INFORMATION SYSTEM INTEGRATION: A TECHNIQUE FOR LEGACY SYSTEM RE-ENGINEERING

EMOKPAE OSATOHANMWEN

Dept Of Computer Engineering, Institute Of Open Cast Mining And Technology.
Agbor Road Benin City Nigeria.
E-mail: osasdanny18@gmail.com

Abstract- Many organizations are still faced with the problems of converting applications written in legacy compilers such as COBOL, PASCAL ANS FORTRAN etc. into internet compliant applications. This has become necessary because most internet-based applications are written in C++, Java and Ms.Net. The objective of this paper is to show how information system integration can aid in the forward engineering of legacy system. Software re-engineering covers the examination and alteration of legacy system in order to rebuild it according to modem software engineering methods and technologies in a forward engineering process. Information system integration provides a means for both understanding and capturing about the application and its domain and re-developing the system based on change requirement. Information system integration helps to rebuild any legacy by adopting modern software engineering principles, methods and technologies, which allows systems to architecture engineers over the years have devises many modernization techniques. This paper evaluates the use of information system integration as a veritable technique for transformation of legacy system.

Keywords- COBOL, PASCAL, FORTRANS, C++, Java Ms.Net etc.

I. INTRODUCTION

Many organizations are still faced with the problem of converting applications written in Legacy compilers such as COBOL, Pascal and FORTRAN etc into internet compliant applications. This has become necessary because most internet-based applications are written in C++, Java, and MS.Net. The objective of this paper is to show how information system integration can aid in the forward engineering of legacy system. Software Re-Engineering covers the examination and alteration of Legacy systems in order to rebuild it according to modern software engineering methods and technologies in a forward engineering process. Information system Integration provides a means for both understanding and capturing knowledge about the application and its domain. And re-developing the system based on changed requirements. Information system integration helps to rebuild any Legacy by adopting modern software engineering principles, methods and technologies, which allow systems to share data and information, interoperate and reduce heterogeneous architecture. Engineers over the years have devised many modernification techniques. This paper evaluates the use of information system integration as a veritable technique for transformation of legacy systems.

Currently, in the world today organizations using software systems are under pressure of reducing cost of running the business and improving performance. Organizations have adopted information system integration for the re-engineering processes of the systems. Use of information systems to re-engineer legacy system has different advantages. Information systems help in maintaining integrity among multiple systems. Likewise, they ease maintenance processes to the software used. In addition, the systems help in streamlining business processes to the software used. There are five levels of integrating information system, which include data, data management, middleware, application and user interface (Bizer, 2003). Similarly, there are different methods used to approach integration at each level. Integration by common data storage is one method used to integrate. In this case, the data stored is moved to a new location. Secondly, it is uniform data access, where integration is performed to the data. Thirdly, it is integration by middleware in which functionalities, which are reusable, are provided to solve integration aspects. Fourthly, involves application integration where integration uses programs that can return integrated results, as well as accessing several data sources to the customer. These approaches are managed by several information systems. One of the foremost information systems is data warehouses where systems use the approach with common data storage for integration. Subsequently, FDMS (Federate Databases Management Systems) is an information system used. These systems provide solutions for uniform access to the data by integrating data from the local databases. The third information system is the use of web-services to integrate. This is where web services are used to integrate data using XML based messages. Likewise, data communicated in exchange must be agreed (Gold-Bernstein and William, 2005).

II. THEORETICAL FRAMEWORK

Software systems lifetime is remarkably inconsistent though some remain operational for many years. Nevertheless, failure of any service of a system may lead to serious effect. Change and continuous evolvement of organization activities affect software system used and this makes the legacy systems

complex. Legacy software systems are those systems that have become obsolete with time, though still remain valuable assets for a company because of its made-to-measure approach to meeting the company's needs. With a breakneck pace of the development of modern technologies sometimes it is difficult to remain abreast in terms of the company's infrastructure. In this case, a legacy systems need to be redesigned properly so that they can integrate with other applications so they exchange data and information. However, scrapping legacy systems and replacing them with more modern software involves significant risk. the various risk associated with replacing legacy systems with model (Warm *et. al.*, 2006) include; (a) rare conflict specification of the legacy system ways to specify new system functionally identical to a system in use of complete methods and business processes are inextricably intertwined. When system software changed, the processes also change (c) Business rules embedded in the software are on (d) The development of new software is extremely risky since unexpected problems arise with the new software system (Zhibiao and Martha, 1994).

Legacy system has different characteristics. Some of the characteristics include high cost of maintenance, lack of technical expertise, poor understanding, Backlog of change request, obsolete support software and critical business among others. There are different reasons in which legacy system are not designed to accommodate changes. One of the reasons is that the system was designed for immediate needs but not for long term. In addition, some constraints were satisfied by the legacy system development. Assessing the feasibility of determining the result of changes and making changes is necessary before attempting any change. To understand legacy system integration challenge different things need to be addresses. They include performance, operation, design, and functionality of the system. Likewise, types of changes need to be anticipated. Another problem emerging from legacy system integration is poor documentation due to the emergence of the new system integration is poor documentation due to the emergence of the new system capabilities. Different composition, physical distribution, functionality assignment, data access and protocol communication among others. Constraints in legacy system integration include semantics, connectors and components. During legacy system are engineering, documentation of code and the system is required. Definition of a subsystem and role of each system is required to be defined to integrate legacy system. Integration strategies are of two types i.e. non-intrusive and intrusive. Non-intrusive is used when information required is available from existing interface. When the code is understood and analyzed intrusive integration includes code and system restructuring.

III. ARCHITECTURAL PRINCIPLES

Legacy production systems are composed of four classes of components;

- Software/applications
- Data files (or sensor input for embedded systems)
- Platforms (hardware, degree of application coupling, operating systems, compilers, etc).
- Interfaces (user queries and updates originating from external or internal sources).

Components from each class can be re-engineered independently even though all four classes are interdependent. Each class contains unique characterizes, requires different re-engineering strategies, and can represent the entire re-engineering solution (instead of the traditional view of system re-engineering projects that must simultaneously encompasses multiple coolant class thus, information integration allows the flexibility to fix the real problem without having to re-engineer other components involved in ripple effects and interdependences.

Legacy system uses three approaches to integrate the system. They include business process, application integration and data integration (Warin *et. al.*, 2003).

IV. BUSINESS PROCESS:

A business process consists of related sets of activities that are performed by human and software actors according to business rules that may be more or less stringently applied. The connection between the software and human agents that perform a process is well integrated or coordinated when the process is efficient, accurate and appropriate to the task at hand from a mechanistic, human and organizational viewpoint. Moreover, the process goals must align with those of the organization as a whole. This is called *within-process integration.*

V. APPLICATION INTEGRATION

When sets of applications are integrated, each can call on the functions of the others. The principle of this approach is that applications contain the business logic of the enterprise, and the solution lies in preserving that business logic by extending the application's interfaces to interoperate with other or sometimes newer applications.

Mechanisms

Application integration is now supported by a wide range of middleware products. Initially, such integration was supported by RPC calls, and then by transaction managers, and now by application servers – each of these advances has incorporated the techniques of the previous mechanisms. IBM, BEA,

Microsoft, and Tibeo all provide many different variants of these mechanisms. However, these systems do not interface with humans directly and so only indirectly support business processes (i.e. through the applications that they integrate).

At a higher level, ERP systems achieve application integration because ERP vendor-developed applications that perform common business functions are united through a common database (also providing data integration – one of the selling points of ERP systems). The integration of the functional applications implies also that integration at the next layer, business processes, is achieved.

VI. DATA INTEGRATION:

The principle behind data integration is that the real currency of the enterprise is its data. The implied business logic in the data and metadata can be easily manipulated directly by applications in the new architecture of the enterprise.

Mechanisms

Simple data integration mechanisms are found in most programming languages. Class definitions in Java fulfill the same functions. At a higher level, data dictionaries also provide a systematic way of integration information with an emphasis on semantics. However, none of these mechanisms scale well when programs are written in different languages and query many different kinds of databases. The use of extensible Mark-up Language (XML) Schemas is the currently favoured solution for data integration. XML combines data and the description of the data in one place, which greatly simplifies integration (Ibbotson). In cross-enterprise integration, EDI has served a data integration role, providing a standard format for the exchange of common documents. Even here, the trend is toward XML.

Table 1.1 Framework for System Integration

Resource/Integration Need	Examples of Integration Mechanism	Enabling Environment	Infrastructure
Business Process (Internal and External to the firm)	Workflows, collaborative systems, SCM, CRM, Web Services.	Standards	
Applications	Interprocess Communication, RPC, Messaging, ERP, Web services.	Networks	System Architecture
Data	Data Dictionaries, Databases, XML	Platforms	

VII. IMPLEMENTATION ARCHITECTURE

There are different generic architectures used to re-engineering. Programmers use Java J2EE Connector architecture to integrate and connect application with back-end enterprise information system. OMG MDA is another generic architecture used in re-engineering. This architectural approach leads to the provision of necessary tools that integrate with different middle ware technologies (Patwardhan & Pedersen, 2006). Some of these middle ware technologies used include NET, SOAP, XML, EJB and CORBA among others. MDA architecture assures productivity, domain specificity, platform independence, and portability. In addition, JDBC approach lets user's access tabular data source from java programming language. This technology allows developers to take advantage of java platform for industrial strength application, which requires access to enterprise data (Farrar, 2003).

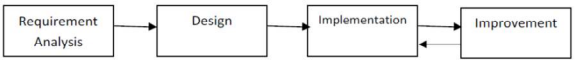


Fig. 4.1 Forward engineering Architecture

VIII. RE-ENGINEERING PROCESS

Different methods are used in the re-engineering process of legacy systems. In the Agreement method the developer and the customer comes to an agreement to design the system. If the outcome is positive, the feasibility study starts and the problem is defined. In the project method, the project is planned and access. In the method the cost of the designing a new system and the resources are determined. In the technical process, the system is designed, developed, tested, and implemented. This ensures the system designed works as required by the organization. In the evaluation process, the software is evaluated to ensure all the requirements are obtained.

CONCLUSIONS

Different approaches to the transformation of legacy systems including reengineering and upgrading exists. Before starting any legacy transformation effort, every possible option should be considered and business and strategic factors also need to be considered for ensuring long-term success. Current systems are the potential source of future legacy problems. To eliminate future legacy problems from current systems, systems should be built by using modular engineering and configurable infrastructure. Information system Integration helps to rebuild any Legacy by adopting modern software engineering principles, methods, and technologies. This allows systems to share data and information, interoperate and reduce heterogeneous architecture. Five levels of integrating information system include data, data management, middleware, application and user

interface. Different methods used to approach integration at each level includes Integration by common data storage, uniform data access, integration by middle ware and integration by application.

Classical system software re-engineering is being rejected by more organizations due to high risks, large upfront costs and an inability include mid-project changes in requirements or technology. In many instance, software system re-engineering is not even an option for an organization's large mission critical software systems.

REFERENCES

[1] Bisbal, J., Lawless, D., Wu., B., & Grimson, J. (1999). Legacy information systems: Issues and Directions. *IEEE Software*. 4.

[2] Bizer, C. (2003). *D2R Map* – A Database to RDF Mapping Language. Budapest, Hungary.

[3] Dittrich, P.Z. (2004). Three decades of data integration. *In 18th IFIP World Computer Congress.*

[4] Gold-Bernstein, B. & William, A. (2005). Enterprise integration: the essential guide to integration solutions, New jersey: Addison Wesley.

[5] Guha, L. (1990). Building large knowledge-based systems. New Jersey: Addison Wesley, reading.

[6] Guido, A. & Paiano, R. (2009). Semantic integration between information systems by ontology. 7(4).

[7] Gruber, T.R. (1993). A translation approach to portable ontology specifications. Knowledge acquisition 5.

[8] Farrar, S. (2003). A linguistic ontology for the semantic web. GLOT Internaitonal. 7(3).

[9] Jesus, B. & O.C.P. (2004). Using worldnet-based context vectors to estimate the semantic relatedness of concepts. *In proceedings of the EACL 2006 Workshop* 1.

[10] Rybinski (1987). On first-order-logic databases. ACM Transaction on Database Systems. 12(3), 325-349.

★★★