Data oriented and Process oriented Strategies for Legacy Information Systems Reengineering

Malleswara Talla, and Raul Valverde
John Molson School of Business, Concordia University, Montreal, Canada.
mrtalla@jmsb.concordia.ca, rvalverde@jmsb.concordia.ca

Abstract–**The legacy information systems often implement manual data updates for information obtained from external systems. The manual updates are cumbersome, error prone, and expensive. The legacy systems miss interfaces to external systems that could be used for automatic updates of system data. Moreover, the legacy systems also lack extensions to supplier or customer systems that are essential for creating supply chain relationships. This paper explores the data oriented and process oriented models of legacy systems, and discusses the details of systems development and evolution models mainly aiming at an ongoing reengineering of legacy systems. This paper proposes simple strategies for creating interfaces to external systems for automatic updates of data, and for adapting to the process evolution that requires a legacy information system to extend its communications with external systems that could help in creating successful supply chain relationships. These strategies can reshape a legacy system to be reengineered into a new enterprise information system whether the legacy system is of a data oriented model, or of a process oriented model.**

Key words–**Legacy system, Data oriented model, Process oriented model, System reengineering, Data structure.**

I. Introduction

A legacy system is an application that uses an obsolete hardware platform or software which is hard and expensive to maintain. Due to the fact that a legacy system is old, it is hard to find the skill set for maintaining the software or replacing hardware parts if a hardware failure occurs. Recent trend is the shortening life period of software systems and to quickly adapt to the new developments in software engineering; otherwise the systems become outdated very soon and become legacy systems. The huge investment in a legacy systems often compel reengineering and reuse of system components for evolution, maintenance and newer hardware platforms. Therefore, legacy systems range from traditional software systems to recent component-based systems. The modeling techniques that were used during legacy systems development also serve as building blocks during reengineering phase. This paper presents both traditional data and process oriented modeling techniques and proposes reengineering approaches that would prolong the life span of a legacy system.

The systems documentation often includes models that describe the requirements collected during initial system analysis phase of software development process [1]. These requirements models simplify the system complexity and help software maintenance process [2]. The requirement models are often remodelled during system architecture phase based on the software development methodology chosen. This paper focuses on traditional data and process models and attempts to propose few reengineering strategies.

The traditional data oriented reengineering focuses on migrating databases of legacy systems to current relational databases, enterprise data standardization, integration of disparate information systems, etc. [3]. The data oriented reengineering can also be conducted by focusing on improving data objects one by one in a pragmatic manner as presented in this paper. This approach eliminates the possibility of system failures, and minimizes the impact on overall system while reengineering the system. The process oriented reengineering considers each work activity and remodels a relevant part of the legacy system. The effort could be as simple as extending the legacy information system to an external process.

II. Traditional data model reengineering

The traditional software development methodologies include *data* models for organizing data and its documentation [4]. Data modeling is also called as *database modeling* because a data model is often implemented as a database [2], as depicted in *Entity Relationship Diagram (ERD)* that presents the data in terms of the entities and their relationships. An *entity* is an instance of a class of persons, places, events, objects, or concepts about which data is captured and stored. An entity is a single occurrence. An *attribute* is a descriptive characteristic of an entity. A compound attribute could include multiple attributes, in other words, a group of attributes or a data structure. A *relationship* is a natural association that exists between two or more entities. The relationship may represent an event for linking the entities. *Cardinality* defines a minimum and a maximum number of occurrences of an entity that may be related to a single occurrence of the other entity. Since all relationships among entities are bi-directional, cardinality must be defined in both directions for a relationship. Every entity must have a *key* which is an attribute, or a group of attributes, which assumes a unique value for each entity instance. It is sometimes called an identifier. A *primary key* is the candidate key that uniquely identifies an entity instance. A *foreign key* is the primary key of another entity that is duplicated in the entity for the purpose of identifying the relationship.

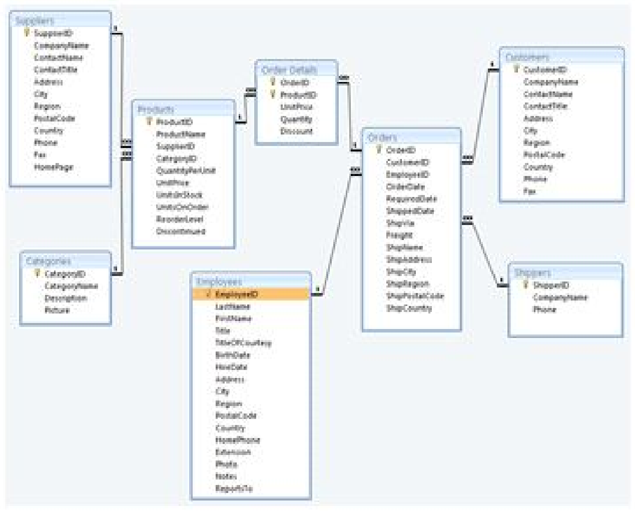

Figure 1. Sample ERD of an ordering system

Reengineering techniques for Data models

The business logic and data are interwoven that makes reengineering harder to accomplish. It is important to focus on the importance of data in the context of application as follows:

- Volume of data: number of entity instances,
- Security: business critical data,
- Mining: data for searching using a *keyword*,
- Complexity: interwoven data that makes it harder to isolate,
- Availability: of data within local database,
- Redundancy: duplicate data for simplifying systems development, or data which is also available on a different system.
- Heterogeneity: different in nature of data, etc.

We may like to conduct the reengineering of entire application at once, but it is cumbersome and it may fail, given the huge task of system reengineering. Conversely, if the reengineering is carried out in a pragmatic manner focussing one feature of the system at a time, reengineering will be successful. Here, we propose an approach for reengineering that focuses on reducing data duplication or redundancy of data. Traditional legacy systems focus on centralized data processing whereas the contemporary systems use distributed data processing architectures for parallel, specialized, and secure processing that increases the speed of response time. Traditional legacy information systems duplicate a lot of data into its local data base; such data requires periodic updates that introduce a margin of error as the data does not reflect upon real time data. The source of such data into the database may be often in the form of spread sheets or flat files. The legacy information system can be reengineered by changing the sourcing of such data to use the original data source over the network, instead of a lengthy spread sheets or data entry. The proposed reengineering technique in figure 2 uses the existing network applications that provide data in real time. This approach improves system performance and reduces the quantization error, introduced due to periodic manual updates of data.

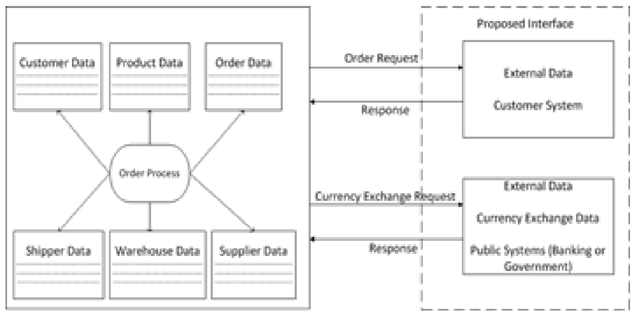

Figure 2. Legacy systems reengineering to use external data sources

In figure 2, two interfaces to different systems are proposed: (a) a customer interface that enables a customer to update an order processing part of a legacy system; (b) an external system interface where the currency rate fluctuations are automatically incorporated into the legacy system for eliminating manual data entry of currency exchange rates. Similarly, system improvements can be carried out one by one till the entire legacy system is reengineered.

III. Traditional process model reengineering

The traditional approach for information system requirements is to describe the business activities as processes carried out within and across organizations. Typically, a business process represents a set of workflows in a department of an organization, and a software application is designed to reflect upon the workflows as a set of interactions among entities and data stores, to eventually deliver a service. The data flows through the processes reflecting upon the procedures, policies and logic for implementing the software, as modeled in *data flow diagrams* (DFD) that serve as tools. In a process, work is performed in response to requests or data flows or conditions [4]. Unlike traditional data models, the process models depict concurrency of interacting processes as a set of data flows among entities and data-stores. The figure 3 reflects upon an order processing process that interacts with entities such as customers, warehouse, shipping, billing, etc. The repository of data are presented in figure 3, as data stores such as customers, inventory, and sales orders each of which maintains the necessary information.

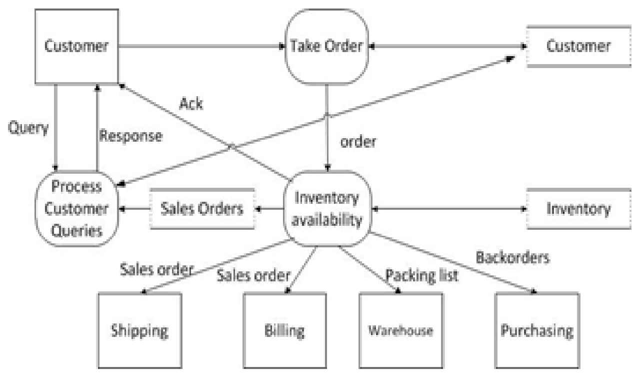

Figure 3. Sample Data Flow Diagram

The system exchanges data as inputs and outputs within its process environment. An *external agent* or an *external entity* defines a person, an organization unit, or a system internal or external to organization that lies outside the software system scope, but interacts with the system being studied [4]. External agents provide inputs into the system, and receive outputs from the system [2]. A data store is an inventory of data which is frequently implemented as a file or database [4]. It is *data at rest* compared to a data flow which is *data in motion*. Data stores depicted on a DFD store all instances of data entities depicted on an ERD. A data flow represents an input of data to a process, or the output of data from a process. It may also be used to represent the creation, reading, deletion, or updating of data in a file or database. A control flow represents a condition or non-data event that triggers a process. The following strategy for process modeling is proposed in [4]:

1. Draw a context DFD to establish initial system scope.

A *context diagram* defines the scope and boundary for the system. Because the scope of a software system frequently changes during the requirements and design phases of software development, the context diagram may also change to reflect upon the same.

2. Draw a *functional decomposition diagram* to partition the system into subsystems. A decomposition diagram shows the top-down functional decomposition or structure of a system. It provides an outline for arriving at data flow diagrams.

3. Create an *event-response* or *use-case* list for the system in order to determine what business events the system must respond to, and what responses are appropriate. Some of the inputs on the context diagram are associated with events.

4. Draw an event DFD or event handler for each event. For each event, illustrate any data stores from which records must be ‘read’ should be added to the event diagram. Data flows should be added to reflect upon the data read. Any data stores in which records must be created, deleted, or updated should be included in the event diagram. Data flows to the data stores should be named to reflect upon the nature of the update.

6.Merge event DFDs into a system diagram. The system diagram is said to be exploded from the single process on context diagram. The system diagram shows either: all of the events for the system on a single diagram, or all of the events for a single subsystem on a single diagram.

7. Draw detailed and primitive DFDs for the more complex event handlers. Each event process on the system diagram(s) must be exploded into either a procedural description or a primitive data flow diagram. For event processes that are not very complex – in other words, they are both an event and an elementary process, they should be described. The data flow diagrams show all the elementary process, data stores and data flows for single events

8. Document the logic of each elementary process using structured English.

9. Document data flows and processes in the data dictionary.

Reengineering techniques for Process models

Usually systems are not reengineered often, as a result, the same legacy software systems are *in-use* although an organization radically changed or reorganized its processes. An organization employs different ways of adapting to process changes, often conducting appropriate data entry *in to* and *out of* a legacy system to continue using it. This additional data entry is cumbersome and expensive; instead, it is profitable to gradually reengineer the legacy system.

A methodology for reengineering is presented in [5] focusing on business process, as follows:

- Prepare for reengineering,
- Map and analyze *As-Is* process,
- Design *To-Be* process
- Implement reengineered process, and
- Improve continuously.

While planning a legacy system reengineering, one should target process improvements such as (a) work balancing, (b) work-flow redesign, (c) eliminating non-value-added activities, and (d) revising administrative controls [6]. A process can be viewed as a set of sub-processes with performance criteria as follows:

- Distributed processing for faster response,
- Data redundancy across processes,
- Increased automation to integrate and minimize manual processes,
- Simplifying cross functional processes,
- Eliminating redundant sub-processes,
- New process creation and integration,
- Integrating current enterprise applications, e.g. Supply Chain Management (SCM), Customer Relationship Management (CRM), Enterprise Resources Planning (ERP), etc.

As discussed earlier, if the reengineering is carried out in a pragmatic manner focussing one feature at a time, reengineering will be successful. Here, we propose an approach for reengineering that focuses on integrating a new process, e.g. supplier process via web interface as presented in figure 4. It is possible to reengineer few parts of a legacy system by providing user interfaces to integrate new processes; for example, automatic inventory replenishment can be achieved by extending the system to integrate a supplier process as follows in Figure 4.

This approach in figure 4 integrates the external supplier processes and eliminates the order transaction processing delays. The legacy system can use the current business rule set and automatically generate new orders while respecting the lead time to suppliers.

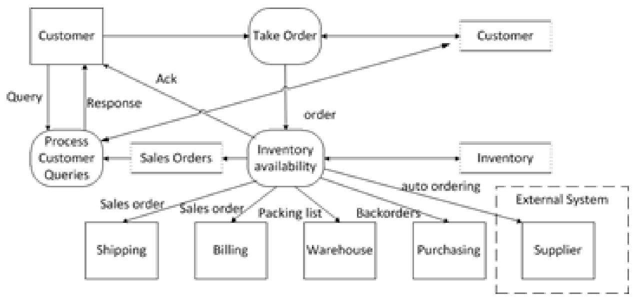

Figure 4. Legacy systems reengineering to integrate external supplier process

It also eliminates or simplifies the purchasing process. Although legacy systems use older software development methodologies and older hardware platforms, they still employ modular design, and it is possible to reengineer a business process, part by part, selectively.

IV. Traditional data models and process models

Both data models and process models are in fact interwoven sine every process has a set of data inputs and outputs following a set of actions. The process reengineering involves changing the business rules whereas data reengineering involves organizing, controlling and managing data. The legacy systems often interface external systems using manual processes, whereas contemporary systems use Intranets/Extranets. The data model is often described using ERD while the process model is described using DFD. The process model reengineering sometimes may require different inputs and may produce different outputs focusing mainly new business rules. The figure 5 presents the process and data models distinctly, identifying the internal and external processes and entities as well. While developing data-oriented reengineered model, both original and reengineered versions of data are kept and used interchangeably for smoother and gradual reengineering. It is sometimes required to switch from reengineered data to original data models during reengineered system development.

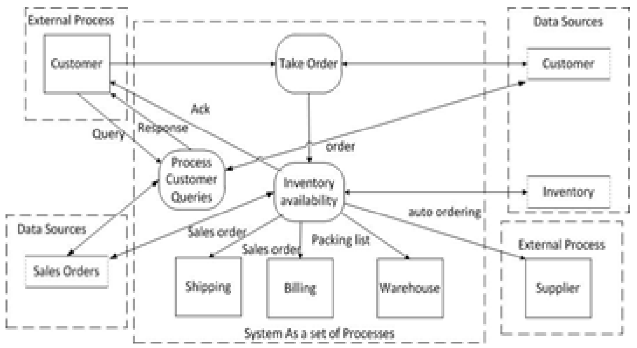

Figure 5. Data oriented vision versus Process oriented vision of a Legacy system

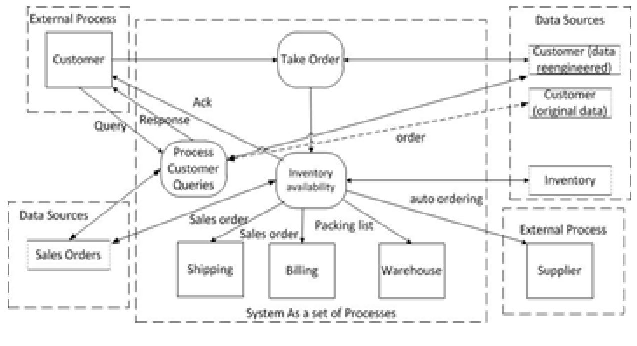

Figure 6. Co-existing original and reengineered data models of a Legacy system

Both original data model and reengineered data model to coexist, the “ifdef … endif, and ifndef ….. endif” pre-processor statements can be used during reengineered software development stage.

```
#ifdef ORIGINAL_DATA
  ….. Software with original data …….
#endif

#ifndef ORIGINAL_DATA
  ….. Software with reengineered data …….
#endif
```

The process reengineering improves the system to match the latest *ways of working*. The business rules and external processes often change and system needs to adapt to such changes quickly, in order to avoid manual update processes and related overhead. The data model reengineering focuses on improving the data structures and the associated implementation. Both data and process models adapting the business model to the most recent requirements effectively. Once the systems reengineering is completed, the systems staff needs to be trained so that they understand the features and the impact on business process. The data model reengineering is relatively easier to accomplish compared to process reengineering. However, well planned reengineering is often cost effective than developing new products that accomplish the same goals. It is also possible to model a system such that some parts of the system can be data oriented whereas the other parts can be of process model, which means the reengineering can be of both data and process models. Moreover, the system reengineering can be a gradual and an ongoing activity that keeps the systems maintenance staff well involved that proves the systems investment worthwhile.

Conclusions

In this paper, the traditional data and process modelling of legacy information systems and reengineering approaches were presented. The reengineering approach focused on systems evolution to benefit from *in-house* experience of systems team, able to maintain and evolve the software system. A fully reengineered system reflects upon the current business model and the feature set required sustaining the business in a medium term. It is argued that the traditional methodologies are not comprehensive enough to accurately model systems, then the only way to meet business requirements in an enterprise setting is to create several models in detail across the width and depth of the business. The paper recommends a gradual reengineering of system rather than a radical one shot, big budget, and expensive reengineering that can fail. The paper recommends a reengineering of both data model and process model in order to reorganizing the data and process models in a pragmatic way. The paper presented the approaches for reengineering with examples. The reengineering approaches proposed in this paper prolong the legacy information system lifespan in a cost effective manner.

References

[1] Jacobson, I., Christerson, M., Jonsson, P. & Overgaard, G., (1993). *Object-oriented Software Engineering: A Use Case Driven Approach,* Addison-Wesley, Wokingham, England

[2] Satzinger, J.W., Jackson R.B., and Burd S.D., (2002). *Systems Analysis ad Design in a Changing World,* Course Technology, Boston, Mass

[3] Alice H. Muntz, and Christian T. Ramiller (1994). "A Requirement-Based Approach to Data modeling and Reengineering", *Proc. of the 20th VLDB Conf. Santiago, Chile, 1994.* pp. 643-646.

[4] Whitten, J. L., Bentley D. L. and Dittman K.V. (2000). *Systems Analysis and Design Methods*, McGraw-Hill, New York.

[5] Subramanian Muthu, Larry Whitman, et al. (1999). "Business Process Reengineering: A Consolidated Methodology", *Proc. of The 4th Annual International Conference on Industrial Engineering Theory, Applications, and Practice*, Nov 1999.

[6] Victor Raj (2008). "Process to Product Orientation – A Re-engineering Experience from a Developing Country", *Proceedings of The 2008 IAJC-IJME International Conference,* ISBN 978-1-60643-379-9.

Malleswara Talla is a Professor (sessional) in the department of Decision Sciences & MIS at John Molson School of Business (JMSB), Concordia University, Montreal. He received B.Tech. degree from J.T.U. College of Engineering, Kakinada, India in 1979, a M.Tech. degree in 1981 from I.I.T., Kharagpur, India, and a Ph.D. degree from Concordia University, Montreal, in 1995 specializing in Computer Communications and Networks. He worked for Tata Consultancy Service (TCS), Bombay, and Societe Internationale de Telecommunications Aeronautique (S.I.T.A), Montreal for several years. Dr. Talla managed several projects involving data communications, computer networks, and business performance excellence. Dr. Talla is a member of Canadian Operations Research Society (CORS), Professional Engineers of Ontario (PEO), Institute of Electrical and Electronics Engineers (IEEE), Project Management Institute (PMI), The Association for Operations Management (APICS), and The Institute for Internal Controls (THEIIC). His teaching and research interests are mainly in Operations Management, Management Information Systems, Systems Re-engineering, Business Intelligence, Data Communications and Computer Network, Software Engineering and Evolution, Software Architecture, Design, and Development. Dr. Talla is a registered professional engineer in Canada.

Raul Valverde is working as a Professor in the department of Decision Sciences & MIS at John Molson School of Business (JMSB), Concordia University in Montreal. He holds a Bachelor of Science degree in Mathematics and Management from Excelsior College (US), a M. Eng. degree in Electrical & Computer Engineering from Concordia University, and a Ph.D. degree in Information Systems from University of Southern Queensland, Australia. He has more than 17 years of professional experience in IT/IS, mathematical modeling, and financial analysis. Dr. Valverde is a member of the Society of Management Accountants, Canadian Operational Research Society, Institute of Internal Controls, Forensic CPA society, Professional Engineers of Ontario and the Association for Operations Management. His main research interests include Supply Chain Systems, Risk Management, E-business, Information Security and Auditing, Accounting and Financial Information Systems, Fraud Detection and Reengineering. Dr. Valverde is a registered professional engineer and accountant in Canada.