A SOFTWARE REVERSE ENGINEERING METHODOLOGY FOR LEGACY MODERNIZATION

Oladipo Onaolapo Francisca[1] and Anigbogu Sylvanus Okwudili[2]
[1, 2]Department of Computer Science, Nnamdi Azikiwe University, Awka, Nigeria.

ABSTRACT

This paper identified that Legacy Systems have embedded within them, a large investment which ranges from low level code items or objects through to higher level business objects; made by the systems developers and/or owners. Most organizations would at one time or the other be confronted with the problem of migrating their legacy applications to new platforms in order to preserve previous investments and the software engineering community had been confronted with the problem of understanding legacy systems. A reverse engineering methodology for modernization of legacy systems based on a transformation paradigm aimed at preserving capital investments and saving production and maintenance costs were described in this paper. The transformation approach involved retaining and extending the value of the investments on the legacy system through migration and modernization of the subject system.

KEYWORDS: *Legacy application, system modernization, reverse engineering, artifacts, software capital investments*

I. INTRODUCTION

Most organizations would at one time or the other be confronted with the problem of converting their legacy applications and the software engineering community had been confronted with the problem of understanding legacy systems. Originally, legacy code was used to refer to programs written in old compilers; however, today's software developers predominantly use Object Oriented languages and this implied that tomorrow's legacy code is being written today, since object oriented programs are even more complex and difficult to comprehend, even when rigorously documented; most organizations end up with software that is even more obscure accompanied by insufficient design documentation [1].

Reverse engineering focuses on obtaining high-level representations of programs (probably written by another programmer). It typically starts with a low level representation of a system (such as binaries, plain source code, or execution traces), and try to distil more abstract representations from these such as, source code, architectural views, or use cases, respectively. The methods and technologies play an important role in many software engineering tasks, such as program comprehension, system migrations, and software evolution [2]. As observed by [3]; upward migration from procedural and structured programs to object-based technologies is very difficult and it is often impossible, to predict how the system is going to evolve during the process of development. Also Software systems; as artifacts will continually change over time, or become increasingly less useful, and the structure of evolving software will degrade unless remedial action is taken.

This paper described a methodology for modernization of legacy systems based on a transformation paradigm and an application to real-life software system. The transformation approach involves retaining and extending the value of the investments on the legacy system through migration and modernization of the subject system and extending it beyond the architectural barriers.

II. REVIEW OF RELATED WORK

A migration approach that involved the identification of software artifacts in the subject system and the aggregation of these artifacts to form more abstract system representations was presented in [4]. This approach led to the emergence of the RIGI tool. Early industrial experience showed that the

software engineers using Rigi can quickly build mental models from the discovered abstractions that are compatible with the mental models formed by the maintainers of the underlying software.
A program analysis approach using Synchronized Refinement as a systematic approach to detecting design decisions in source code and relating the detected decisions to the functionality of the system was described by [5] in 1994. The methodology; in addition to this approach, used approaches and representations typically found in the forward software development process, including a high-level, textual overview and graphical representations of data flows and file structures.
A case for legacy transformation, as opposed to complete discard of the legacy system based on the fact that existing application are a result of past capital investments for the organization was made by [6]. The work took the view that *J2EE* or *.NET* were suitable target platforms for transformation. The arguments in favour were based on technical and cost factors, on the fact that most automatic translation products target these platforms, on a growing skill-base in J2EE and .NET, making it easier to recruit staff, and on the availability of standard XML-based protocols for use by other applications, which facilitate the publication of application function to a network (usually referred to as 'Web Services').
A process to extract original architecture from a legacy system was developed in [3]. The methodology was a cognitive design recovery process and utilized several sources of domain knowledge to obtain relevant information about the application and get into the minds of the earlier developers with the aim of reconstructing the architecture. The re-constructed architecture is further compared with the original architecture to obtain the level of conformance.
A model for industrial large-scale software modification projects was described by [7]. The paper comprised a discussion on the process for problem analysis, pricing and contracting for such projects, design and implementation of tools for code exploration and code modification, as well as details of service delivery. These concerns were illustrated by way of a real-world example where a deployed management information system required an invasive modification to make the system fit for future use.
A report submitted by [8] described an enterprise framework that characterized the global environment in which system evolution takes place and provided insight into the activities, processes, and work products that shape the disciplined evolution of legacy systems. The work included exemplary checklists that identified critical enterprise issues corresponding to each of the framework's elements. Preliminary results indicated that the enterprise model was a useful tool for probing and evaluating planned and ongoing system evolution initiatives and the model served to draw out important global issues early in the planning cycle and provided insight for developing a synergistic set of management and technical practices to achieve a disciplined approach to system evolution. [9]; outlined a comprehensive system evolution approach that incorporated an enterprise framework for the application of the promising technologies in the context of legacy systems. The report revealed that the approach that one chooses to evolve software-intensive systems depends on the organization, the system, and the technology and concluded that there must be a framework in which to motivate the organization to understand its business opportunities, its application systems, and its road to an improved target system.
A white paper by [10] pointed out that Legacy systems contain useful business knowledge, but extracting that value is becoming increasingly difficult. The paper described the Cognizant approaches, methodologies, and its proven processes in restoring the legacy applications that are developed using various technologies that require specific answers.

III. METHODOLOGY

The model is an Architecture reconstruction process where the "as-built" architecture is obtained from an existing legacy system based on a modernization process sub-divided into many process steps which cut across understanding the goals of the evolutional changes that will have to be made to the legacy system and the actual modernization exercise (Figure 1). Modernization generally transforms a legacy system in in three phases: Initialization, Extraction and Modernization (Figure 2).
The first process required that the reverse engineering effort have a goal and a set of objectives or questions in mind before undertaking an architecture reconstruction project. An important goal might be for example, reusing part of the system in a new application and without these goals and

objectives, a lot of efforts could be spent on extracting information and generating architectural views that may not be helpful or serve any purpose.

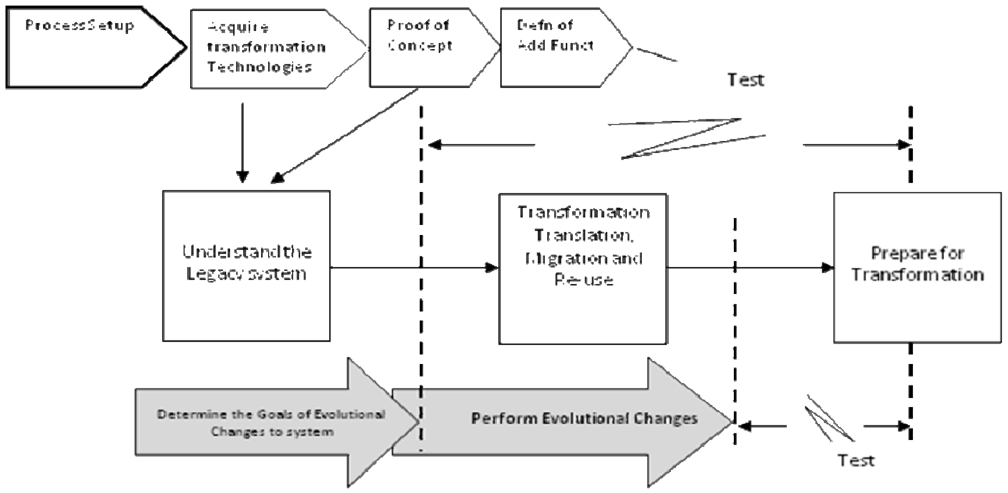

ProcessSetup
Acquire transformation Technologies
Proof of Concept
Defn of Add Funct
Test
Understand the Legacy system
Transformation Translation, Migration and Re-use
Prepare for Transformation
Determine the Goals of Evolutional Changes to system
Perform Evolutional Changes
Test

Figure 1: Evolution of a Modernized Legacy System

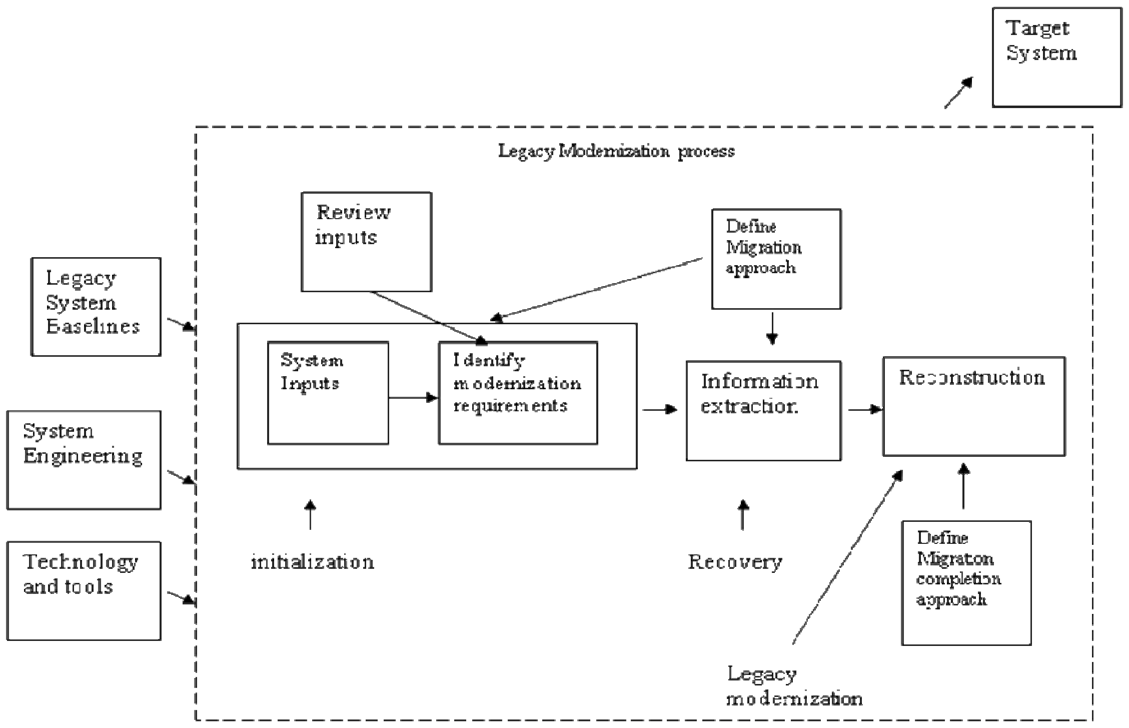


Figure 2. Modernization Model

Architectural extraction involved obtaining a high-level view of the legacy system after extracting helpful information and using the extracted information to generate a different view of the system in a higher level of abstraction. Other factors considered include the operating environments for the legacy system and the modernized version and the required support environments. The methodology also involved an evaluation of how the on-going enhancements to the legacy systems will be managed while the target system is phased in and the mechanisms to ensure that users will be able to fully convert to the new system at specific points.
The input to the system is the legacy system and the output is the modernized version with enhanced legacy assets. Other inputs required to bring about the modernization process include relevant technologies and tools and system engineering processes.

IV. RESULTS AND DISCUSSIONS

The authors in this work presented a multi-level legacy modernization roadmap that involved; information extraction; which in turn involves artifacts gathering from many sources, knowledge organization, analysis, and information abstraction involving aggregating components, relationships, synthesizing abstractions, building hierarchical mental models and ensuring that the subject system is not altered; but additional knowledge about the system is produced (Figure 2).
Within this multi-level view of transformations, the methodology was intended to depict architecture-level transformations as the context in which lower-level transformations subsist; and the various levels of abstractions identified were application, structure, function and implementation. The methodology defined application architecture and legacy modernization roadmap after evaluating various modernization options with ETCR (Effort, Time, Cost and Risk) technique to build multiple hierarchical mental models and subsystems based on software engineering principles (classes, modules, directories, cohesion, data & control flows, slices), design and change patterns, business and technology models, function, system, and application architectures, common services and infrastructure. The methodology also supported building on the foundations of a legacy asset, procuring enabling technologies for translation, data migration, and re-use, or a suitable partner identified to provide the technologies thereby leading to a smooth transition. The concept here is that the modernization project plan needs to gradually build up the knowledge about the existing and target applications, and create the knowledge for its extended support.

V. CONCLUSION

There will always be old software that needs to be understood. It is critical for the software industry to deal with the problems of software evolution and the understanding of legacy software systems. Bearing in mind that legacy systems are products of capital investments of a firm and since the primary focus of the industry is changing from completely new software construction to software maintenance and evolution, software engineering research and education must make some major adjustments. In particular, more resources should be devoted to software analysis in balance with software construction.
The authors in this paper had proffered a transformation of legacy software using modernization process. Legacy transformation project exhibits many of the characteristics of traditional development projects such as objectives setting, user involvement, testing, scheduling, monitoring, and so on. There are however some factors that differentiate it from the traditional software development and these includes:

- The solution is built on the foundations of a legacy asset, rather than starting with a discovery of business requirements. Of course there may be additional functional requirements to be added, but the usual procedure is to add this functionality after the transformation is complete.
- Enabling technologies need to be procured for translation, data migration, and re-use, or a suitable partner identified to provide the technologies.
- Because the legacy application is already part of today's business operations, a smooth transition is vital.
- Know-how needs to be built up over the course of the project so that support capabilities are in place on completion.
- Adjustments will be needed to existing development methodologies to ensure that the work is structured to fit the needs of a transformation project and delivers to business and technical objectives, schedule, and budget.

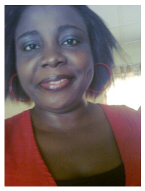
REFERENCES

[1]. Du Bois, B. (2005). Towards an ontology of factors influencing reverse engineering. In STEP '05: Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice, pages 74–80, Washington, DC, USA. IEEE Computer Society.

[2]. Osuagwu, O. E., Oladipo, O. F., and Banjo, C. (2008). Deploying Reverse Software Engineering as tool for Converting Legacy Applications in critical-sensitive systems for Nigerian Industries. In *Proceedings of the 22nd National conference and AGM of the Nigeria Computer Society Conference (ENCTDEV 2008).* 24-27 June

[3]. Oladipo, O.F. (2010). Software Reverse Engineering of Legacy Applications. Ph.D. Dissertation; Computer Science Department, Nnamdi Azikiwe University, Awka Nigeria. External Assessment, March, 2010.
[4]. Hausi A. M., Kenny W. Scott R. T. (1994). Understanding Software Systems Using Reverse Engineering Technology. Colloquium on Object Orientation in Databases and Software Engineering; The 62nd Congress of L'Association Canadienne Francaise pour l'Avancement des Sciences (ACFAS)"; May 16-17, Montreal, Quebec, Canada. Pp 240-252
[5]. Kamper, K. and Rugaber, S. (1994). A reverse engineering methodology for data processing applications Georgia Tech Technical Report , School of Information and Computer Science and Software Engineering Research Center.
[6]. Declan Wood (2002). Legacy Transformation. Edited and published by Club de Investigación Tecnológica San José, Costa Rica
[7]. Klusener, A.S., L¨ammel, R. Verhoef, C. (2004). Architectural modifications to deployed software. Science of Computer Programming 54 (2005) 143–211
[8]. Bergey, J.K., Northrop, L.M., Smith, D. B. (1997). Enterprise Framework for the Disciplined Evolution of Legacy Systems. Technical Report CMU/SEI-97-TR-007 ESC-TR-97-007. Reengineering Product Line Practice, Software Engineering Institute, Carnegie Mellon University. Pittsburgh.
[9]. Weiderman, N. H., Bergey, J. K., Smith, D. B., Tilley, S.R. Approaches to Legacy System Evolution. Reengineering Center Product Line Systems, Software Engineering Institute, Carnegie Mellon University. Pittsburgh.
[10]. Cognizant Technology Solutions (2001). Legacy Value Legacy Value Restoration Cognizant Technology Solutions, White Paper. Downloaded 10th August 2011 from http://www.cognizant.com/InsightsWhitepapers/LegacyValueRestoration.pdf

Authors' Biography



Oladipo Onaolapo Francisca is a Lecturer in the Department of Computer Science, Nnamdi Azikiwe University, Awka, Nigeria. Her research interests spanned various areas of Computer Science and Applied Computing. She has published numerous papers detailing her research experiences in both local and international journals and presented research papers in a number of international conferences. She is also a reviewer for many international journals and conferences. She is a member of several professional and scientific associations both within Nigeria and beyond; they include the British Computer Society, Nigerian Computer Society, Computer Professionals (Regulatory Council) of Nigeria, the Global Internet Governance Academic Network (GigaNet), International Association Of Computer Science and Information Technology (IACSIT), the Internet Society (ISOC), Diplo Internet Governance Community and the Africa ICT Network.

Sylvanus Okwudil Anigbogu is an Associate Professor and former head of the Department of Computer Science, Nnamdi Azikiwe University, Awka, Nigeria. His research interests are in the areas of Artificial Intelligence, Database Design and Management, Cyber security, etc and he has published his research works in several local and international journals. He is a fellow of the Nigerian Computer Society and a member of council of the Computer Professionals (Regulatory Council) of Nigeria.