Chapter 3
Traditional Data Oriented versus Process Oriented Reengineering of Legacy Systems

Raul Valverde
Concordia University, Canada

Malleswara Talla
Concordia University, Canada

ABSTRACT

The chapter presents data oriented and process oriented models of legacy systems. It discusses the details of systems development and evolution models mainly aiming at an ongoing reengineering of legacy systems. It proposes few strategies for reengineering of both data oriented model and process oriented models. The legacy systems often miss automatic interfaces to external systems, so the chapter presents a strategy focusing on automatic update of data of the system. Likewise, the chapter also presents a strategy for process reengineering in order to integrate external systems. Finally, a legacy system is envisioned as a comprehensive mix of both data and process oriented, while proposing a gradual ongoing reengineering of both data structures and process methods.

DOI: 10.4018/978-1-4666-0155-0.ch003
1. INTRODUCTION

A legacy system is an application that uses an outdated hardware or software platform. Due to the fact that a legacy system is very old, it is hard to find the skill set for maintaining the software or replacing hardware parts if a software bug is encountered or a hardware failure occurs. Recent trend is the shortening life period of systems to adapt to the new systems quickly; several systems are becoming outdated too soon and joining the group of legacy systems. The huge investment in a legacy system often compels reengineering and reuse of the system for evolution, maintenance and adaptation to component based software models and newer hardware platforms. Therefore, legacy systems range from traditional systems to recent component-based systems. The same modeling techniques that were used during legacy systems development also serve as building blocks during reengineering phase. This chapter presents both traditional and component-based modeling techniques and proposes few reengineering approaches that would prolong the life span of a legacy system that uses either of the modeling techniques.

The systems documentation includes models that describe the requirements collected during system analysis stage of software development process (Jacobson, I., Christerson, M., Jonsson, P. & Overgaard, G., 1993). These models simplify the system complexity and help software maintenance team to quickly understand what was done and aid in software maintenance or enhancement (Satzinger, J.W., Jackson R.B., and Burd S.D., 2002). These requirement models for a system are often remodelled during system architecture based on the methodology chosen. This chapter focuses on traditional data and process models and attempts to propose reengineering of such systems.

The traditional data oriented reengineering focuses on migrating databases of legacy systems to contemporary relational databases, enterprise data standardization, integration of disparate information systems, data quality assurance, etc. (Alice H. Muntz, and Christian T. Ramiller, 1994). The data oriented reengineering can also be conducted in a simplified manner by focusing on improving aspects of data one by one in a more practical manner as presented in this chapter. This approach eliminates the possibility of system failures, or minimizes the impact on overall system while reengineering the system as needed. On the other hand, the process oriented reengineering takes into account a sequence of work activities and remodels them in an attempt to revamp a part of the legacy system. The effort could be as simple as extending the legacy system to an external process as proposed this chapter.

2. TRADITIONAL DATA MODELS AND REENGINEERING TECHNIQUES

The traditional methodologies include data models of organization’s data, a technique for organizing and documenting a system’s data (Whitten, J. L., Bentley D. L. and Dittman K.V., 2000). Data modeling is often called database modeling because a data model is often implemented as a database (Satzinger & Jackson & Burd 2002), which is depicted in an Entity Relationship Diagram (ERD) for presenting the data in terms of the entities and their relationships.

An entity is a class of persons, places, objects, events, or concepts about which data is captured and stored.

An entity instance is a single occurrence of an entity. An attribute is a descriptive property or 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 business association that exists between one or more entities. The relationship may represent an event that links the entities or merely a logical affinity that exists between entities. Cardinality defines a minimum and a maximum number of occurrences of one