However, this common communication layer specification might change with every iteration and these changes should be made taking into account the implications they will have on the already completed activities and the ones planned for the future.

5.3 Coordination of Life Cycle Activities

Our framework coordinates schedules and specifies the various activities that have commonalities within the two processes. For such activities, the framework indicates where and when those activities should be performed, who the involved stakeholders are, and communicates this information to the two groups. For example, if the schedule says it is time for usability engineers to visit the clients/users for ethnographic analysis, the process model automatically alerts the software engineers and prompts them to consider joining the usability team and to coordinate for the SE's user related activities.

5.4 Communication between Development Roles

Another important contribution of this framework is the facilitation of communication between the two roles. Communication between the two roles takes place at different levels during the development lifecycle. The three main levels in any development effort are: requirements analysis, architecture analysis, and design analysis. Each of these stages results in a set of different artifacts based on the lifecycle. The framework has the functionality to communicate these requirements between the two domains. For example, at the end of UE task analysis the usability group enters the task specifications into the model and the SE group can view these specifications to guide their functional decomposition activities. At the end of such an activity, the SE group enters their functional specifications to the model for the usability people to cross check. This communication also helps in minimizing the effects of change and the costs to fix these changes. By communicating the documents at the end of each stage, the potential for identifying errors or incompatibilities increases as compared to waiting till the conventional usability specifications stage, which usually comes up after the SE team are 'done' with their input. This early detection of mismatches is important because the cost to fix an error in the requirements that is detected in the requirements stage itself is typically four times less than fixing it in the integration phase and 100 times less than fixing it in the maintenance stage [2].

5.5 Constraints and Dependencies

The design representation model incorporates automatic mapping features, which will map the SE and UE part of the overall design based on their dependencies on each other. For example, there exists a many-to-many mapping between the tasks on the user interface side and the functions on the functional side. In the event of identifying a new task after a particular iteration by the usability group, the design representation model will automatically alert the software group about the missing function(s) and vice versa. So when the software engineer tries to view the latest task addition, he is given a description that clearly describes what the task does and what the function should do to make that task possible. This way the developers can check the dependencies at regular time intervals to see that all the tasks have functions and vice versa and that there are no 'dangling' tasks or functions that turn up as surprises when the two roles finally do get together.

6. CONCLUDING REMARKS

Research on the development of a coordinated framework that incorporates UE and SE processes in CMS is motivated by the required exchange of information among activities of these processes during CMS development. The exchange of information ensures common design understanding between the UE and SE teams. The intent behind the identification of interactions between the UE and SE teams is to design a framework that incorporates the UE and SE processes. The differences in focus, methods, and terminology used by the UE and SE teams make integration of the two difficult. However, the UE and SE processes have to be performed in coordination during CMS development. In our research, we have identified the interactions necessary between the UE and SE teams, and have highlighted issues in the design of a coordinated development framework

7. CONTRIBUTIONS TO KNOWLEDGE

The coordinated development framework is required to facilitate interactions between the UE and SE teams. The definition of an interface between the UE and SE processes is the crux of this framework. The interface provides a much-needed structure to the communication between the UE and the SE teams during CMS development. Identification of exchange of information that should ideally exist among activities of the UE and SE process is a major contribution of this research. The exchange of information among major activities has been identified and visually represented at a high level. This exchange has also been detailed at a lower level of decomposition of major activities. The high-level representation serves the purpose of implementations that follow the same generic UE and SE activities, but have tailored processes. Implementations that do not follow a formal usability process, or follow the SBD for usability engineering, and, at the same time, use incremental software engineering also, can use the detailed exchange of information to tailor their own process.

REFERENCES

[1] Bass, L., John, B.E. "*Usability and Software Architecture.*" Behaviour and Information Technology, 20(5), 329 – 338 2001
[2] Boehm, B., *"A Spiral Model for Software Development and Enhancement"*, IEEE Computer, vol. 21, no. 5, pp.61-72.1988.
[3] B. Boiko. Content Management Bible. John Wiley & Sons, Hoboken, New Jersey, U.S.A., December 2001.
[4] Carmo, J. L. V. d. "*Web Content Management Systems; Experiences and Evaluations with the WebComfort Framework.*" Master's thesis, Instituto Superior Tecnico, Portugal 2006
[5] Carroll, J.M., Scenario-based Usability Engineering, 2002 http://old.sigchi.org/dis2002/documents/Tut-RossonCarroll.pdf
[6] Constantine, L., Biddle, R., Noble, J.; *"Usage-Centered Design and Software Engineering: Models for Integration"*, Proceedings of the ICSE 2003, pp.106-113.2003.
[7] Douglas, S., Tremaine, M., Leventhal, L., Wills, C. E., and Manaris, B. (2002), *"Incorporating human-computer interaction into the undergraduate computer science curriculum."* http://acm.org/citation.cfm?id=563419 12/11/2014
[8] Ferré, X., *"Integration of Usability Techniques into the Software Development*
[9] *Process"*, Proceedings of the International Conference on Software Engineering, pp.28-35.2003.
[10] IEEE, *IEEE Standards Collection: Software Engineering*, IEEE Standard 610.12- 1990, IEEE, 1993.
[11] Juristo, N., Windl, H., Constantine, L.; *"Special Issue on Usability Engineering in Software Development"*, IEEE Software, Vol. 18, no. 1, 2001.
[12] Krasner G.E., Pope S.T.: "*A description of the model-view-controller user interface paradigm in the smalltalk-80 system.*" Journal of Object Oriented Programming, 1(3):26-49, 1988
[13] Lewis, J.R. "*Psychometric evaluation of the post-study system usability questionnaire: The PSSUQ.*" Proceedings of the Human Factors Society 36th Annual Meeting, HFES, Santa Monica, CA pp 1259 – 1263 1992
[14] Madsen, K.H.: *Special Issue on "The Diversity of Usability Practices".* Comm. ACM, 42 (5) (1999)
[15] Matera M., Rizzo F., Carughi G.T.: *Web Usability: Principles and Evaluation Methods.* Dipartimento di Elettronica e Informazione, Politecnico di Milano Piazza Leonardo da Vinci, 32 – 2013 – Milano – Italy
[16] Mcdermid, J., Rook, P., *"Software Development Process Models"*, Software Engineer's Reference Book, CRC Press, pp.15/26 - 15/28.1993.
[17] Microsoft Enterprise Content Management 2006 http://download.microsoft.com/download/6/1/0/6105fe33-d27a-415a-a8a3-580c451ca06e/ECM_WhitePaper.doc
[18] Milewski, A., *"Software Engineering Overlaps with Human-Computer Interaction: A Natural Evolution"*, Proceedings of the International Conference on Software Engineering, pp.69-71.2003.
[19] Pressman, R. S., *Software Engineering: a Practitioner's Approach, 5th Edition*, McGraw-Hill, New York, NY, 2001.
[20] Pyla, P. S., Pérez-Quiñones, M. A., Arthur, J. D., Hartson, H. Rex; *"Towards a Model-Based Framework for Integrating Usability and Software Engineering Life Cycles"*, ACM: Computing Research Repository (CoRR), Technical report, cs.HC/0402036, 2004.
[21] P. Browning and M. Londes JISC TechWatch Report: Content Management Systems. 2001.

[22] Raccoon, L. B. S., *"The Chaos Model and the Chaos Life Cycle"*, ACM Software Engineering Notes, vol. 20, no. 1 January, 1995, pp.55-66.1995.
*[23]*Rosson, M. B., Carroll, J. M., *Usability Engineering: Scenario-based development of human-computer interaction*, Morgan Kauffman Publishers, 2002.
[24] Royce, W. W., *"Managing the Development of Large Software Systems: Concepts and Techniques"*, Proc. WESCON, 1970.
[25] SIGCSE Technical Symposium on Computer Science Education Conference Proceedings, 211-212.
[26] Sommerville, I., *Software Engineering, 5th edition* Addison Wesley Publishing Co, Reading, MA, 1996.
[27] Souer, J., van de Weerd, I., Versendaal, J., & Brinkkemper, S. (2007). Situational requirements engineering for the development of content management system-based web applications. Int. J. Web Eng. Technol., 3(4), 420.
[28] Standish, *Chaos Report, The Standish Group*, 1995,
[29] http://www.standishgroup.com/sample_research/chaos_1994_1.php