International Journal of Sciences: Basic and Applied Research (IJSBAR)

ISSN 2307-4531

http://gssrr.org/index.php?journal=JournalOfBasicAndApplied

Cyclic Website Reengineering Process Model Based On Website Auditing

Mohammad Othman Nassar[a]*, Feras Fares Al Mashagba [b]

[a,b] *Amman Arab University for Graduate Studies, Amman, Jordan.*

[a]*Email: moanassar@yahoo.com*

[b]*Email: ferasfm79@yahoo.com*

Abstract:

Websites like all web applications evolved very quickly, E-Companies are in highly competitive environment, they have to follow the customer needs and their competitors' performance. To survive in this highly competitive environment; companies can adopt many alternatives; amongst them is the continues corrections and enhancements for the company's website to meet the customer requirements. These corrections and enhancements occur much more frequent than those in traditional software, this mean that the need for reengineering will occur in shorter period of time compared to traditional software. Reengineering takes time, resources, and money ,so we want to make sure that we will do it in a way that will help us to reduce the need for website reengineering in the future, this can be accomplished if we use suitable software reengineering process model, that will address the needs of the website as a part of the software reengineering process, such model is not there yet ,so in this paper we will propose new cyclic process model that is suitable for reengineering websites, there are two main benefits for the proposed cyclic model; first: for each of the activities presented within the model it can be revisited, and for any particular cycle during the reengineering process, the process can be terminated after any one of these activities. Second: based on its cyclic nature; the model is suitable to be used with any approach to software re-engineering such as the Big Bang approach, incremental approach, and evolutionary approach.

Keywords: websites reengineering; process models.

----

* Corresponding author.
E-mail address: moanassar@yahoo.com.

1. Introduction

re-engineering can be done by taking existing legacy system that is now expensive and difficult to maintain or its architecture or implementation are now out of date, and updating and rearranging this system with current software or / and hardware technology will not achieve the required goals [9,10]. The difficulty came from the ability in understanding of the current system, because the system requirements documentation, its code, and design documentation are no longer obtainable, or they are now tremendously out of date.

Web based system development is different from traditional software development; it calls for knowledge and expertise from many different disciplines. Web based systems needs different Engineering Approaches, and a different development processes for them [3]. Small websites projects can evolve quickly, For example[4], the ICSE 2001Website , grew from a Small number of Web pages in May 1999 to a few hundred pages with over 3,000 links by May 2001, in just two years, 44 major Editions has been occurred . It is not usual for a traditional software project to go through that number of editions in two years.

websites are destined to become legacy software systems, the rate of change for web based systems, and especially for websites, compared to traditional software systems are relatively high, so they destined to become legacy software systems in short period of time. legacy systems are considered as an important asset, since they are containing many valuable information about current practices and business rules, they also containing corporate knowledge [11], thus it is not always reasonable to start developing a software system from scratch.

The need for reengineering clearly appears, because the current systems have become obsolescent in terms of their current architecture, the current platforms on which they run, and in their suitability and stability to support the system evolution to support the changing needs in the organization. Software reengineering is very important for recovering and reusing existing company's software assets, putting the high costs for the software maintenance under control, and establishing a clear and robust base for future software evolution.

Reengineering of websites differs from reengineering of traditional software [4], reengineering need time, resources, and money, so if we can do something when we reengineer the website to decrease the need for another reengineering process in the future, then we are reducing the cost for the company that own the website, this can be done by different ways, we will introduce them in the next context.

If we use appropriate system architecture for website then the future maintenance and evolution can be made easier and cheaper, in [7] the authors propose an approach that aim to restructure an existing website by adapting them to a controller centric architecture, this approach design a system architecture as a reference model for restructuring the Website to the new structure, also it defines a domain model to represent dependencies between Web pages in order to abstract current structure of the Website.

Other researchers [4] discuss different approaches that can serve as a starting point for the development of a Website architecture framework characterizing Websites for a better understanding of how Website reengineering can be made easier.

Some researchers provide a reengineering solution to specific type of problems, such as [5], they introduce a model based approach for reengineering web pages to solve the problem of today's websites , these websites are rarely de-signed and developed to be accessed throw a wide Varity of computing platforms, but this solution is limited to solve that specific problem only. In [1] the researchers provide a tool supported methodology to reengineer websites, but a gain this effort introduced to solve specific problem which is how to migrate data stored in static pages into databases.

The authors in [12] argues that websites are increasingly being accessed through a wide variety of computing platforms such as laptops, Internet Screen Phone, PDA, and cellular phones, then they propose a reengineering method for websites specialized to produce new user interfaces (UIs) for different contexts of use, thus creating a capability to produce UIs for different computing platforms.

In [14] the authors introduce an industrial case study related to incremental and iterative reengineering towards a Software Product Line (SPL). They applied the principles of agile development in the process of SPL reengineering, they analyze the reengineering process of the system's major component qualitatively and quantitatively, they focus on the initial investment needed, the trend for the investment, the quality improvement, and the returns on the investment.

In [15] the authors combined and integrate two reverse engineering approaches together. They combined the clustering and pattern detection to the process of reengineering the component-based software systems. And they show that this combination can detect and remove certain bad smells in a software system.

In [16] The SOAMIG Process Model is presented and described. This model is used in software migration. The model is divided into several phases and disciplines, each phase and discipline describe then organize the general migration activities. Usually the Activities in migration projects include legacy analysis, and legacy conversion which are considered as reengineering and reverse engineering activities [17].

In [18] the authors proposed and introduced Model-Based approach to Migrate Legacy Software Systems into the Cloud computing technologies to utilize their dynamic capacity and also to utilize their management capabilities.

As we can see from the available literature; there is no general reengineering software process model to guide the process of reengineering for websites, we believe this process model if created and presented will provide organization, stability, and control to the website reengineering activities that can, if left uncontrolled and unmanaged, become chaotic.

2. Software Reengineering Process Model

Pressman [6] in his book provides a software reengineering process model which can be shown in figure 1, this model as many models in software engineering proposed to provide a certain amount of useful structure to software reengineering work, the model shown in figure 1 is a cyclical model, which means that each of the activities presented as a part of the model may be revisited, and for any particular cycle, the process can terminate after any one of these activities, these activities will be discussed in the following lines.

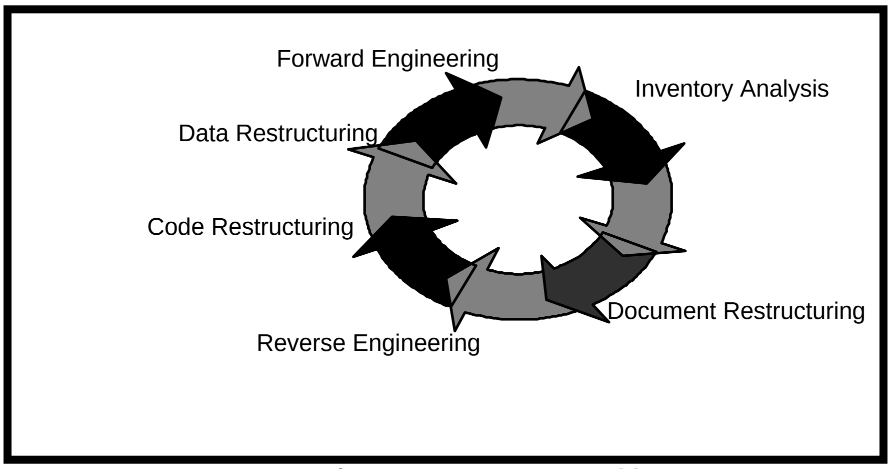

Figure 1: software reengineering process model

- **Inventory analysis:** This activity used for sorting active software applications by business criticality, longevity, current maintainability, and other local criteria, and helps to identify reengineering candidates in the organization .
    - **Document restructuring:** many legacy systems have designed with weak documentation; we have three options in this activity:
    - To live with weak documentation.
    - To update poor documents if they are used.
    - To fully rewrite the documentation for critical systems focusing on the "essential minimum".

- **Reverse engineering:** It is the process of recovering the software design, in this stage we analyze the program code to create a representation of the program at some abstraction level higher than the source code to help us in understanding the current code.
- **Code restructuring:** This activity is concerns with systems that have relatively solid and clear program architecture, but the individual modules (sub systems) where coded in a way that makes them difficult to understand, test, and maintain. The source code for those modules is analyzed and violations of structured programming practices are noted and repaired, finally; and after the repairing process; the revised code should be reviewed and tested.
- **Data restructuring:** In this stage the current data architecture and data models are defined,

the existing data structures are reviewed for quality; and if any violations are noted then they should be repaired.

- **Forward engineering:** This stage is designed to recovers the design information from existing source code, and the using this design information to reconstitute the existing system to improve its overall quality or performance.

The previous process model presented by [6] is not suitable for reengineering websites in its current state, this process model is designed for traditional software systems not for websites. The process of Website reengineering is still an open question, the reengineering of websites differs from reengineering of traditional software [4]. also the Web legacy systems has unique features different from traditional legacy software, such as single-instantiation, non-stop, fault-tolerant, and incrementally upgradeable [1]. So the model for reengineering traditional legacy software is not the perfect model to use unless certain changes implemented to that model to make it suitable for reengineering the websites. What we are going to do is to introduce a number of modifications to Pressman [6] model to make it suitable for reengineering websites. We will modify the activity called “inventory analysis” in the model, this modification will be in the activity description. We are trying to make the new proposed model suitable to be used with all approaches to software re-engineering; such as the Big Bang approach, incremental approach, and evolutionary approach. The new description for the activity will be based on the software re-engineering approach:

1- Inventory analysis for the Big Bang approach: this description is similar to the description presented by [6], the difference is that we add the company’s website as a new candidate within the reengineering candidates to find out if it is the suitable candidate for reengineering. So in this activity we should compare and sort the active software applications within the organization including its website by business criticality, longevity, current maintainability, and other local criteria.

2- Inventory analysis for the incremental approach: in the incremental approach; the system sections are reengineered and added incrementally as new versions of the system, those sections are added to satisfy the new system goals to identify reengineering candidates in the organization. So the inventory analysis in this approach should compare and sort the system sections within the organization website by business criticality, longevity, current maintainability, and other local criteria to find out what is the most critical section that should be chosen for the current increment.

3- Inventory analysis for the evolutionary approach: the evolutionary approach is an incremental approach, in this approach the sections for the old system are replaced with the new reengineered sections. The sections in this approach are chosen based on their functionality, not based on the structure of the existing system [14]. So the inventory analysis in this approach should compare and sort the system sections *based on their functionality* within the organization website to find out what is the most critical *functionalities* that should be chosen for the current increment.

This improvement alone will not be able to solve the problem, and make the model suitable for reengineering the websites.

3. Framework For Understanding The Importance Of Website Auditing

The following section is intended to modify the model in figure 1 to make it suitable for reengineering websites. When any organization faces so many problems in website maintenance and management; they usually decide to reengineer it. We know that reengineering needs time, resources, and money, so it can’t be done every day, so if it possible to consider other recommendations than those addressed by the website owner, then we are increasing the time for the next reengineering process. Website auditing can address the needs for a website in detailed and precise manner. website auditing as introduced by [2] provide a frame work for the auditing process in websites, this frame work will finally produce a set of recommendations and actions needed to be taken to suit the changing purposes, circumstances, and the web environment for the audited website.

Website audit plan as described in [2] consists of three phases, Phase(1) Information gathering, Phase (2) As Is Analysis, and Phase (3) Final Report and recommendations, we are going to integrate these phases to the model from [6] after the removal of inventory analyses activity as explained earlier. Website auditing provide valuable recommendations and actions for reengineering websites, this will allow us to make reengineering less frequent in the future because the outcome from the audit process as explained in [2] is a almost a complete list for the web site needs , this will save time, recourses, and money, because we will decrease the need for reengineering process in the future, we are going to introduce the three phases in more details to understand the importance of each phase.

- **Phase 1 –Information gathering:** This phase involves interviewing and/or surveying all the auditees involved in the development and maintenance of a Website. This information should focus on the requirements, the purpose of the Website, the business strategies and policies, and the information related to technical and technological issues. The information should be collected also from the site users where important information could be gathered. Finally Information on security and accessibility should be collected from the system administrators or the ISP depending on whether the site is hosted internally or externally.
- **Phase 2 –As Is Analysis:** The information collected in the first phase needs to be sorted and categorized in relation to the practices and quality of developmental processes. To remove any mistakes or misunderstandings; the analysis must double-check with the people consulted in the first phase.
- **Phase 3 –Final Report and Recommendations:** Based on the ‘As is Analysis’ we should choose the appropriate technologies and tools to use. The final report would include a full and detailed critique of the site, website features and recommendations to improve the site performance and functionality, and the re-engineering possibilities.

4. The Final Software Reengineering Process Model

According to [13] there are three different approaches to software re-engineering and they differ by the amount and by the rate of replacements that are made in the current software to get the target software. Those approaches are:

a- Big Bang approach: in this approach (figure 2) we replace the entire system at one time. The advantage in this approach is that the new system is migrated into a new environment all at once. So we do not need interfaces between the old and the new components. The disadvantage in this approach is that it is usually not suitable for large systems.

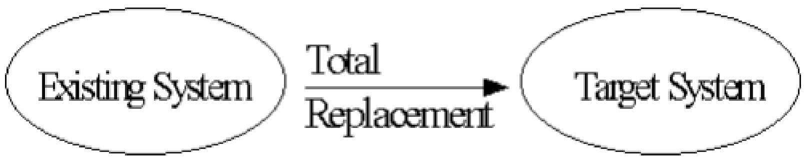

Figure 2: Big Bang approach

b- Incremental/Phase-out approach: the system sections in this approach are reengineered and added incrementally as new versions of the system, the sections are added to satisfy the new system goals. Figure 3 shows this approach. This approach has an advantages of the ability of producing the system faster and it is easier to trace errors, this is because the new components are clearly identified. the customer also can see progress and he can quickly identify any lost functionality [14]. Another advantage is that the changes to components that are not being reengineered have no direct impact on the current reengineered component. The disadvantage for the Incremental approach is that the system takes longer to complete with multiple Increments, this requires a very careful and difficult configuration control. Another disadvantage is that the entire structure of the system cannot be changed, only the structure within the specific component sections being reengineered.

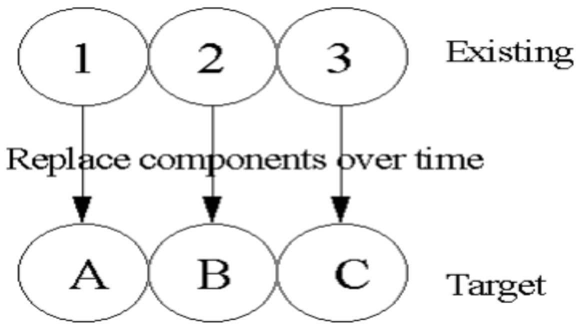

Figure 3: Incremental/Phase-out approach

c- Evolutionary approach : this approach (figure 4) is also an incremental approach, in this approach the sections for the old system are replaced with the new reengineered sections. The sections in this approach are chosen based on their functionality, not based on the structure of the existing system [14]. This approach allows developers to focus their efforts on identifying functional objects regardless of where the tasks are in the current system. The advantages are: the resulting modular design, and the reduced scope for a single component. This approach works well when converting to object-oriented. The disadvantage is that the similar functions should be identified in the entire existing system then they should be refined as a single functional unit, and this is a difficult task to do.

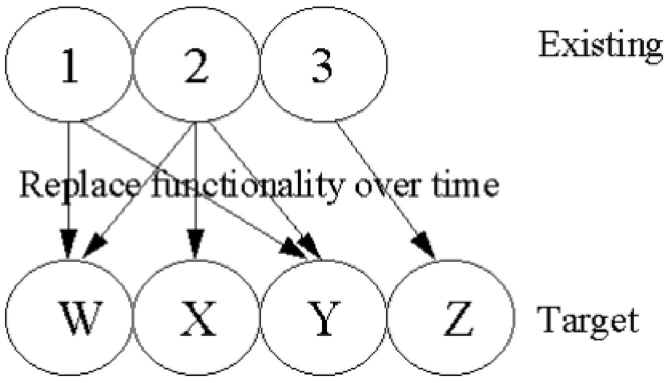

Figure 4: Evolutionary approach

we intentionally choose to create our final software reengineering process model based on two cyclical models; which means that each of the activities presented as a part of the model may be revisited, and for any particular cycle, the process can terminate after any one of these activities, this is so important to us because we want our new model to be suitable to be used with the three approaches to software reengineering presented earlier. The cyclical nature for the proposed model will allow it to be used in different situations.

we propose to integrate the previous frame work in section 3 for the websites auditing process within the proposed software reengineering process model, the auditing frame work will finally produce a set of recommendations and actions needed to be taken to suit the changing purposes, circumstances, and the web environment for the audited website. When these recommendations are integrated and implemented within the present reengineering process then we will have a complete, and real list of present and future website requirements. These requirements if implemented; then the need for reengineering will be far away in the future.

After the integration of the three phases of website auditing to the software reengineering process model presented in section 2, we present our proposed model in figure 5.

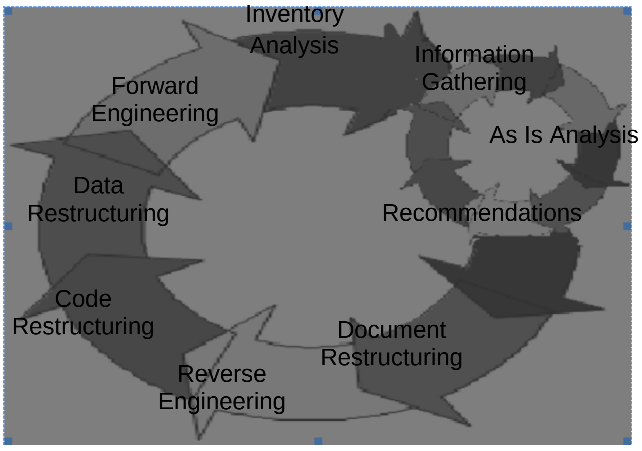

Figure 5: the proposed Software Reengineering Process Model.

5. Conclusions And Future Directions

Because Websites evolves very quickly, the need for reengineering will occur in short period of time compared to traditional software, and because reengineering takes time, resources, and money, we propose a reengineering process model that could reduce the need for reengineering in the future. Our proposed model is based on two widely used and tested models, the integration of both models and using the new model for reengineering websites needs to be tested, so as a future work we are going to use the proposed model and to report the results. We also encourage the researchers to do so. We also recommend using the AVISPA tool [8] to help in the evaluation process.

References

[1] Eichmann D., Evolving an engineered web. 1st International Workshop on Web Site Evolution (WSE '99), *Belgium*, October 1999.

[2] Deshpande Y., Chandrarathna A., and Ginige A.," Workshop on web engineering: Web site auditing: first step towards re-engineering", Proceedings of the 14th international conference on Software engineering and knowledge engineering, ACM, USA, July 2002.

[3] Ginige A., "Workshop on web engineering: Web engineering: managing the complexity of web systems development, Proceedings of the 14th international conference on Software engineering and knowledge engineering, ACM, USA, July 2002.

[4] Holger M. Kienle and Hausi A. M¨uller, Towards a Web Site Architecture Framework for Reengineering, University of Victoria, Canada, 2001.

[5] Laurent Bouillon, Jean Vanderdonckt, Jacob Eisenstein, Model-Based Approaches to Reengineering Web Pages, INFOREC Publishing House, Bucharest, July 2002.

[6] Pressman, R. S., Software Engineering: A practitioner’s Perspective, 6th Edition, McGraw-Hill, New York, 2005.

[7] Yu Ping, Kostas Kontogiannis, Refactoring Web sites to the Controller-Centric Architecture, Proceedings of the Eighth Euromicro Working Conference on Software Maintenance and Reengineering, IEEE , March 2004.

[8] Julio Ariel Hurtado, Maria Cecilia Bastarrica, and Alexandre Bergel. “Analyzing Software Process Models with AVISPA.” In Proceedings of the 2011 International Conference on Software and Systems Process, 23–32. New York, NY, USA: ACM. doi:10.1145/1987875.1987882.

[9] Chikofsky E J and Cross J 'Reverse Engineering and Design Recovery: A Taxonomy' IEEE Software (January 1990) pp 13-17

[10] Yu D 'A View On Three R's (3Rs): Reuse, Re-engineering, and Reverse-engineering'. ACM Sigsoft Software Engineering Notes Vol 16 No 3 (July 1991) p 69.

[11] Heygate R and Brebach G 'Memo to a CEO: Corporate Reengineering' The McKinsey Quarterly No 2 (Summer 1991) pp 44-55.

[12] L. Bouillon, J. Vanderdonckt, J. Eisenstein, "Model-Based Approaches to Reengineering Web Pages", International Workshop on Task Model and Diagrams for user interface design TAMODIA 2002.

[13] Radosevic, D.; Orehovacki, T. & Konecki, M. (2007). Web oriented applications generator development through reengineering process, Chapter 39 in DAAAM International Scientific Book 2007, B. Katalinic (Ed.), Published by DAAAM International, ISBN 3-901509-60-7, ISSN 1726-9687, Vienna, Austria, DOI: 10.2507/daaam.scibook.2007.39.

[14] Gang Zhang , Liwei Shen , Xin Peng , Zhenchang Xing , Wenyun Zhao, Incremental and iterative reengineering towards Software Product Line: An industrial case study, Proceedings of the 2011 27th IEEE International Conference on Software Maintenance, p.418-427, September 25-30, 2011 [doi>10.1109/ICSM.2011.6080809].

[15] M. von Detten and S. Becker, "Combining Clustering and Pattern Detection for the Reengineering of Component-based Software Systems," in Proceedings of the 7th International Conference on the Quality of Software Architectures. ACM, Jun. 2011, pp. 23-32.

[16] U. Erdmenger, A. Fuhr, A. Herget, T. Horn, U. Kaiser, V. Riediger, W. Teppe, M. Theurer, D. Uhlig, A. Winter, C. Zillmann, and Y. Zimmermann, "The SOAMIG Process Model in Industrial Applications," in Proceedings of the 15th European Conference on Software Maintenance and Reengineering. IEEE, 2011, pp. 339-342.

[17] De Lucia, A., Francese, R., Scanniello, G., Tortora, G.: Developing legacy system migration methods and tools for technology transfer, Software: Practice and Experience,38(13):1333–1364, 2008.

[18] S. Frey and W. Hasselbring, "Model-Based Migration of Legacy Software Systems to Scalable and Resource-Efficient Cloud-Based Applications: The CloudMIG Approach," in Proceedings of the First International Conference on Cloud Computing, GRIDs and Virtualization. Xpert Publishing Services, Nov. 2010, pp. 155-158.